dynamicreports Documentation

Release 5.1.0

DynamicReports

Mar 15, 2021

DynamicReports

CHAPTER 1

Table of Contents

1.1 Home

1.1.1 Free and open source Java reporting tool

DynamicReports is based on JasperReports. It allows to create dynamic report designs and it doesn't need a visual report designer. You can very quickly create reports and produce documents that can be displayed, printed or exported into many popular formats such as PDF, Excel, Word and others.

1.1.2 Why should I use DynamicReports?

You can get a lot of benefits while using this library, such as

Easy to use

Very easy to use, which not only saves a lot of your time and money but also increases reporting productivity and decreases training time. Only a very small amount of code is needed to create a report and even more complex reports will be very clear, easy to maintain and understandable.

Dynamic report design

A design is created by a pure simple Java code. While it is generated at runtime, you can implement any logic in your code which decides how the report will look like, unlike the static reports (jrxml JasperReports templates) where the defined design cannot be changed at runtime. It's ideal for ad hoc reporting where it is needed to create reports dynamically.

Inherited report design

A design can even be inherited from another design. This is impossible in static report designs designed in visual designers.

No need for a visual designer

As mentioned above, a design is defined by a java code and hence you don't have to use a visual report designer. Visual designers are quite robust and require more effort and time for report creation. This library enables you to create reports in your favorite IDE at the same speed as you write a java code.

Ability to mix dynamic designs with static designs (jasper jrxml templates) You are able to design a part of the report in e.g. iReport and another part in DynamicReports. There are two ways how they can work together:

- static designs can be embedded in a dynamic design through subreports and placed into any band and any numbers of them
- static design as a base to which a dynamic design is added

1.1.3 Usage

DynamicReports is synchronized with a Maven central repository. For Maven projects you just add dependency to your maven configuration. In case you would like to use a development version, add a Sonatype Nexus snapshot repository to your maven configuration.

For non Maven projects you have to download a project with dependencies from the Download site.

To get started have a look at Getting started step-by-step tutorial which shows the basic usage of the tool. For more examples visit the Examples page where you will find a lot of useful examples.

In case you have questions regarding the usage, have a look at the Documentation and Forum. If you didn't get your answers there, feel free to post a question in the forum. To post in the forum you will need an account. If you don't have one, you can register at any time here.

If you think the library is missing a feature or has a bug, you can create a feature request or bug report in the Project tracker. Please provide a comprehensive description of your request when you are creating a new issue.

1.2 Getting Started

1.2.1 Overview

Creating a report with DynamicReports is very easy, take a look at the example below

```
1  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
2  public class Report {
3     private void build() {
4       try {
5         report()//create new report design
6           .columns(...) //adds columns
7           .groupBy(...) //adds groups
8           .subtotalsAtSummary(...) //adds subtotals
9           ...
10          //set datasource
11          .setDataSource(...)
```

(continues on next page)

(continued from previous page)

```
        //export report
        .toPdf(...) //export report to pdf
        .toXls(...) //export report to excel
        ...
        //other outputs
        .toJasperPrint() //creates jasperprint object
        .show() //shows report
        .print() //prints report
        ...
    } catch (DRException e) {
      e.printStackTrace();
    }
  }
  ...
}
```

See ReportBuilder class for all available features and JasperReportBuilder class for all available exports and outputs.

1.2.2 Simple report

Please note that this is a tutorial example and that this tutorial doesn't describe all features!

You will learn in this section how to create a simple report step by step. First we need to create an empty report desing, configure it and set a datasource. Afterwards the report will be ready to export into any format. It's important to remember the DynamicReports class, through which you will have available most of features. The class provides methods for building a particular piece of a report. Let's start!

Step 1 : Start

First define columns through DynamicReports.col.column(title, field name, datatype) (the field name must match with the name of a field contained in the datasource) and pass them into the report as follows:

```
.columns(//add columns
   //             title,     field name     data type
   col.column("Item",       "item",      type.stringType()),
   col.column("Quantity",   "quantity",  type.integerType()),
   col.column("Unit price", "unitprice", type.bigDecimalType()))
```

now define some text at the title and number of pages at the page footer as follows:

```
.title(cmp.text("Getting started"))//shows report title
.pageFooter(cmp.pageXofY())//shows number of page at page footer
```

DynamicReports.cmp.text(some text) - creates a component that shows a text DynamicReports.cmp.pageXofY() - creates a component that shows page X of Y Methods title(...) and pageFooter(...) allows to customize a particular report band by adding components to it

Step 2 : Styles

Each style can have a parent style from which it will inherit its properties. Empty style can be created by DynamicReports.stl.style()

```
StyleBuilder boldStyle          = stl.style().bold();
StyleBuilder boldCenteredStyle = stl.style(boldStyle)
                                    .setHorizontalAlignment(HorizontalAlignment.
↪CENTER);

StyleBuilder columnTitleStyle  = stl.style(boldCenteredStyle)
                                    .setBorder(stl.pen1Point())
                                    .setBackgroundColor(Color.LIGHT_GRAY);
report()
    .setColumnTitleStyle(columnTitleStyle)
    .highlightDetailEvenRows()
    .title(cmp.text("Getting started").setStyle(boldCenteredStyle))
    .pageFooter(cmp.pageXofY().setStyle(boldCenteredStyle))
```

Step 3 : Additional columns

You can very easy multiply, divide, add or subtract column of numbers by another column of numbers or by a number value

```
//price = unitPrice * quantity
TextColumnBuilder<BigDecimal> priceColumn = unitPriceColumn.multiply(quantityColumn)
                                                           .setTitle("Price");
```

Adding percentage of any column of numbers is simple DynamicReports.col.percentageColumn(title, column)

```
PercentageColumnBuilder pricePercColumn = col.percentageColumn("Price %",␣
↪priceColumn);
```

DynamicReports.col.reportRowNumberColumn(title) creates a column that shows row number

```
TextColumnBuilder<Integer> rowNumberColumn =
   col.reportRowNumberColumn("No.")
   //sets the fixed width of a column, width = 2 * character width
   .setFixedColumns(2)
   .setHorizontalAlignment(HorizontalAlignment.CENTER);
```

Step 4 : Group

We continue by adding a group as shown below

```
.groupBy(itemColumn)
```

Step 5 : Subtotals

Now we can try to sum a column of numbers. Subtotal of sum is created through DynamicReports.sbt.sum(column)

```
.subtotalsAtSummary(
     sbt.sum(unitPriceColumn), sbt.sum(priceColumn))
.subtotalsAtFirstGroupFooter(
     sbt.sum(unitPriceColumn), sbt.sum(priceColumn))
```

Method subtotalsAtSummary(...) allows to add subtotals to the summary band Method subtotalsAtFirstGroupFooter(...) will find first defined group and add subtotals to the group footer band

Step 6 : Charts

DynamicReports.cht provide methods for building charts. Category and series are required.

```
Bar3DChartBuilder itemChart = cht.bar3DChart()
                                 .setTitle("<a href="https://web.archive.org/web/
↪20180130194401/http://www.dynamicreports.org/examples/sales" title="Sales">Sales</a>
↪ by item")
                                 .setCategory(itemColumn)
                                 .addSerie(
                                    cht.serie(unitPriceColumn), cht.
↪serie(priceColumn));
Bar3DChartBuilder itemChart2 = cht.bar3DChart()
                                 .setTitle("<a href="https://web.archive.org/web/
↪20180130194401/http://www.dynamicreports.org/examples/sales" title="Sales">Sales</a>
↪ by item")
                                 .setCategory(itemColumn)
                                 .setUseSeriesAsCategory(true)
                                 .addSerie(
                                   cht.serie(unitPriceColumn), cht.serie(priceColumn));
```

Chart is a component and can be placed into any report band.

```
.summary(itemChart, itemChart2)
```

Step 7 : Column grid & Containers

Components inserted into the bands are arranged vertically, each component is below the previously added component. To arrange components horizontally it is needed to wrap these components with a horizontal container. Container is a component as well and therefore it can be added to any of the report bands.

```
.summary(
   cmp.horizontalList(itemChart, itemChart2))
```

Columns layout can be modified by a column grid. The layout is applied to the columns title, details and subtotals.

```
.columnGrid(
   rowNumberColumn, quantityColumn, unitPriceColumn,
   grid.verticalColumnGridList(priceColumn, pricePercColumn))
```

Step 8 : Hide subtotal

Subtotal for the group notebook is unnecessary because contains only one row. We need to change the group declaration and set the boolean expression condition on which depends whether subtotal is printed.

DynamicReports.exp.printWhenGroupHasMoreThanOneRow(itemGroup) creates a boolean condition which returns true when itemGroup has more than one row.

```
ColumnGroupBuilder itemGroup = grp.group(itemColumn);

itemGroup.setPrintSubtotalsWhenExpression(
           exp.printWhenGroupHasMoreThanOneRow(itemGroup));
.groupBy(itemGroup)
```

Step 9 : Title

First define a title style.

```
StyleBuilder titleStyle = stl.style(boldCenteredStyle)
                             .setVerticalAlignment(VerticalAlignment.MIDDLE)
                             .setFontSize(15);
```

DynamicReports.cmp.image() creates an image component
DynamicReports.cmp.filler() creates an empty component

```
.title(//shows report title
    cmp.horizontalList()
 .add(
   cmp.image(getClass().getResourceAsStream("../images/dynamicreports.png")).
↪setFixedDimension(80, 80),
   cmp.text("DynamicReports").setStyle(titleStyle).
↪setHorizontalAlignment(HorizontalAlignment.LEFT),
   cmp.text("Getting started").setStyle(titleStyle).
↪setHorizontalAlignment(HorizontalAlignment.RIGHT))
 .newRow()
 .add(cmp.filler().setStyle(stl.style().setTopBorder(stl.pen2Point())).
↪setFixedHeight(10)))
```

The defined filler creates an additional blank space between the title and the column header.
Setting top border of a filler draws the line at the bottom of the title.
Horizontal list, as previously mentioned, arranges components horizontally in one row but by calling the method row() a new horizontal list is created which is located at the bottom of the previous horizontal list.

Step 10 : Currency data type

Unit price and price column are currency types. Showing currency is possible by setting pattern to both columns (via method setPattern()), but the best practice is to create a custom data type and apply it to the columns. The custom data type then can be used in other reports.

```
CurrencyType currencyType = new CurrencyType();

TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice",␣
↪currencyType);
//price = unitPrice * quantity
TextColumnBuilder<BigDecimal> priceColumn     = unitPriceColumn.
↪multiply(quantityColumn).setTitle("Price")
                                                                    .
↪setDataType(currencyType);
private class CurrencyType extends BigDecimalType {
  private static final long serialVersionUID = 1L;

  @Override
  public String getPattern() {
    return "$ #,###.00";
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Step 11 : Detail row highlighters

```
ConditionalStyleBuilder condition1 = stl.conditionalStyle(cnd.greater(priceColumn,␣
↪150))
                                       .setBackgroundColor(new Color(210, 255, 210));
ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.smaller(priceColumn,␣
↪30))
                                       .setBackgroundColor(new Color(255, 210, 210));
 .detailRowHighlighters(
 condition1, condition2)
```

Condition1 is applied only if price is greater than 150 and sets background color of a row to green.
Condition2 is applied only if price is smaller than 30 and sets background color of a row to red.

Step 12 : Conditional styles

```
ConditionalStyleBuilder condition3 = stl.conditionalStyle(cnd.greater(priceColumn,␣
↪200))
                                       .setBackgroundColor(new Color(0, 190, 0))
                                       .bold();
ConditionalStyleBuilder condition4 = stl.conditionalStyle(cnd.smaller(priceColumn,␣
↪20))
                                       .setBackgroundColor(new Color(190, 0, 0))
                                       .bold();
StyleBuilder priceStyle = stl.style()
                              .conditionalStyles(
                                condition3, condition4);

priceColumn = unitPriceColumn.multiply(quantityColumn).setTitle("Price")
                      .setDataType(currencyType)
                      .setStyle(priceStyle);
```

Condition3 is applied only if price is greater than 200 and sets background color of a price to green.
Condition4 is applied only if price is smaller than 20 and sets background color of a price to red.

1.3 Syntax

Before you start creating reports with DynamicReports, you should consider which java syntax you will use. Java 5 added static imports that allow static methods or constants to be referenced without qualifying them with a class name. DynamicReports provides a lot of static methods of creating every part of a report and therefore you can consider using a static import for these methods. And DynamicReports also provides the abillity to use Method chaining (also known as Fluent interface). Both are useful for code readability, the ability to replace multiple calls to a class with a single one, and for invoking multiple method calls. Let's see in detail the difference between using and not using a static import and method chaining:

1.3.1 Classic java Syntax

Classic java syntax (without using a static import and method chaining)

```
JasperReportBuilder report = DynamicReports.report();
report.addColumn(Columns.column("Item", "item", DataTypes.stringType()));
report.addColumn(Columns.column("Quantity", "quantity", DataTypes.integerType()));
report.addTitle(Componens.text("Report title"));
report.setDataSource(dataSource)
report.show();
```

1.3.2 Using a static import

Using a static import for DynamicReports methods and method chaining

```
import static net.sf.dynamicreports.DynamicReports.*;
report()
.columns(
    col.column("Item", "item", type.stringType()),
    col.column("Quantity", "quantity", type.integerType()))
.title(cmp.text("Report title"))
.setDataSource(dataSource)
.show();
```

As you can see, static imports and method chaining improve readability of the code but they are not mandatory. It's only up to you which code syntax you will prefer to use.

All examples and tutorials in the documentation use static imports and method chaining to provide more readability of the code.

1.3.3 Examples

SimpleReport_ClassicSyntax, SimpleReport_Step01

1.4 Report Builder

1.4.1 JasperReportBuilder

The most used report builder for creating reports. It allows constructing and customizing the whole report content. A report consists of bands, columns, subtotals, groups, and other parts. Each part is created and configured using a particular builder method and it's passed to the report builder instance.

In the table below, you will find a list of all available builders:

Builder name	Using a static import for DynamicReports methods import static net.sf.dynamicreports .DynamicReports.*	Without a static import	Using a particular builder	Description
ColumnBuilders	col.*	DynamicReports.col.*	Columns.*	A set of methods of creating report columns
GridBuilders	grid.*	DynamicReports.grid.*	Grids.*	A set of methods of customizing columns layout
GroupBuilders	grp.*	DynamicReports.grp.*	Groups.*	A set of methods of creating report groups
SubtotalBuilders	sbt.*	DynamicReports.sbt.*	Subtotals.*	A set of methods of creating column subtotals
StyleBuilders	stl.*	DynamicReports.stl.*	Styles.*	A set of methods of creating and customizing styles
ComponentBuilders	cmp.*	DynamicReports.cmp.*	Components.*	A set of methods of creating components
ExpressionBuilders	exp.*	DynamicReports.exp.*	Expressions.*	A set of build in expressions
ConditionBuilders	cnd.*	DynamicReports.cnd.*	Conditions.*	A set of build in condition expressions
DataTypeBuilders	type.*	DynamicReports.type.*	DataTypes.*	A set of build in data types
ChartBuilders	cht.*	DynamicReports.cht.*	Charts.*	A set of methods of creating and customizing charts
ExporterBuilders	export.*	DynamicReports.export.*	Exporters.*	A set of methods of creating exporters
BarcodeBuilders	bcode.*	DynamicReports.bcode.*	Barcodes.*	A set of methods of creating barcodes
CrosstabBuilders	ctab.*	DynamicReports.ctab.*	Crosstabs.*	A set of methods of creating and customizing crosstabs

Examples Quick usage:

```
report()
.columns(
    col.column("Item", "item", type.stringType()),
    col.column("Quantity", "quantity", type.integerType()))
.title(cmp.text("Report title"))
.setDataSource(...)
.show();
```

Another example: SimpleReport_Step01

1.4.2 JasperConcatenatedBuilder

This report builder allows concatenating several separated reports into one single document. Each report starts on a new page with its own page dimension.

Examples Quick usage:

```
JasperReportBuilder report1 = ...;
JasperReportBuilder report2 = ...;
concatenatedReport()
.concatenate(report1, report2)
.toPdf(...)
```

Another examples: ConcatenatedReport1, ConcatenatedReport2

1.5 Data source

The important part of reporting is the data source which fills the report design with given data. Data can be retrieved from various sources such as a database, a xml file or using an interface JRDataSource. JasperReports itself has a lot of implementations of JRDataSource interface and basically implementation of this interface by you is quite often unneeded. Data in reports can be additionally sorted or filtered and some components such as chart, subreport, and crosstab can have their own data source if it's needed. Please refer to the link below for examples. Configuration options

Method	Description
setFilterExpression(DRIExpression<Boolean> filterExpression)	Sets a dataset filter expression. The expression must be a type of Boolean
sortBy(TextColumnBuilder<?> ... sortCols), sortBy(SortBuilder ... sorts), addSort(SortBuilder ... sorts)	Adds a sort field to the dataset

1.5.1 Database

In this type of data source, data are retrieved from a database. You only need a properly configured database connector and a SQL query.

Configuration options

Method	Description
setDataSource(String sql, Connection connection), setDataSource(QueryBuilder query, Connection connection), etDataSource(ResultSet resultSet)	Sets a database data source

Examples
Quick usage:

```
java.sql.Connection connection = ...;
String query = "SELECT * FROM ...";
report()
  .setDataSource(query, connection)
```

Another example: *DatabaseDatasourceReport*

1.5.2 Data source object

In this type, a JRDataSource object is expected as a data source. A good explanation of various implementations of data sources, as well as a few examples can be found using the following link: JasperReports data sources

Configuration options

Method	Description
setDataSource(JRDataSource dataSource), setDataSource(Collection<?> collection)	Sets a data source object

Examples
Quick usage:

```
List<JavaBean> data = new ArrayList<JavaBean>();
report()
  .setDataSource(new JRBeanCollectionDataSource(data))
```

Another example: *CollectionDatasourceReport*
Tags: datasource

1.6 Report Bands

Each report consists of several sections (bands). Each band has its particular location, signification, dimension, and can contain components. The height of the band is based on the component's dimension that it contains and can grow when the components are stretched.

Examples
Quick usage:

```
report()
.title(cmp.text("This is a title band"))
.pageHeader(cmp.text ("This is a page header band"))
.pageFooter(cmp.text ("This is a page footer band"))
```

Another example: BandReport

A report contains these bands:

1.6.1 Title

The band is printed on the first page and only once.

Configuration options

#	method	description
1	titleOnANewPage()	Prints the title band on a separate page
2	setTitleOnANewPage(Boolean titleOnANewPage)	Sets whether or not the title band is printed on a separate page
3	setTitleSplitType(SplitType splitType)	Sets the title band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
4	setTitleStyle(StyleBuilder style)	Sets a title band style
5	title(ComponentBuilder<?, ?> ... comp), addTitle(ComponentBuilder<?, ?> ... comp)	Adds components to the title band

1.6.2 Page header

The band is printed on each page at the top of the page.

Configuration options

#	method	description
1	setPageHeaderSplitType(SplitType splitType)	Sets the page header band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setPageHeaderStyle(StyleBuilder style)	Sets a page header band style
3	pageHeader(ComponentBuilder<?, ?> ... co), addPageHeader(ComponentBuilder<?, ?> ... components)	Adds components to the page header band

1.6.3 Page footer

The band is printed on each page at the bottom of the page.

Configuration options

#	method	description
1	setPageFooterSplitType(SplitType splitType)	Sets the page footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setPageFooterStyle(StyleBuilder style)	Sets a page footer band style
3	pageFooter(ComponentBuilder<?, ?> ... co), addPageFooter(ComponentBuilder<?, ?> ... components)	Adds components to the page footer band

1.6.4 Column header

The band is printed on each page at the bottom of the page and it's placed above the page footer band.

Configuration options

#	method	description
1	setColumnFooterSplitType(SplitType type)	Sets the column footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	floatColumnFooter()	Prints the column footer band at the bottom of the column
3	setFloatColumnFooter(Boolean floatColFooter)	Sets whether or not the column footer band is printed at the bottom of the column
4	setColumnFooterStyle(StyleBuilder style)	Sets a column footer band style
5	columnFooter(ComponentBuilder<?, ?> ... components), addColumnFooter(ComponentBuilder<?, ?> ... components)	Adds components to the column footer band

1.6.5 Group header

The band is printed for each data group. It's placed above the grouped data and between the column header and footer.

#	method	description
1	setGroupHeaderSplitType(GroupBuilder<?> grop, SplitType splitType)	Sets the group header band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setGroupHeaderStyle(GroupBuilder<?> group, StyleBuilder style)	Sets a group header band style
3	groupHeader(GroupBuilder<?> group, ComponentBuilder<?, ?> . . . components), addGroupHeader(GroupBuilder<?> group, ComponentBuilder<?, ?> . . . components)	Adds components to the group header band

1.6.6 Group footer

The band is printed for each data group. It's placed below the grouped data and between the column header and footer.

#	method	description
1	setGroupFooterSplitType(GroupBuilder<?> group, SplitType splitType)	Sets the group footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setGroupFooterStyle(GroupBuilder<?> group, StyleBuilder style)	Sets a group footer band style
3	groupFooter(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components), addGroupFooter(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components)	Adds components to the group footer band

1.6.7 Detail

The band is printed for each record row in the data source and it's placed between the column header and footer band.

Configuration options

#	method	description
1	setDetailSplitType(SplitType splitType)	Sets the detail band split type SplitType.PREVENT - prevent band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setDetailStyle(StyleBuilder style)	Sets a detail band style
3	detail(ComponentBuilder<?, ?> ... components), addDetail(ComponentBuilder<?, ?> ... components)	Adds components to the detail band

1.6.8 Detail header

The band is printed for each record row in the data source and it's placed above the detail band.

Configuration options

#	method	description
1	setDetailHeaderSplitType(SplitType type)	Sets the detail header band split type SplitType.PREVENT - prevent band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setDetailHeaderStyle(StyleBuilder style)	Sets a detail header band style
3	detailHeader(ComponentBuilder<?, ?> ... components), addDetailHeader(ComponentBuilder<?, ?> ... components)	Adds components to the detail header band

1.6.9 Detail footer

The band is printed for each record row in the data source and it's placed below the detail band.

Configuration options

#	method	description
1	setDetailFooterSplitType(SplitType type)	Sets the detail footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setDetailFooterStyle(StyleBuilder style)	Sets a detail footer band style
3	detailFooter(ComponentBuilder<?, ?> ... components), addDetailFooter(ComponentBuilder<?, ?> ... components)	Adds components to the detail footer band

1.6.10 Last page footer

The band is printed only on the last page at the bottom of the page.

Configuration options

#	method	description
1	setLastPageFooterSplitType(SplitType type)	Sets the last page footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	setLastPageFooterStyle(StyleBuilder style)	Sets a last page footer band style
3	lastPageFooter(ComponentBuilder<?, ?> ... components), addLastPageFooter(ComponentBuilder<?, ?> ... components)	Adds components to the last page footer band

1.6.11 Summary

The band is printed on the last page and only once.

Configuration options

#	method	description
1	summaryOnANewPage()	Prints the summary band on a separate page
2	setSummaryOnANewPage(Boolean sumOnANewPage)	Sets whether or not the summary band is printed on a separate page
3	summaryWithPageHeaderAndFooter()	Prints the summary band with the page header and footer
4	setSummaryWithPageHeaderAndFooter(Boolean summaryWithPageHeaderAndFooter)	Sets whether or not the summary band is printed with the page header and footer
5	setSummarySplitType(SplitType splitType)	Sets the summary band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
6	setSummaryStyle(StyleBuilder style)	Sets a summary band style
7	summary(ComponentBuilder<?, ?> ... comps), addSummary(ComponentBuilder<?, ?> ... cos)	Adds components to the summary band

1.6.12 No data

The band is printed only when the data source is empty. It's used to show the information that there are not any data in the report.

Configuration options

#	method	description
1	setNoDataStyle(StyleBuilder style)	Sets a no data band style
2	noData(ComponentBuilder<?, ?> ... comps), addNoData(ComponentBuilder<?, ?> ... comp)	Adds components to the no data band

1.6.13 Background

The band is printed on each page. It's mostly used for adding watermarks to the report.

Configuration options

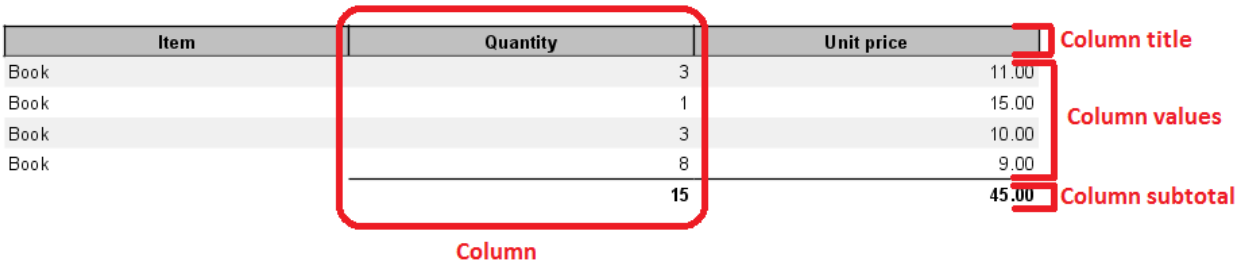
#	method	description
1	setBackgroundStyle(StyleBuilder style)	Sets a background band style
2	background(ComponentBuilder<?, ?> ... co), addBackground(ComponentBuilder<?, ?> ... components)	Adds components to the background band

Tags: layout

1.7 Columns

1.7.1 Overview

DynamicReports has an ability to display data in a multi-column layout. There is no limit in number of columns added to the report, only keep in mind that the more columns you add to the report, the less space will be available for each column. All columns automatically fit the available page width.



The picture above shows what is a column and how it is divided.

Configuration options of a column title

#	method	description
1	setTitle(String title), setTitle(DRIExpression<?> titleExpression)	Sets the column title
2	setTitleStyle(StyleBuilder titleStyle)	Sets the column title style
3	setTitleRows(Integer rows), setTitleFixedRows(Integer rows), setTitleMinRows(Integer rows)	This method is used to define the height of a column title. The height is set to the rows multiplied by height of the font
4	setTitleHeight(Integer height), setTitleFixedHeight(Integer height), setTitleMinHeight(Integer height)	Sets the height of a column title

Configuration options of column values

#	method	description
1	setStyle(StyleBuilder style)	Sets the column value style
2	setPrintWhenExpression(DRIExpression<Boolean> printWhenExpression)	Sets the print when expression. The expression must be a type of Boolean and it decides whether or not a column value will be print
3	setRows(Integer rows), setFixedRows(Integer rows), setMinRows(Integer rows)	This method is used to define the height of a column value. The height is set to the rows multiplied by height of the font
4	setHeight(Integer height), setFixedHeight(Integer height), setMinHeight(Integer height)	Sets the height of a column value
5	setPrintRepeatedDetailValues(Boolean printRepeatedDetailValues)	Specifies whether or not print a value if the value is the same as the previous value
6	setHorizontalAlignment(HorizontalAlignment horizontalAlignment)	Sets the column value horizontal alignment
7	setPattern(String pattern), setPattern(DRIExpression<String> patternExpression)	Sets the column value format pattern
8	setValueFormatter(DRIValueFormatter<?, ? super U> valueFormatter)	Sets the column value format expression
9	setHyperLink(HyperLinkBuilder hyperLink)	Sets the column value hyperlink
a	setStretchWithOverflow(Boolean stretchWithOverflow)	
b	addProperty(DRIPropertyExpression	Adds a jasper property to the column value

Configuration options of a column

#	method	description
1	setColumns(Integer columns), setFixedColumns(Integer columns), setMinColumns(Integer columns)	This method is used to define the width of a a column. The width is set to the columns multiplied by width of the character m for the used font
2	setWidth(Integer width), setFixedWidth(Integer width), setMinWidth(Integer width)	Sets the width of a column

Examples
Column examples

1.7.2 Text column

It is represented by a TextColumnBuilder instance and it is used to show values from the data source.

Builders

#	method	description
1	col.column(String fieldName, Class<T> valueClass), col.column(String title, String fieldName, Class<T> valueClass)	Creates a new column - title (optional) - the column title - fieldName - the name of the field - valueClass - the field value class
2	col.column(String fieldName, DRIDataType<? super T, T> dataType), col.column(String title, String fieldName, DRIDataType<? super T, T> dataType)	Creates a new column - title (optional) - the column title - fieldName - the name of the field - dataType - the field data type
3	col.column(FieldBuilder<T> field), col.column(String title, FieldBuilder<T> field)	Creates a new column - title (optional) - the column title - field - the field definition
4	columnInstance.add(TextColumnBuilder<? extends Number> column), columnInstance.add(Number number)	Creates a new column by adding a value or a column value to the columnInstance column
5	columnInstance.subtract(TextColumnBuilder<? extends Number> column), columnInstance.subtract(Number number)	Creates a new column by subtracting a value or a column value from the columnInstance column
6	columnInstance.multiply(TextColumnBuilder<? extends Number> column), columnInstance.multiply(Number number)	Creates a new column by multiplying the columnInstance column with a value or a column value
7	columnInstance.divide(int scale, TextColumnBuilder<? extends Number> col), columnInstance.divide(int scale, Number number)	Creates a new column by dividing the columnInstance column with a value or a column value

Examples
Quick usage:

```
report()
.columns(
    col.column("Item", "item", type.stringType()),
    col.column("Quantity", "quantity", type.integerType()))
.setDataSource(...)
```

Another example: *ColumnDataTypesReport*

1.7.3 Expression column

It is represented by a TextColumnBuilder instance and the displayed values are defined in an expression.

Builders

#	method	description
1	col.column(DRIExpression<T> expression), col.column(String title, DRIExpression<T> expression)	Creates a new expression column - title (optional) - the column title - expression - the value expression

Examples
Quick usage:

```
report()
.columns(
    col.column("Expression column", new ExpressionColumn()))
.setDataSource(...)
private class ExpressionColumn extends AbstractSimpleExpression<String> {
public String evaluate(ReportParameters reportParameters) {
    return ...;
}
}
```

Another example: *ExpressionColumnReport*

1.7.4 Percentage column

It is represented by a PercentageColumnBuilder instance. It calculates percentage values from the field or column values.

Builders

#	method	description
1	col.percentageColumn(ValueColumnBuilder<?, ? extends Number> column), col. percentageColumn(String title, ValueColumnBuilder<?, ? extends Number> col)	Creates a new percentage column from the column values - title (optional) - the column title - column - the column definition
2	col.percentageColumn(String field, Class<? extends Number> valueClass), col.percentageColumn(String fld, String s1, Class<? extends Number> valueClass)	Creates a new percentage column from the field values - title (optional) - the column title - fieldName - the name of the field - valueClass - the field value class
3	col.percentageColumn(FieldBuilder<? extends Number> field), col.percentageColumn(String title, FieldBuilder<? extends Number> field)	Creates a new percentage column from the field values - title (optional) - the column title - field - the field definition

Configuration options

#	method	description
1	setTotalType(PercentageTotalType totalType)	Sets the total type. Has effect only when the report contains at least one group
2	setTotalGroup(GroupBuilder<?> totalGroup)	Sets the total group. Has effect only when the report contains at least one group

Examples

Quick usage:

```
1 TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.
  ↪integerType());
2 PercentageColumnBuilder quantityPercColumn = col.percentageColumn("Quantity [%]",␣
  ↪quantityColumn);
3 report()
4 .columns(
5     quantityColumn, quantityPercColumn)
6 .setDataSource(...)
```

Another example: *PercentageColumnsReport*

1.7.5 Row number column

It is represented by a TextColumnBuilder instance and displays row numbers. Builders

#	method	description
1	col.columnRowNumberColumn(), col.columnRowNumberColumn(String title)	Create a new row number column, the row number	is reset on each new column - title (optional) - the column title
2	col.pageRowNumberColumn(), col.pageRowNumberColumn(String title)	Create a new row number column, the row number is reset on each new page - title (optional) - the column title
3	col.reportRowNumberColumn(), col.reportRowNumberColumn(String title)	Creates a new row number column - title (optional) - the column title

Examples
Quick usage:

```
report()
.columns(
    col.reportRowNumberColumn("Report row"))
.setDataSource(...)
```

Another example: *RowNumberColumnsReport*

1.7.6 Boolean column

It is represented by a BooleanColumnBuilder instance and shows a boolean value either as a text or as an image.

Builders

#	method	description
1	col.booleanColumn(String field), col.booleanColumn(String titl, String fld)	Creates a new boolean column - titl (optional) - the column title - fld - the name of the field
2	col.booleanColumn(FieldBuilder<Boolean> fld), col.booleanColumn(String title, FieldBuilder <Boolean> fld)	Creates a new boolean column - title (optional) - the column title | - fld - the field definition
3	col.booleanColumn(DRIExpression<Boolean> ex), col.booleanColumn(String title, DRIExpression <Boolean> expression)	Creates a new boolean column - title (optional) - the column title - expression - the boolean value expression

Configuration options

#	method	description
1	setComponentType(BooleanComponentType booleanComponentType)	Sets the boolean presentation type. BooleanComponentType.TEXT_* - shows a text value BooleanComponentType.IMAGE_* - shows an image
2	setImageDimension(Integer we, Integer ht), setImageWidth(Integer width), setImageHeight(Integer height)	Sets the boolean image dimension. Has effect only when the boolean value is presented as an image

Examples

Quick usage:

```
report()
.columns(
    col.booleanColumn("Boolean", "boolean"),
col.booleanColumn("Boolean", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪STYLE_1))
.setDataSource(...)
```

Another example: *BooleanColumnReport*

1.7.7 Component column

It is represented by a ComponentColumnBuilder instance and is used to display custom components (e.g. images or complex content) in columns.

Builders

#	method	description
1	col.componentColumn(ComponentBuilder<?, ?> component), col.componentColumn(String title, ComponentBuilder<?, ?> component)	Creates a new component column - title (optional) - the column title - component - the component definition

Examples
Quick usage:

```
ImageBuilder image = cmp.image(...);
ComponentBuilder<?, ?> component = ...;
report()
.columns(
    col.componentColumn("Image", image),
    col.componentColumn("Component", component))
.setDataSource(...)
```

Another example: *ComponentColumnReport*
Tags: column

1.8 Expression

Expressions are used to define various calculations, conditions, text field content, specific report groups, etc. Every expression can access the declared report fields, variables and other expressions and get their values to calculate the expression value. It must be an instance of DRIExpression.

1.8.1 Simple expression

The basic and the simplest implementation of an expression.

A simple expression must be an instance of DRISimpleExpression and implement the inherited abstract method evaluate.

public Object evaluate(ReportParameters reportParameters);
- reportParameters - access to report fields, variables, parameters, expressions, and other report values
- the method returns the result of the expression evaluation

Examples
Quick usage:

```
private class SimpleExpression extends AbstractSimpleExpression<BigDecimal> {
public BigDecimal evaluate(ReportParameters reportParameters) {
    Integer quantity = reportParameters.getValue("quantity");
    BigDecimal unitPrice = reportParameters.getValue("unitprice");
    return new BigDecimal(quantity).multiply(unitPrice);
}
}
```

Another example: *SimpleExpressionReport*

1.8.2 Complex expression

The difference between a simple and complex expression is that a complex expression allows registering additional fields or variables that are not defined in the report and are needed for calculating the value.

A complex expression must be an instance of DRIComplexExpression and must implement the inherited abstract method evaluate.

public Object evaluate(List<?> values, ReportParameters reportParameters);
- values - the values of the registered expressions
- reportParameters - access to report fields, variables, parameters, expressions, and other report values
- the method returns the result of the expression evaluation

Examples
Quick usage:

```
private class ComplexExpression extends AbstractComplexExpression<BigDecimal> {
public ComplexExpression() {
    addExpression(field("quantity", Integer.class));
    addExpression(field("unitprice", BigDecimal.class));
}

@Override
```

(continues on next page)

(continued from previous page)

```
public BigDecimal evaluate(List<?> values, ReportParameters reportParameters) {
    Integer quantity = (Integer) values.get(0);
    BigDecimal unitPrice = (BigDecimal) values.get(1);
    return new BigDecimal(quantity).multiply(unitPrice);
}
}
```

Another example: *ComplexExpressionReport*

1.8.3 Jasper expression

This expression allows declaring an expression in a Jasper native syntax. Knowledge of the jasper syntax is also required for proper use.

Configuration options

#	method	description
1	exp.jasperSyntax(String expression, Class class), exp.jasperSyntax(String expression)	Creates a new jasper expression - expression - the jasper expression - class - the expression class
2	exp.jasperSyntaxText(String text)	Creates a new jasper string expression, useful only for showing a static text. This method escapes the characters in a String using Java String rules. - text - text to be shown

Examples

Quick usage:

```
JasperExpression<BigDecimal> columnExpression = exp.jasperSyntax("new BigDecimal($F
↪{quantity}).multiply($F{unitprice})", BigDecimal.class);
JasperExpression<String> columnTitle = exp.jasperSyntaxText("Price");
TextColumnBuilder<BigDecimal> priceColumn = col.column(columnExpression).
↪setTitle(columnTitle);
```

Another example: *JasperExpressionReport*

1.8.4 Value formatter

The purpose of this expression is to format a value only. For instance, when it is necessary to display a currency next to the value or just show a value in another format. It can be applied in any report column, group, subtotal, or text field component.

A value formatter expression must be an instance of DRIValueFormatter and must implement the inherited abstract method format.

public Object format(Object value, ReportParameters reportParameters);
- value - the value to be formatted
- reportParameters - access to report fields, variables, parameters, expressions, and other report values
- the method returns the formatted value

Examples
Quick usage:

The code below converts a big decimal value to a string and adds a currency symbol next to the value.

```
TextColumnBuilder<BigDecimal>  column = ...;
column. setValueFormatter(new ValueFormatter()))
private class ValueFormatter extends AbstractValueFormatter<String, BigDecimal> {
    public String format(BigDecimal value, ReportParameters reportParameters) {
        return value + " EUR";
    }
}
```

Another example: *ValueFormatterReport*
Tags: expression

1.9 Fonts

Font configuration is very important for a proper report generation and exports. In pdf exports in particular, where it is possible to embed fonts into a document. Keep in mind that if a font is not found, a default system font is used, and this may cause a lot of problems with the layout of the report. The font configuration itself is not very complicated, just follow these few steps:

Say, we want to register a font named *FreeUniversal*. First we need to create a font configuration file fonts.xml and add it in the root of the classpath. Using this configuration, we should register all fonts that are used in the reports. It's recommended to register all font styles (normal, bold, italic and bolditalic). The configuration file will look like this:

```
<fontFamilies>
    <fontFamily name="FreeUniversal">
        <normal>net/sf/dynamicreports/examples/fonts/FreeUniversal-regular.ttf</
↪normal>
        <bold>net/sf/dynamicreports/examples/fonts/FreeUniversal-Bold.ttf</bold>
        <italic>net/sf/dynamicreports/examples/fonts/FreeUniversal-Italic.ttf</
↪italic>
        <boldItalic>net/sf/dynamicreports/examples/fonts/FreeUniversal-BoldItalic.ttf
↪</boldItalic>
```

(continues on next page)

(continued from previous page)

```
        <pdfEncoding>Cp1252</pdfEncoding>
        <pdfEmbedded>true</pdfEmbedded>
    </fontFamily>
</fontFamilies>
```

Now we have to add the font files to the classpath or pack them into a separate jar file and put the jar to the classpath. The font net/sf/dynamicreports/examples/fonts/FreeUniversal-regular.ttf means that the font file FreeUniversal-regular.ttf has to be found in package net.sf.dynamicreports.examples.fonts.

And finally, we need to create a jasperreports_extension.properties file and register the font configuration file:

```
net.sf.jasperreports.extension.registry.factory.simple.font.families=net.sf.
↪jasperreports.engine.fonts.SimpleFontExtensionsRegistryFactory
net.sf.jasperreports.extension.simple.font.families.drfonts=fonts.xml
```

After applying the configuration above, you can use the font in this way:

```
StyleBuilder plainStyle = stl.style()
.setFontName("FreeUniversal");
report().
    title(cmp.text("text - plain").setStyle(plainStyle))
```

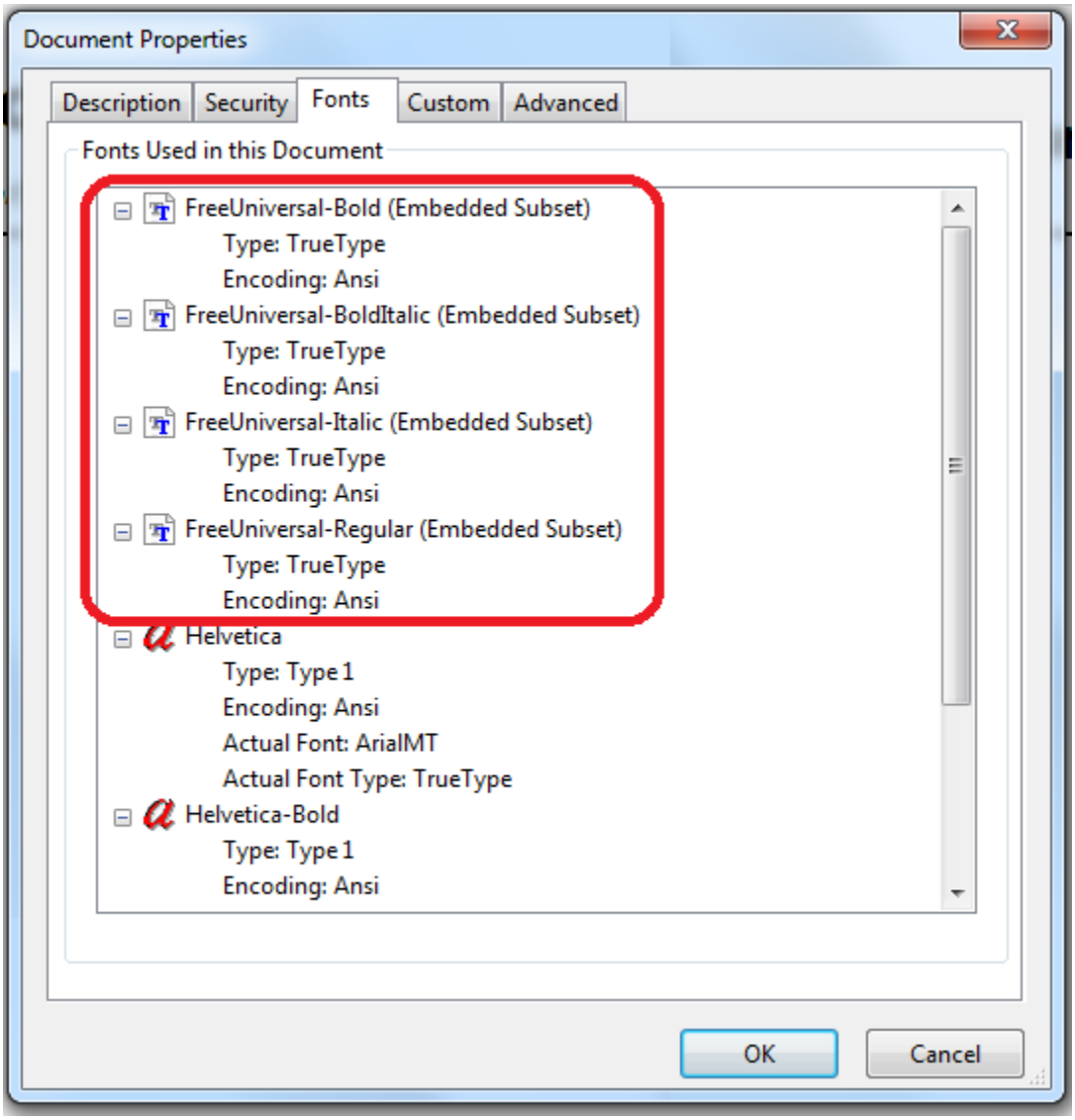
If you want to set the font FreeUniversal as a deafult font, create a dynamicreports-defaults.xml file and set the default font name:

```
<DynamicReports>
<!-- DEFAULT FONT -->
    <font fontName="FreeUniversal" fontSize="10"/>
</DynamicReports>
```

If you don't set the style in a column or any other text component, the default font is used

```
report().
    title(cmp.text("text - default font"))
```

You can check if the configuration is working properly by exporting the report into pdf. Open the pdf document and look at the document properties and select fonts, there should be a list of all embedded FreeUniversal fonts.



For more information about font configuration visit the following site: Font extensions

Examples
FontsReport
Tags: style

1.10 Servlets

Code below shows using the DynamicReports within a servlet. The servlet will display a pdf document on the web browser.

```
1 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
2 import java.io.IOException;
3 import java.io.OutputStream;
4 import javax.servlet.ServletException;
```

(continues on next page)

(continued from previous page)

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.dynamicreports.report.exception.DRException;

public class PdfReportServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
@Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("application/pdf");
        OutputStream out = resp.getOutputStream();
        try {
        report()
            ...
            .toPdf(out);
        } catch (DRException e) {
        throw new ServletException(e);
        }
        out.close();
    }
}
```

1.11 Configuration

1.11.1 Default settings

DynamicReports automatically looks for a file named *dynamicreports-defaults.xml* on startup. In this file you can change the default settings (default font, pattern and horizontal alignment for all data types).

```
<?xml version="1.0" encoding="UTF-8"?>
<DynamicReports>
  <!-- DEFAULT FONT -->
  <font fontName="SansSerif" fontSize="10" pdfFontName="Helvetica" pdfEncoding="Cp1252" pdfEmbedded="false"/>

  <!-- DEFAULT DATA TYPES -->
  <bigDecimalType pattern="#,##0.00#" horizontalAlignment="RIGHT"/>
  <dateType pattern="MM/dd/yyyy" horizontalAlignment="RIGHT"/>
...
</DynamicReports>
```

Copy this file in the root of the classpath.

1.11.2 Maven configuration

DynamicReports is synchronized with Maven Central Repository. Add the dependencies listed below to your pom.xml configuration file.

```
<dependency>
  <groupId>net.sourceforge.dynamicreports</groupId>
  <artifactId>dynamicreports-core</artifactId>
```

(continues on next page)

(continued from previous page)

```
    <version>5.1.0</version>
  </dependency>
  <dependency>
    <groupId>net.sourceforge.dynamicreports</groupId>
    <artifactId>dynamicreports-adhoc</artifactId>
    <version>5.1.0</version>
  </dependency>
  <dependency>
    <groupId>net.sourceforge.dynamicreports</groupId>
    <artifactId>dynamicreports-googlecharts</artifactId>
    <version>5.1.0</version>
  </dependency>
```

In some situations, maven may have a problem with downloading the itext-2.1.7.js6 dependency to your local maven repository. The problem may happen because the library itext-2.1.7.js6 is not available in maven central repository, it is only available through JasperReports maven repository. If you are behind a firewall check if the firewall is not blocking that repository link or download the dependency and add it to your repository manually.

Maven snapshot repository configuration To use the development snapshots of DynamicReports, change your pom.xml configuration: just update the dynamicreports-core version

```
<dependency>
  <groupId>net.sourceforge.dynamicreports</groupId>
  <artifactId>dynamicreports-core</artifactId>
  <version>5.1.1-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>net.sourceforge.dynamicreports</groupId>
  <artifactId>dynamicreports-adhoc</artifactId>
  <version>5.1.1-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>net.sourceforge.dynamicreports</groupId>
  <artifactId>dynamicreports-googlecharts</artifactId>
  <version>5.1.1-SNAPSHOT</version>
</dependency>
```

and add the Sonatype Nexus snapshot repository

```
<repositories>
  <repository>
    <id>sonatype-nexus-snapshots</id>
    <name>Sonatype Nexus Snapshots</name>
    <url>https://oss.sonatype.org/content/repositories/snapshots</url>
    <releases>
      <enabled>false</enabled>
    </releases>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
</repositories>
```

1.11.3 dynamicreports-defaults.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<DynamicReports>
    <!-- DEFAULT FONT -->
    <!--
    <font fontName="SansSerif" fontSize="10"/>
    -->

    <!-- DEFAULT DATA TYPES -->
    <!--
    <bigDecimalType pattern="#,##0.00#" horizontalAlignment="RIGHT"/>
    <bigIntegerType pattern="#,##0" horizontalAlignment="RIGHT"/>
    <byteType pattern="#,##0" horizontalAlignment="RIGHT"/>
    <doubleType pattern="#,##0.#" horizontalAlignment="RIGHT"/>
    <floatType pattern="#,##0.#" horizontalAlignment="RIGHT"/>
    <integerType pattern="#,##0" horizontalAlignment="RIGHT"/>
    <longType pattern="#,##0" horizontalAlignment="RIGHT"/>
    <shortType pattern="#,##0" horizontalAlignment="RIGHT"/>
    <dateType pattern="MM/dd/yyyy" horizontalAlignment="RIGHT"/>
    <dateYearToMonthType pattern="MM/yyyy" horizontalAlignment="RIGHT"/>
    <dateYearToHourType pattern="MM/dd/yyyy h a" horizontalAlignment="RIGHT"/>
    <dateYearToMinuteType pattern="MM/dd/yyyy h:mm a" horizontalAlignment="RIGHT"/>
    <dateYearToSecondType pattern="MM/dd/yyyy h:mm:ss a" horizontalAlignment="RIGHT"/>
    <dateYearToFractionType pattern="MM/dd/yyyy h:mm:ss,SSS a" horizontalAlignment=
↪"RIGHT"/>
    <dateYearType pattern="yyyy" horizontalAlignment="RIGHT"/>
    <dateMonthType pattern="MMMM" horizontalAlignment="RIGHT"/>
    <dateDayType pattern="dd" horizontalAlignment="RIGHT"/>
    <timeHourToMinuteType pattern="h:mm a" horizontalAlignment="RIGHT"/>
    <timeHourToSecondType pattern="h:mm:ss a" horizontalAlignment="RIGHT"/>
    <timeHourToFractionType pattern="h:mm:ss,SSS a" horizontalAlignment="RIGHT"/>
    <percentageType pattern="#,##0.00%" horizontalAlignment="RIGHT"/>
    <booleanType horizontalAlignment="CENTER"/>
    <characterType horizontalAlignment="LEFT"/>
    <stringType horizontalAlignment="LEFT"/>
    -->

    <!-- LOAD SYSTEM FONTS -->
    <!--
    <loadSystemFonts>true</loadSystemFonts>
    -->
</DynamicReports>
```

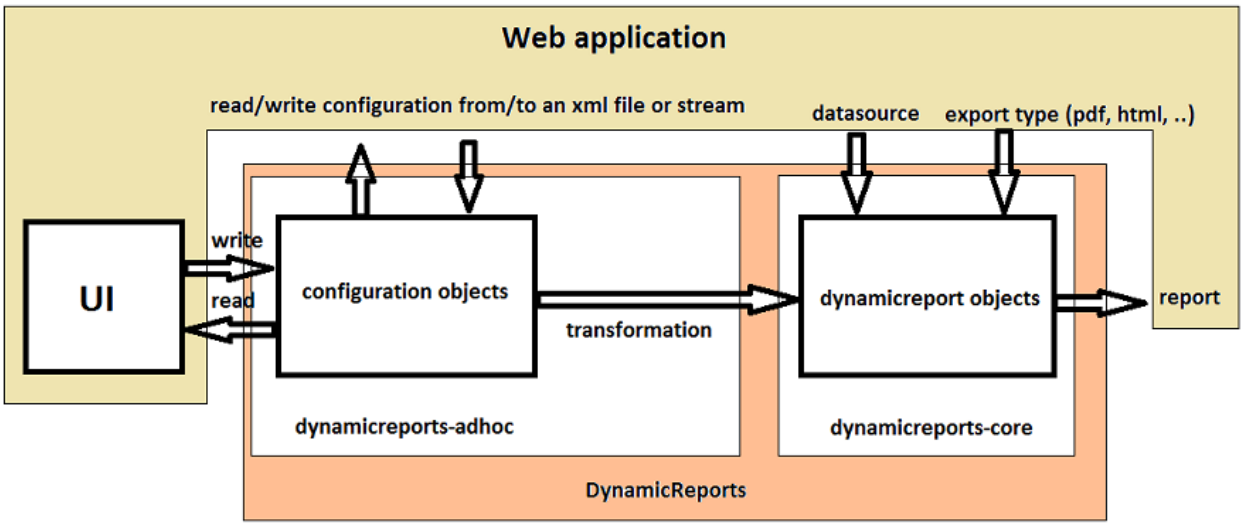
1.12 Adhoc Reports

Adhoc report applications allow users themselves to create specific reports and filter queries. A report is created on the fly, displaying values or charts that a user has specified via GUI. This project simplifies creating adhoc report applications. It allows to configure reports and filter queries and this configuration can be stored to an xml file or loaded from an xml file. The main configuration object is AdhocConfiguration that consists of AdhocReport and AdhocFilter object.

AdhocReport - a report configuration object that can be transformed to a dynamic report builder object.

AdhocFilter - a filter configuration object. This project doesn't generate sql queries from this object, it only allows you to store and load a filter configuration.

Another important class of this project is **AdhocManager**. It is a helper class that contains methods of loading/storing an AdhocConfiguration from/to a stream and methods of creating a report builder instance from a given AdhocReport object.



Examples

Quick usage:

```
AdhocConfiguration configuration = new AdhocConfiguration();
AdhocReport report = new AdhocReport();
configuration.setReport(report);
AdhocColumn column = new AdhocColumn();
column.setName("item");
report.addColumn(column);
```

The following code stores a configuration to an output stream

```
AdhocManager.saveConfiguration(configuration, outputStream);
```

The following code loads a configuration from an input stream

```
AdhocConfiguration configuration = AdhocManager.loadConfiguration(configuration,␣
↪inputStream);
```

The following code converts a configuration to a report builder instance

```
JasperReportBuilder reportBuilder = AdhocManager.createReport(configuration.
↪getReport());
reportBuilder.setDatasource(...);
```

Tags: adhoc

1.13 Features

- Columns - field column, expression column, percentage column, report row number column, page row number column, column row number column, component column, boolean column
- Column grid - for arranging columns layout
- Groups - field group, custom group, column group
- Subtotals - at title, page header/footer, column header/footer, group header/footer, first group header/footer, last group header/footer, last page footer, summary aggregation subtotal, custom subtotal, percentage subtotal
- Components
  - **Containers** - can hold components horizontal list, horizontal flow list, vertical list, multi page list
  - **Text fields** text field, current date field, page x of y field, page x slashed y field, page number field, total pages field
  - Charts area, stacked area, bar, layered bar, bar 3D, stacked bar, grouped stacked bar, stacked bar 3D, waterfall bar, line, pie, pie 3D, time series, difference, scatter, xy area, xy bar, xy line, xy step, spider, multi axis, xy block, bubble, candlestick, highlow, meter, thermometer, gantt
  - Barcodes *barcode4j* - codabar, code128, ean128, dataMatrix, code39, interleaved2Of5, upca, upce, ean13, ean8, uspsIntelligentMail, royalMailCustomer, postnet, pdf417 *barbecue* - 2of7, 3of9, bookland, codabar, code128, code128A, code128B, code128C, code39, code39 (Extended), ean128, ean13, globalTradeItemNumber, int2of5, monarch, nw7, pdf417, postNet, randomWeightUpca, scc14ShippingCode, shipmentIdentificationNumber, sscc18, std2of5, ucc128, upca, usd3, usd4, usps
  - **Other components** crosstab, subreports, map, image, line, filler, page break, column break, generic element, ellipse, rectangle
  - **Google charts** geo map
- Report bands - can hold components title, page header/footer, column header/footer, detail, detail header/footer, last page footer, summary, no data, background
- Row highlighters - conditional highlighter for highlighting detail rows
- Styles style, conditional style
- **Table of contents** - a table of contents generator
- Others concatenated reports, adhoc reporting, jrxml template design, sorting, data types, expressions, report template, template styles, parameters, fields, variables, scriptlets, data filters, ...

1.14 Requirements

1.14.1 Required libraries

dynamicreports
java 1.7 or above
jasperreports 6.5.1
commons-beanutils 1.9.3
commons-logging 1.1.1
commons-collections 3.2.2
commons-digester 2.1
commons-code 1.10

xml-apis 1.3.04
xml-apis-ext 1.3.04
ecj 4.4.2
jfreechart 1.0.19
jcommon 1.0.23
commons-lang 2.6
commons-lang3 3.1
jackson-core 2.1.4
jackson-databind 2.1.4
jackson-annotations 2.1.4
log4j 1.2.14
velocity 1.7

1.14.2 Required libraries for barcodes

barcode4j 2.0
barbecue 1.5-beta1
core 3.2.1
avalon-framework-impl 4.2.0
batik-anim 1.9
batik-awt-util 1.9
batik-bridge 1.9
batik-css 1.9
batik-dom 1.9
batik-ext 1.9
batik-gvt-1.9
batik-parser 1.9
batik-script 1.9
batik-svg-dom 1.9
batik-util 1.9
batik-xml 1.9
batik-i18n 1.9

1.14.3 Required libraries for exporting

PDF itext 2.1.7.js6
Excel poi 3.15

1.14.4 Optional

mlgraphics-commons 2.1
commons-collections4 4.1
bcprov-jdk15on 1.52
spring-core 3.2.18.RELEASE
spring-beans 3.2.18.RELEASE

hsqldb 1.8.0.10
xalan 2.7.2
serializer 2.7.2

1.15 Links & Resources

1.15.1 JasperReports documentation

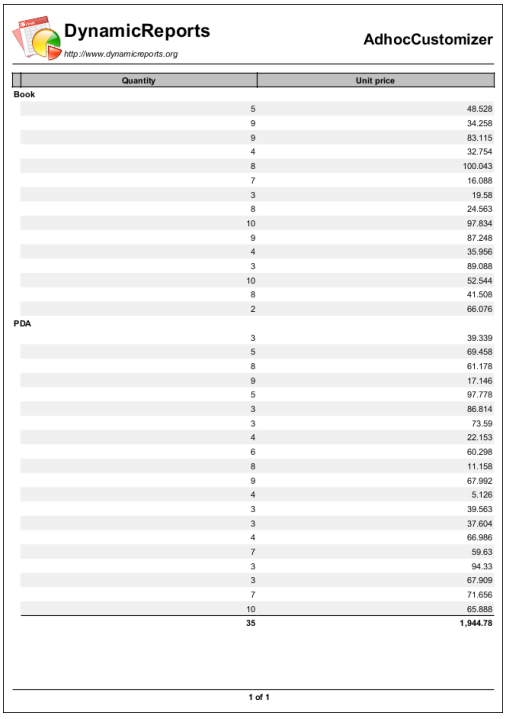
JasperReports ultimate guide, legacy 3.0 version.
JasperStudio IDE Download
JasperStudio User guide
JasperReports Sample reference
JasperReports Configuration Reference

1.15.2 Ivan Lagunov's blog

DynamicReports and Spring MVC integration
DynamicReports and Cocoon integration

1.16 Adhoc

1.16.1 Adhoc Customizer



DynamicReports
http://www.dynamicreports.org

AdhocCustomizer

	Quantity	Unit price
Book		
	5	48.528
	9	34.258
	9	83.115
	4	32.754
	8	100.043
	7	16.088
	3	19.58
	8	24.563
	10	97.834
	9	87.248
	4	35.956
	3	89.088
	10	52.544
	8	41.508
	2	66.076
PDA		
	3	39.339
	5	69.458
	8	61.178
	9	17.146
	5	97.778
	3	86.814
	3	73.59
	4	22.153
	6	60.298
	8	11.158
	9	67.992
	4	5.126
	3	39.563
	3	37.604
	4	66.986
	7	59.63
	3	94.33
	3	67.909
	7	71.656
	10	65.888
	35	1,944.78

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.adhoc;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.adhoc.AdhocManager;
import net.sf.dynamicreports.adhoc.configuration.AdhocCalculation;
import net.sf.dynamicreports.adhoc.configuration.AdhocColumn;
import net.sf.dynamicreports.adhoc.configuration.AdhocConfiguration;
import net.sf.dynamicreports.adhoc.configuration.AdhocGroup;
import net.sf.dynamicreports.adhoc.configuration.AdhocReport;
import net.sf.dynamicreports.adhoc.configuration.AdhocSort;
import net.sf.dynamicreports.adhoc.configuration.AdhocSubtotal;
import net.sf.dynamicreports.adhoc.report.DefaultAdhocReportCustomizer;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.builder.ReportBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.datatype.DRIDataType;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class AdhocCustomizerReport {

        public AdhocCustomizerReport() {
                build();
        }

        private void build() {
                AdhocConfiguration configuration = new AdhocConfiguration();
```

(continues on next page)

(continued from previous page)

```
58                  AdhocReport report = new AdhocReport();
59                  configuration.setReport(report);
60
61                  // columns
62                  AdhocColumn column = new AdhocColumn();
63                  column.setName("quantity");
64                  report.addColumn(column);
65                  column = new AdhocColumn();
66                  column.setName("unitprice");
67                  report.addColumn(column);
68                  // groups
69                  AdhocGroup group = new AdhocGroup();
70                  group.setName("item");
71                  report.addGroup(group);
72                  // subtotal
73                  AdhocSubtotal subtotal = new AdhocSubtotal();
74                  subtotal.setName("quantity");
75                  subtotal.setCalculation(AdhocCalculation.COUNT);
76                  report.addSubtotal(subtotal);
77                  subtotal = new AdhocSubtotal();
78                  subtotal.setCalculation(AdhocCalculation.SUM);
79                  subtotal.setName("unitprice");
80                  report.addSubtotal(subtotal);
81                  // sorts
82                  AdhocSort sort = new AdhocSort();
83                  sort.setName("item");
84                  report.addSort(sort);
85
86                  try {
87                          JasperReportBuilder reportBuilder = AdhocManager.
    ↪createReport(configuration.getReport(), new ReportCustomizer());
88                          reportBuilder.setDataSource(createDataSource());
89                          reportBuilder.show();
90                  } catch (DRException e) {
91                          e.printStackTrace();
92                  }
93          }
94
95          private JRDataSource createDataSource() {
96                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪"quantity", "unitprice");
97                  for (int i = 0; i < 15; i++) {
98                          dataSource.add("Book", new Date(), (int) (Math.random() * 10)␣
    ↪+ 1, new BigDecimal(Math.random() * 100 + 1));
99                  }
100                 for (int i = 0; i < 20; i++) {
101                         dataSource.add("PDA", new Date(), (int) (Math.random() * 10)␣
    ↪+ 1, new BigDecimal(Math.random() * 100 + 1));
102                 }
103                 return dataSource;
104         }
105
106         public static void main(String[] args) {
107                 new AdhocCustomizerReport();
108         }
109
110         private class ReportCustomizer extends DefaultAdhocReportCustomizer {
```

(continues on next page)

(continued from previous page)

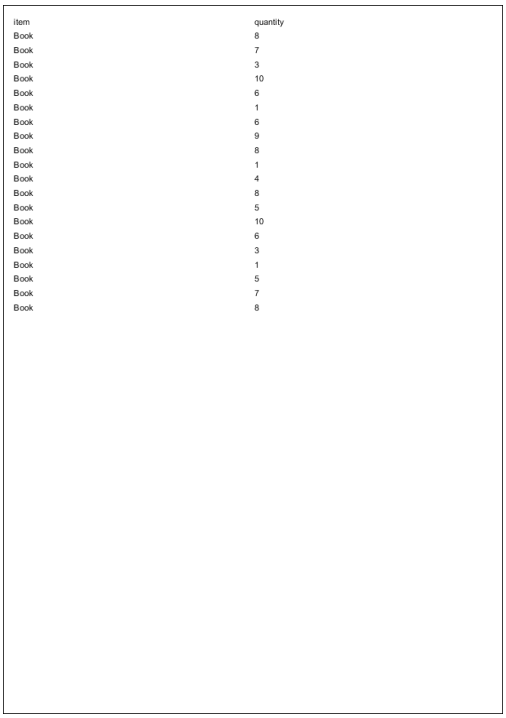
```
111
112                 /**
113                  * If you want to add some fixed content to a report that is not␣
    ↪needed to store in the xml file.
114                  * For example you can add default page header, footer, default fonts,
    ↪...
115                  */
116                 @Override
117                 public void customize(ReportBuilder<?> report, AdhocReport␣
    ↪adhocReport) throws DRException {
118                         super.customize(report, adhocReport);
119                         // default report values
120                         report.setTemplate(Templates.reportTemplate);
121                         report.title(Templates.createTitleComponent("AdhocCustomizer
    ↪"));
122                         // a fixed page footer that user cannot change, this␣
    ↪customization is not stored in the xml file
123                         report.pageFooter(Templates.footerComponent);
124                 }
125
126                 /**
127                  * This method returns a field type from a given field name.
128                  */
129                 @Override
130                 protected DRIDataType<?, ?> getFieldType(String name) {
131                         if (name.equals("item")) {
132                                 return type.stringType();
133                         }
134                         if (name.equals("orderdate")) {
135                                 return type.dateType();
136                         }
137                         if (name.equals("quantity")) {
138                                 return type.integerType();
139                         }
140                         if (name.equals("unitprice")) {
141                                 return type.bigDecimalType();
142                         }
143                         return super.getFieldType(name);
144                 }
145
146                 /**
147                  * If a user doesn't specify a column title, getColumnTitle is␣
    ↪evaluated and the return value is used as a column title.
148                  */
149                 @Override
150                 protected String getFieldLabel(String name) {
151                         if (name.equals("item")) {
152                                 return "Item";
153                         }
154                         if (name.equals("orderdate")) {
155                                 return "Order date";
156                         }
157                         if (name.equals("quantity")) {
158                                 return "Quantity";
159                         }
160                         if (name.equals("unitprice")) {
161                                 return "Unit price";
```

(continues on next page)

(continued from previous page)

```
                                }
                                return name;
                        }

                }
}
```

1.16.2 Simple Adhoc



item	quantity
Book	8
Book	7
Book	3
Book	10
Book	6
Book	1
Book	6
Book	9
Book	8
Book	1
Book	4
Book	8
Book	5
Book	10
Book	6
Book	3
Book	1
Book	5
Book	7
Book	8

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
```

(continues on next page)

(continued from previous page)

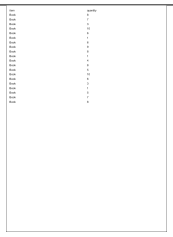
```
 */

package net.sf.dynamicreports.examples.adhoc;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.adhoc.AdhocManager;
import net.sf.dynamicreports.adhoc.configuration.AdhocColumn;
import net.sf.dynamicreports.adhoc.configuration.AdhocConfiguration;
import net.sf.dynamicreports.adhoc.configuration.AdhocReport;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SimpleAdhocReport {

        public SimpleAdhocReport() {
                build();
        }

        private void build() {
                AdhocConfiguration configuration = new AdhocConfiguration();
                AdhocReport report = new AdhocReport();
                configuration.setReport(report);

                AdhocColumn column = new AdhocColumn();
                column.setName("item");
                report.addColumn(column);

                column = new AdhocColumn();
                column.setName("quantity");
                report.addColumn(column);

                try {
                        // The following code stores the configuration to an xml file
                        AdhocManager.saveConfiguration(configuration, new␣
↪FileOutputStream("c:/temp/configuration.xml"));
                        @SuppressWarnings("unused")
                        // The following code loads a configuration from an xml file
                        AdhocConfiguration loadedConfiguration = AdhocManager.
↪loadConfiguration(new FileInputStream("c:/temp/configuration.xml"));

                        JasperReportBuilder reportBuilder = AdhocManager.
↪createReport(configuration.getReport());
                        reportBuilder.setDataSource(createDataSource());
                        reportBuilder.show();
                } catch (DRException e) {
                        e.printStackTrace();
                } catch (FileNotFoundException e) {
```

(continues on next page)

(continued from previous page)

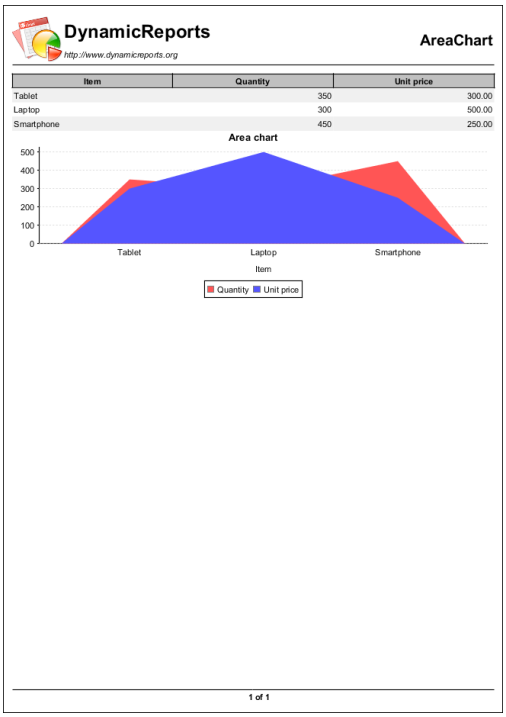
```
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                for (int i = 0; i < 20; i++) {
                        dataSource.add("Book", new Date(), (int) (Math.random() * 10)␣
↪+ 1, new BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new SimpleAdhocReport();
        }
}
```

Table 1: Adhoc Examples

	
AdhocCustomizer	SimpleAdhoc

1.17 Chart

1.17.1 Area Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
```

(continues on next page)

(continued from previous page)

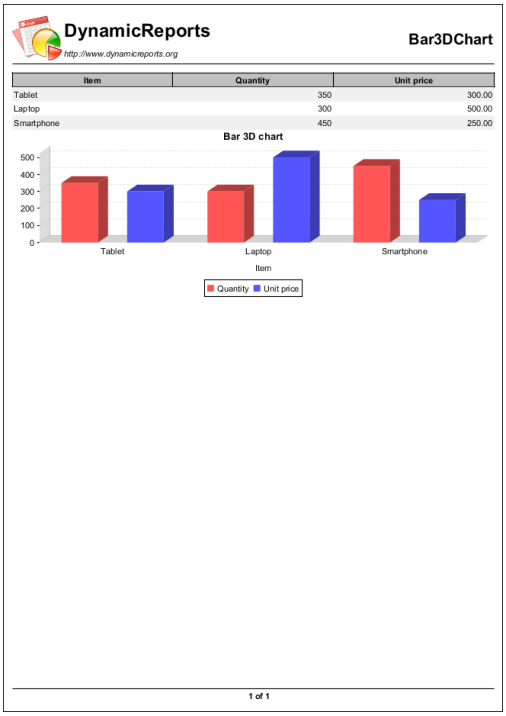
```
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.builder.style.FontBuilder;
32  import net.sf.dynamicreports.report.datasource.DRDataSource;
33  import net.sf.dynamicreports.report.exception.DRException;
34  import net.sf.jasperreports.engine.JRDataSource;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class AreaChartReport {
40
41          public AreaChartReport() {
42                  build();
43          }
44
45          private void build() {
46                  FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48                  TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
    ↪type.stringType());
49                  TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
    ↪"quantity", type.integerType());
50                  TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
    ↪", "unitprice", type.bigDecimalType());
51
52                  try {
53                          report()
54                                          .setTemplate(Templates.reportTemplate)
55                                          .columns(itemColumn, quantityColumn,␣
    ↪unitPriceColumn)
56                                          .title(Templates.createTitleComponent(
    ↪"AreaChart"))
57                                          .summary(
58                                                          cht.areaChart()
59                                                                          .setTitle(
    ↪"Area chart")
60                                                                          .
    ↪setTitleFont(boldFont)
61                                                                          .
    ↪setCategory(itemColumn)
62                                                                          .series(
63                                                                                          cht.
    ↪serie(quantityColumn), cht.serie(unitPriceColumn))
64                                                                          .
    ↪setCategoryAxisFormat(
65                                                                                          cht.
    ↪axisFormat().setLabel("Item")))
66                                          .pageFooter(Templates.footerComponent)
67                                          .setDataSource(createDataSource())
68                                          .show();
69                  } catch (DRException e) {
70                          e.printStackTrace();
71                  }
72          }
73
```

(continues on next page)

(continued from previous page)

```
74          private JRDataSource createDataSource() {
75                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
76                  dataSource.add("Tablet", 350, new BigDecimal(300));
77                  dataSource.add("Laptop", 300, new BigDecimal(500));
78                  dataSource.add("Smartphone", 450, new BigDecimal(250));
79                  return dataSource;
80          }
81
82          public static void main(String[] args) {
83                  new AreaChartReport();
84          }
85  }
```

1.17.2 Bar 3D Chart




```
1   /**
2    * DynamicReports - Free Java reporting library for creating reports dynamically
3    *
4    * Copyright (C) 2010 - 2018 Ricardo Mariaca
5    * http://www.dynamicreports.org
6    *
7    * This file is part of DynamicReports.
8    *
9    * DynamicReports is free software: you can redistribute it and/or modify
10   * it under the terms of the GNU Lesser General Public License as published by
11   * the Free Software Foundation, either version 3 of the License, or
12   * (at your option) any later version.
13   *
```

(continues on next page)

(continued from previous page)

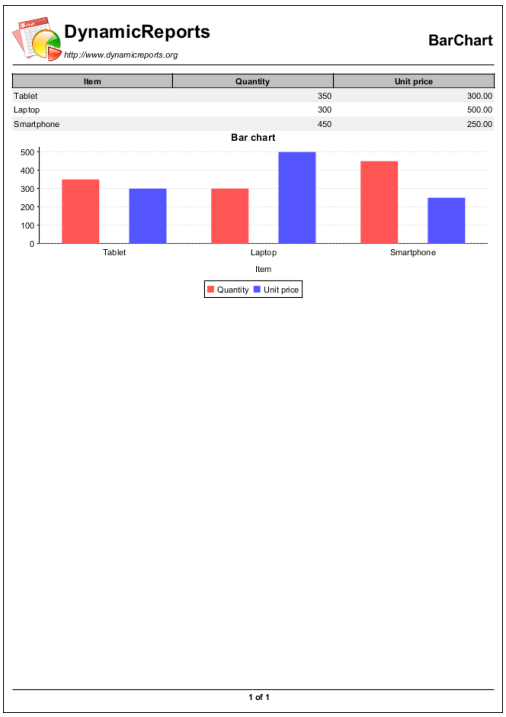
```
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class Bar3DChartReport {

        public Bar3DChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("Bar3DChart"))
                                        .summary(
                                                        cht.bar3DChart()
                                                                        .setTitle("Bar 3D chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setCategory(itemColumn)
                                                                        .series(
```

(continues on next page)

(continued from previous page)

```
                                                                                cht.
↪serie(quantityColumn), cht.serie(unitPriceColumn))
                                                                                  .
↪setCategoryAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new Bar3DChartReport();
        }
}
```

1.17.3 Bar Chart



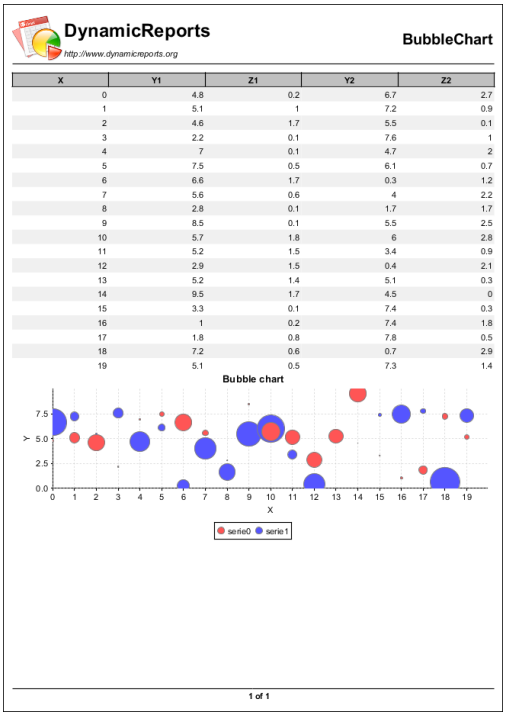
```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class BarChartReport {

        public BarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
```

(continues on next page)

(continued from previous page)

```
                                        .columns(itemColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("BarChart"))
                                        .summary(
                                                cht.barChart()
                                                        .setTitle("Bar chart")
                                                        .setTitleFont(boldFont)
                                                        .setCategory(itemColumn)
                                                        .series(cht.serie(quantityColumn), cht.serie(unitPriceColumn))
                                                        .setCategoryAxisFormat(cht.axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new BarChartReport();
        }
}
```

1.17.4 Bubble Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

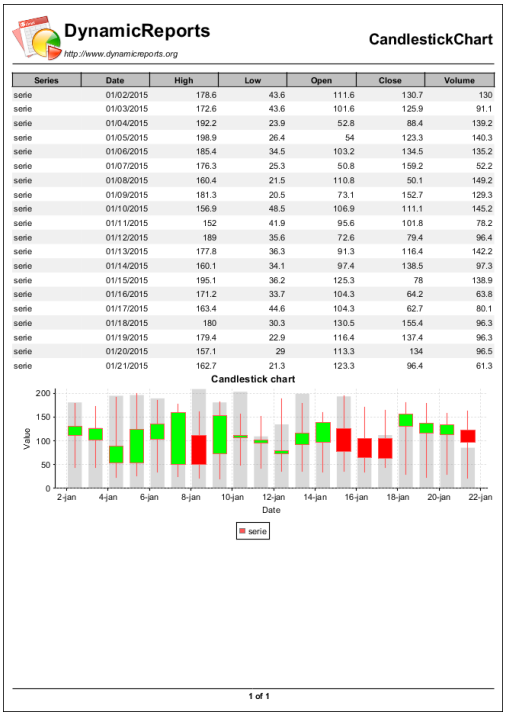
```
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class BubbleChartReport {

        public BubbleChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Double> xColumn = col.column("X", "x", type.doubleType());
                TextColumnBuilder<Double> y1Column = col.column("Y1", "y1", type.doubleType());
                TextColumnBuilder<Double> z1Column = col.column("Z1", "z1", type.doubleType());
                TextColumnBuilder<Double> y2Column = col.column("Y2", "y2", type.doubleType());
                TextColumnBuilder<Double> z2Column = col.column("Z2", "z2", type.doubleType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(xColumn, y1Column, z1Column, y2Column, z2Column)
                                        .title(Templates.createTitleComponent("BubbleChart"))
                                        .summary(
                                                        cht.bubbleChart()
                                                                        .setTitle("Bubble chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setXValue(xColumn)
                                                                        .series(
                                                                                        cht.xyzSerie().setYValue(y1Column).setZValue(z1Column),
                                                                                        cht.xyzSerie().setYValue(y2Column).setZValue(z2Column))
                                                                        .setXAxisFormat(
                                                                                        cht.axisFormat().setLabel("X"))
                                                                        .setYAxisFormat(
                                                                                        cht.axisFormat().setLabel("Y")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
```

(continues on next page)

(continued from previous page)

```
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y1", "z1", "y2", "z2
↪");
                for (int i = 0; i < 20; i++) {
                        dataSource.add((double) i, Math.random() * 10, Math.random()␣
↪* 2, Math.random() * 8, Math.random() * 3);
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new BubbleChartReport();
        }
}
```

1.17.5 Candlestick Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CandlestickChartReport {

        public CandlestickChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> seriesColumn = col.column("Series", "series
↪", type.stringType());
                TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
↪dateType());
                TextColumnBuilder<Double> highColumn = col.column("High", "high",␣
↪type.doubleType());
                TextColumnBuilder<Double> lowColumn = col.column("Low", "low", type.
↪doubleType());
                TextColumnBuilder<Double> openColumn = col.column("Open", "open",␣
↪type.doubleType());
                TextColumnBuilder<Double> closeColumn = col.column("Close", "close",␣
↪type.doubleType());
                TextColumnBuilder<Double> volumeColumn = col.column("Volume", "volume
↪", type.doubleType());

                try {
                        report()
```

(continues on next page)

(continued from previous page)

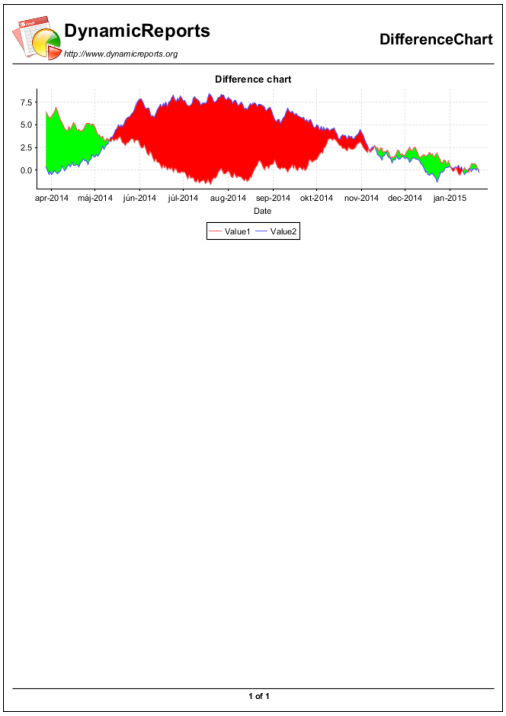
```
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(seriesColumn, dateColumn, highColumn,
↪ lowColumn, openColumn, closeColumn, volumeColumn)
                                        .title(Templates.createTitleComponent(
↪"CandlestickChart"))
                                        .summary(
                                                cht.candlestickChart()
                                                                .setTitle(
↪"Candlestick chart")
                                                                .
↪setTitleFont(boldFont)
                                                                .
↪setSeries(seriesColumn)
                                                                .
↪setDate(dateColumn)
                                                                .
↪setHigh(highColumn)
                                                                .
↪setLow(lowColumn)
                                                                .
↪setOpen(openColumn)
                                                                .
↪setClose(closeColumn)
                                                                .
↪setVolume(volumeColumn)
                                                                .
↪setTimeAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Date"))
                                                                .
↪setValueAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Value")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREception e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("series", "date", "high",
↪"low", "open", "close", "volume");
                Calendar c = Calendar.getInstance();
                c.add(Calendar.DAY_OF_MONTH, -20);
                for (int i = 0; i < 20; i++) {
                        dataSource.add("serie", c.getTime(), 150 + Math.random() * 50,
↪ 20 + Math.random() * 30, 50 + Math.random() * 90, 50 + Math.random() * 110,
                                        50 + Math.random() * 100);
                        c.add(Calendar.DAY_OF_MONTH, 1);
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new CandlestickChartReport();
```

(continues on next page)

(continued from previous page)

```
99              }
100 }
```

1.17.6 Difference Chart




```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
```

(continues on next page)

(continued from previous page)

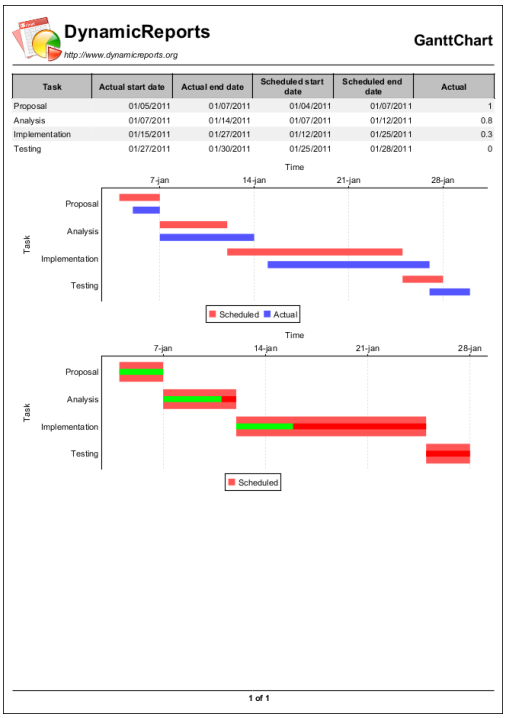
```
import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.DynamicReports;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.constant.TimePeriod;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DifferenceChartReport {

        public DifferenceChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent("DifferenceChart"))
                                        .summary(
                                                        cht.differenceChart()
                                                                        .setTitle("Difference chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setTimePeriod(DynamicReports.<Date>field("date", type.dateType()))
                                                                        .setTimePeriodType(TimePeriod.DAY)
                                                                        .series(
                                                                                        cht.serie(DynamicReports.<Number>field("value1", type.doubleType())).setLabel("Value1"),
                                                                                        cht.serie(DynamicReports.<Number>field("value2", type.doubleType())).setLabel("Value2"))
                                                                        .setTimeAxisFormat(
                                                                                        cht.axisFormat().setLabel("Date")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

```

(continues on next page)

(continued from previous page)

```
73          private JRDataSource createDataSource() {
74                  DRDataSource dataSource = new DRDataSource("date", "value1", "value2
    ↪");
75                  double value1 = 0;
76                  double value2 = 0;
77                  Calendar c = Calendar.getInstance();
78                  for (int i = 0; i < 300; i++) {
79                          c.add(Calendar.DAY_OF_MONTH, -1);
80                          value1 += Math.random() - 0.5;
81                          value2 += Math.random() - 0.5;
82                          dataSource.add(c.getTime(), value1, value2);
83                  }
84                  return dataSource;
85          }
86
87          public static void main(String[] args) {
88                  new DifferenceChartReport();
89          }
90  }
```

1.17.7 Gantt Chart




```
1   /**
2    * DynamicReports - Free Java reporting library for creating reports dynamically
3    *
4    * Copyright (C) 2010 - 2018 Ricardo Mariaca
5    * http://www.dynamicreports.org
6    *
7    * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.chart.GanttChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GanttChartReport {

        public GanttChartReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> taskColumn = col.column("Task", "task",
↪type.stringType());
                TextColumnBuilder<Date> actualStartDateColumn = col.column("Actual
↪start date", "actualstartdate", type.dateType());
                TextColumnBuilder<Date> actualEndDateColumn = col.column("Actual end
↪date", "actualenddate", type.dateType());
                TextColumnBuilder<Date> scheduledStartDateColumn = col.column(
↪"Scheduled start date", "scheduledstartdate", type.dateType());
                TextColumnBuilder<Date> scheduledEndDateColumn = col.column(
↪"Scheduled end date", "scheduledenddate", type.dateType());
                TextColumnBuilder<Double> actualColumn = col.column("Actual", "actual
↪", type.doubleType());

                GanttChartBuilder chart1 = cht.ganttChart()
                                .setTask(taskColumn)
                                .series(
                                                cht.ganttSerie()
                                                                .
↪setStartDate(scheduledStartDateColumn)
```

(continues on next page)

(continued from previous page)

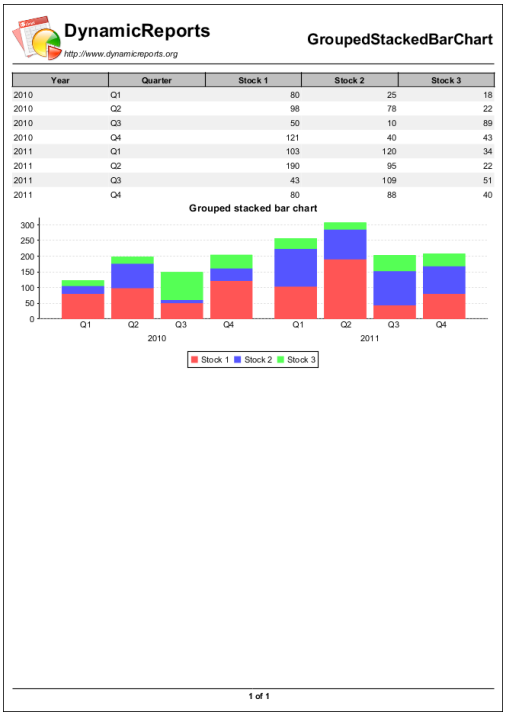
```
                                                                .
↪setEndDate(scheduledEndDateColumn)
                                                                .setLabel("Scheduled
↪"),
                                                cht.ganttSerie()
                                                                .
↪setStartDate(actualStartDateColumn)
                                                                .
↪setEndDate(actualEndDateColumn)
                                                                .setLabel("Actual"))
                                .setTimeAxisFormat(
                                                cht.axisFormat().setLabel("Time"))
                                .setTaskAxisFormat(
                                                cht.axisFormat().setLabel("Task"));

                GanttChartBuilder chart2 = cht.ganttChart()
                                .setTask(taskColumn)
                                .series(
                                                cht.ganttSerie()
                                                                .
↪setStartDate(scheduledStartDateColumn)
                                                                .
↪setEndDate(scheduledEndDateColumn)
                                                                .
↪setPercent(actualColumn)
                                                                .setLabel("Scheduled
↪"))
                                .setTimeAxisFormat(
                                                cht.axisFormat().setLabel("Time"))
                                .setTaskAxisFormat(
                                                cht.axisFormat().setLabel("Task"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(taskColumn, actualStartDateColumn,␣
↪actualEndDateColumn, scheduledStartDateColumn, scheduledEndDateColumn, actualColumn)
                                        .title(Templates.createTitleComponent(
↪"GanttChart"))
                                        .summary(chart1, chart2)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("task", "actualstartdate",
↪"actualenddate", "scheduledstartdate", "scheduledenddate", "actual");
                dataSource.add("Proposal", toDate(2011, 1, 5), toDate(2011, 1, 7),␣
↪toDate(2011, 1, 4), toDate(2011, 1, 7), 1d);
                dataSource.add("Analysis", toDate(2011, 1, 7), toDate(2011, 1, 14),␣
↪toDate(2011, 1, 7), toDate(2011, 1, 12), 0.8d);
                dataSource.add("Implementation", toDate(2011, 1, 15), toDate(2011, 1,␣
↪27), toDate(2011, 1, 12), toDate(2011, 1, 25), 0.3d);
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Testing", toDate(2011, 1, 27), toDate(2011, 1, 30),␣
↪toDate(2011, 1, 25), toDate(2011, 1, 28), 0d);
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.clear();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new GanttChartReport();
        }
}
```

1.17.8 Grouped Stacked Bar Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

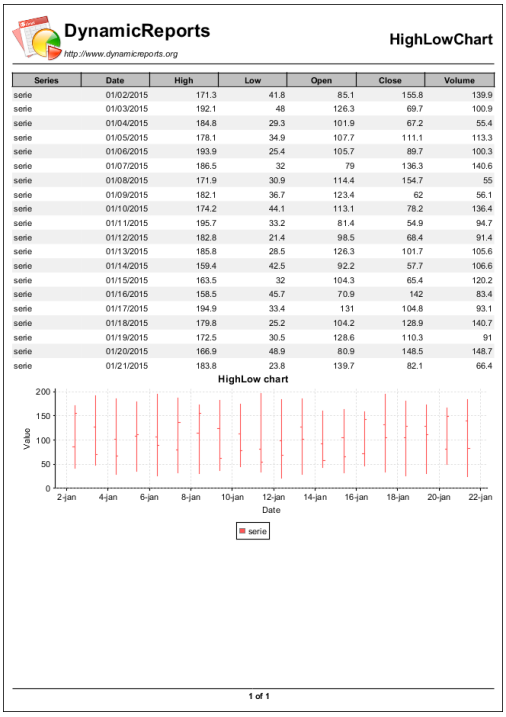
```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupedStackedBarChartReport {

        public GroupedStackedBarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> yearColumn = col.column("Year", "year", type.stringType());
                TextColumnBuilder<String> quarterColumn = col.column("Quarter", "quarter", type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1", "stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2", "stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3", "stock3", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(yearColumn, quarterColumn, stock1Column, stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent("GroupedStackedBarChart"))
                                        .summary(
                                                        cht.groupedStackedBarChart()
                                                                        .setTitle("Grouped stacked bar chart")
```

(continues on next page)

(continued from previous page)

```
                                                                                .
↪setTitleFont(boldFont)
                                                                                .
↪setCategory(yearColumn)
                                                                                .series(
                                                                                cht.
↪groupedSerie(stock1Column).setGroup(quarterColumn),
                                                                                cht.
↪groupedSerie(stock2Column).setGroup(quarterColumn),
                                                                                cht.
↪groupedSerie(stock3Column).setGroup(quarterColumn)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("year", "quarter", "stock1
↪", "stock2", "stock3");
                dataSource.add("2010", "Q1", 80, 25, 18);
                dataSource.add("2010", "Q2", 98, 78, 22);
                dataSource.add("2010", "Q3", 50, 10, 89);
                dataSource.add("2010", "Q4", 121, 40, 43);
                dataSource.add("2011", "Q1", 103, 120, 34);
                dataSource.add("2011", "Q2", 190, 95, 22);
                dataSource.add("2011", "Q3", 43, 109, 51);
                dataSource.add("2011", "Q4", 80, 88, 40);
                return dataSource;
        }

        public static void main(String[] args) {
                new GroupedStackedBarChartReport();
        }
}
```

1.17.9 High Low Chart



Series	Date	High	Low	Open	Close	Volume
serie	01/02/2015	171.3	41.8	85.1	155.8	139.9
serie	01/03/2015	192.1	48	126.3	69.7	100.9
serie	01/04/2015	184.8	29.3	101.9	67.2	55.4
serie	01/05/2015	178.1	34.9	107.7	111.1	113.3
serie	01/06/2015	193.9	25.4	105.7	89.7	100.3
serie	01/07/2015	186.5	32	79	136.3	140.6
serie	01/08/2015	171.9	30.9	114.4	154.7	55
serie	01/09/2015	182.1	36.7	123.4	62	56.1
serie	01/10/2015	174.2	44.1	113.1	78.2	136.4
serie	01/11/2015	195.7	33.2	81.4	54.9	94.7
serie	01/12/2015	182.8	21.4	98.5	68.4	91.4
serie	01/13/2015	185.8	28.5	126.3	101.7	105.6
serie	01/14/2015	159.4	42.5	92.2	57.7	106.6
serie	01/15/2015	163.5	32	104.3	65.4	120.2
serie	01/16/2015	158.5	45.7	70.9	142	83.4
serie	01/17/2015	194.9	33.4	131	104.8	93.1
serie	01/18/2015	179.8	25.2	104.2	128.9	140.7
serie	01/19/2015	172.5	30.5	128.6	110.3	91
serie	01/20/2015	166.9	48.9	80.9	148.5	148.7
serie	01/21/2015	183.8	23.8	139.7	82.1	66.4

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

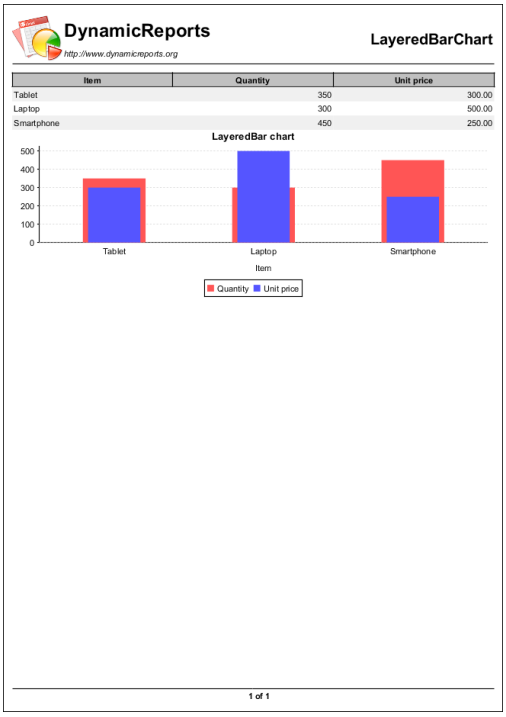
```
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class HighLowChartReport {

        public HighLowChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> seriesColumn = col.column("Series", "series
↪", type.stringType());
                TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
↪dateType());
                TextColumnBuilder<Double> highColumn = col.column("High", "high",␣
↪type.doubleType());
                TextColumnBuilder<Double> lowColumn = col.column("Low", "low", type.
↪doubleType());
                TextColumnBuilder<Double> openColumn = col.column("Open", "open",␣
↪type.doubleType());
                TextColumnBuilder<Double> closeColumn = col.column("Close", "close",␣
↪type.doubleType());
                TextColumnBuilder<Double> volumeColumn = col.column("Volume", "volume
↪", type.doubleType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(seriesColumn, dateColumn, highColumn,
↪ lowColumn, openColumn, closeColumn, volumeColumn)
                                        .title(Templates.createTitleComponent(
↪"HighLowChart"))
                                        .summary(
                                                        cht.highLowChart()
                                                                        .setTitle(
↪"HighLow chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setSeries(seriesColumn)
                                                                        .
↪setDate(dateColumn)
                                                                        .
↪setHigh(highColumn)
                                                                        .
↪setLow(lowColumn)
                                                                        .
↪setOpen(openColumn)
                                                                        .
↪setClose(closeColumn)
```

(continues on next page)

(continued from previous page)

```
                                                                                .
↪setVolume(volumeColumn)
                                                                                .
↪setShowOpenTicks(true)
                                                                                .
↪setShowCloseTicks(true)
                                                                                .
↪setTimeAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Date"))
                                                                                .
↪setValueAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Value")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("series", "date", "high",
↪"low", "open", "close", "volume");
                Calendar c = Calendar.getInstance();
                c.add(Calendar.DAY_OF_MONTH, -20);
                for (int i = 0; i < 20; i++) {
                        dataSource.add("serie", c.getTime(), 150 + Math.random() * 50,
↪ 20 + Math.random() * 30, 50 + Math.random() * 90, 50 + Math.random() * 110,
                                        50 + Math.random() * 100);
                        c.add(Calendar.DAY_OF_MONTH, 1);
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new HighLowChartReport();
        }
}
```

1.17.10 Layered Bar Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

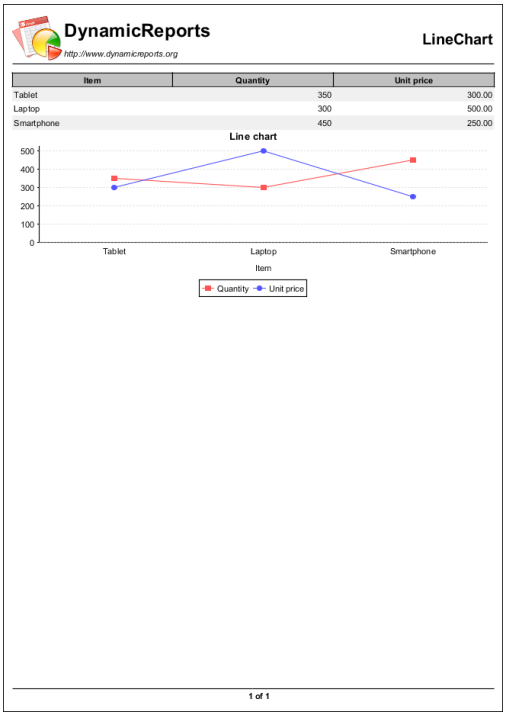
```
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class LayeredBarChartReport {

        public LayeredBarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("LayeredBarChart"))
                                        .summary(
                                                        cht.layeredBarChart()
                                                                        .setTitle("LayeredBar chart")
                                                                        .setTitleFont(boldFont)
                                                                        .seriesBarWidths(1.0, 0.5)
                                                                        .setCategory(itemColumn)
                                                                        .series(cht.serie(quantityColumn), cht.serie(unitPriceColumn))
                                                                        .setCategoryAxisFormat(cht.axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
```

(continues on next page)

(continued from previous page)

```
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new LayeredBarChartReport();
        }
}
```

1.17.11 Line Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
```

(continues on next page)

(continued from previous page)

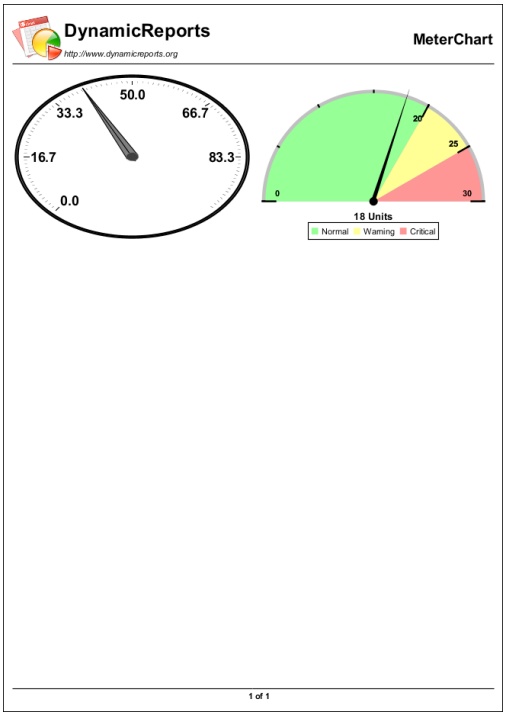
```
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class LineChartReport {

        public LineChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("LineChart"))
                                        .summary(
                                                        cht.lineChart()
                                                                        .setTitle("Line chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setCategory(itemColumn)
                                                                        .series(
                                                                                        cht.serie(quantityColumn), cht.serie(unitPriceColumn))
```

(continues on next page)

(continued from previous page)

```
                                                                                .
↪setCategoryAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Item")))
                                                .pageFooter(Templates.footerComponent)
                                                .setDataSource(createDataSource())
                                                .show();
                        } catch (DREException e) {
                                e.printStackTrace();
                        }
                }

                private JRDataSource createDataSource() {
                        DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                        dataSource.add("Tablet", 350, new BigDecimal(300));
                        dataSource.add("Laptop", 300, new BigDecimal(500));
                        dataSource.add("Smartphone", 450, new BigDecimal(250));
                        return dataSource;
                }

                public static void main(String[] args) {
                        new LineChartReport();
                }
}
```

1.17.12 Meter Chart



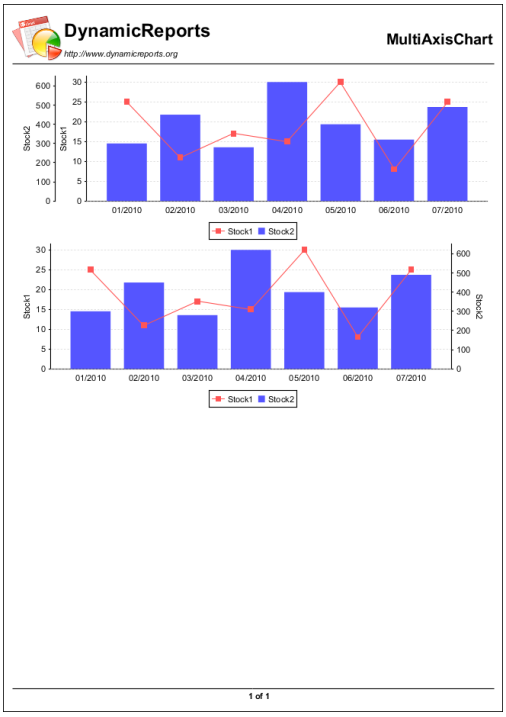
```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.DynamicReports;
import net.sf.dynamicreports.report.builder.chart.MeterChartBuilder;
import net.sf.dynamicreports.report.constant.MeterShape;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class MeterChartReport {

        public MeterChartReport() {
                build();
        }

        private void build() {
                MeterChartBuilder chart1 = cht.meterChart()
                                .setValue(DynamicReports.<Number>field("value", type.
↪integerType()))
                                .setShape(MeterShape.DIAL);

                MeterChartBuilder chart2 = cht.meterChart()
                                .setValue(18)
                                .setDataRangeHighExpression(30)
                                .setTickInterval(5d)
                                .setTickColor(Color.BLACK)
                                .setNeedleColor(Color.BLACK)
```

(continues on next page)

(continued from previous page)

```
                                .setValueColor(Color.BLACK)
                                .setMeterBackgroundColor(Color.LIGHT_GRAY)
                                .intervals(
                                                cht.meterInterval()
                                                                .setLabel("Normal")
                                                                .
↪setBackgroundColor(new Color(150, 255, 150))
                                                                .
↪setDataRangeLowExpression(0)
                                                                .
↪setDataRangeHighExpression(20),
                                                cht.meterInterval()
                                                                .setLabel("Warning")
                                                                .
↪setBackgroundColor(new Color(255, 255, 150))
                                                                .
↪setDataRangeLowExpression(20)
                                                                .
↪setDataRangeHighExpression(25),
                                                cht.meterInterval()
                                                                .setLabel("Critical")
                                                                .
↪setBackgroundColor(new Color(255, 150, 150))
                                                                .
↪setDataRangeLowExpression(25)
                                                                .
↪setDataRangeHighExpression(30));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"MeterChart"))
                                        .summary(
                                                        cmp.horizontalList(chart1,␣
↪chart2))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("value");
                dataSource.add(40);
                return dataSource;
        }

        public static void main(String[] args) {
                new MeterChartReport();
        }
}
```

1.17.13 Multi Axis Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

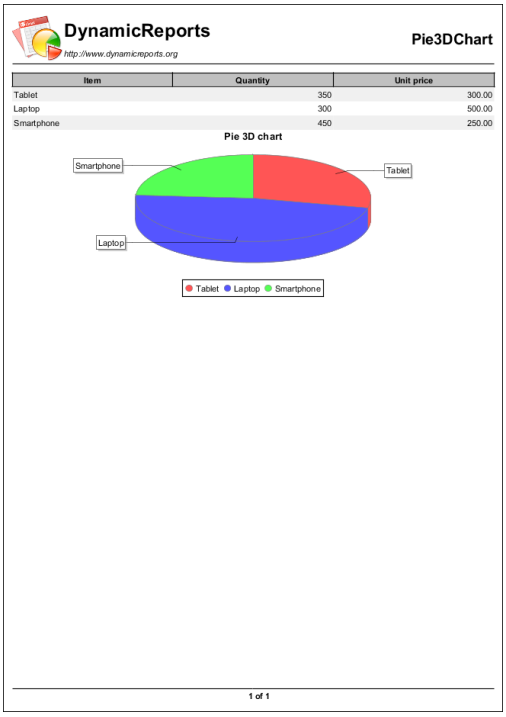
```
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.chart.CategoryChartSerieBuilder;
import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
import net.sf.dynamicreports.report.constant.AxisPosition;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class MultiAxisChartReport {

        public MultiAxisChartReport() {
                build();
        }

        private void build() {
                FieldBuilder<Date> dateField = field("date", type.dateType());
                FieldBuilder<Integer> stock1Field = field("stock1", type.
↪integerType());
                FieldBuilder<Integer> stock2Field = field("stock2", type.
↪integerType());

                CategoryChartSerieBuilder stock1Serie = cht.serie(stock1Field).
↪setLabel("Stock1");
                CategoryChartSerieBuilder stock2Serie = cht.serie(stock2Field).
↪setLabel("Stock2");

                LineChartBuilder chart1 = cht.lineChart()
                                .setCategory(new CategoryExpression())
                                .series(stock1Serie)
                                .setValueAxisFormat(
                                                cht.axisFormat().setLabel("Stock1"));

                BarChartBuilder chart2 = cht.barChart()
                                .setCategory(new CategoryExpression())
                                .series(stock2Serie)
                                .setValueAxisFormat(
                                                cht.axisFormat().setLabel("Stock2"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(dateField)
                                        .title(
                                                        Templates.
↪createTitleComponent("MultiAxisChart"),
                                                        cht.multiAxisChart(chart1,␣
↪chart2),
                                                        cht.multiAxisChart()
                                                                        .
↪addChart(chart1, AxisPosition.LEFT_OR_TOP)
                                                                        .
↪addChart(chart2, AxisPosition.RIGHT_OR_BOTTOM))
```

(continues on next page)

(continued from previous page)

```
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("date", "stock1", "stock2
↪");
                dataSource.add(toDate(2010, 1), 25, 300);
                dataSource.add(toDate(2010, 2), 11, 450);
                dataSource.add(toDate(2010, 3), 17, 280);
                dataSource.add(toDate(2010, 4), 15, 620);
                dataSource.add(toDate(2010, 5), 30, 400);
                dataSource.add(toDate(2010, 6), 8, 320);
                dataSource.add(toDate(2010, 7), 25, 490);
                return dataSource;
        }

        private Date toDate(int year, int month) {
                Calendar c = Calendar.getInstance();
                c.clear();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                return c.getTime();
        }

        public static void main(String[] args) {
                new MultiAxisChartReport();
        }

        private class CategoryExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return type.dateYearToMonthType().valueToString("date",␣
↪reportParameters);
                }
        }
}
```

1.17.14 Pie 3D Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

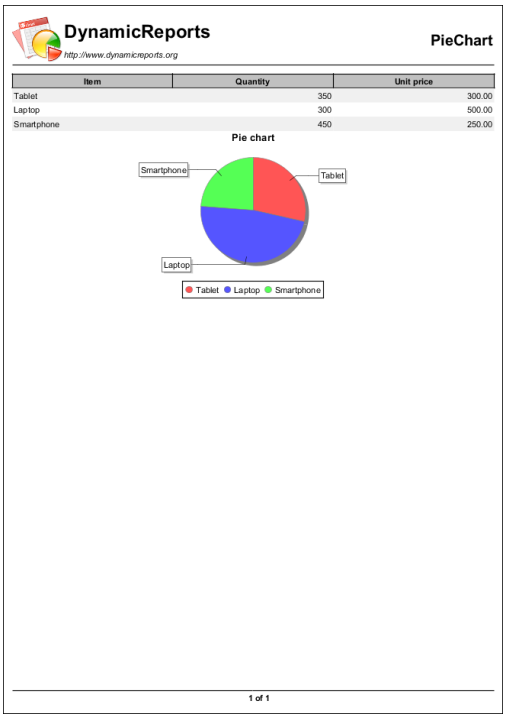
```
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class Pie3DChartReport {

        public Pie3DChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn,␣
↪unitPriceColumn)
                                        .title(Templates.createTitleComponent(
↪"Pie3DChart"))
                                        .summary(
                                                        cht.pie3DChart()
                                                                        .setTitle(
↪"Pie 3D chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setKey(itemColumn)
                                                                        .series(
                                                                                        cht.
↪serie(unitPriceColumn)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
```

(continues on next page)

(continued from previous page)

```
        }

        public static void main(String[] args) {
                new Pie3DChartReport();
        }
}
```

1.17.15 Pie Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
```

(continues on next page)

(continued from previous page)

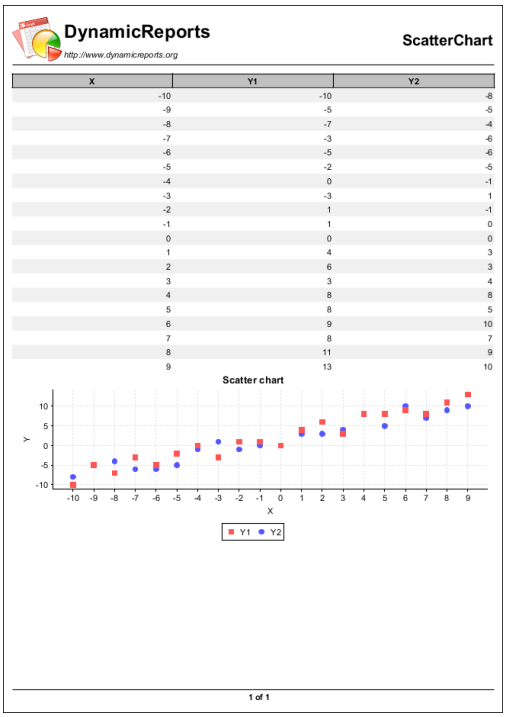
```
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PieChartReport {

        public PieChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn,␣
↪unitPriceColumn)
                                        .title(Templates.createTitleComponent(
↪"PieChart"))
                                        .summary(
                                                        cht.pieChart()
                                                                        .setTitle(
↪"Pie chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setKey(itemColumn)
                                                                        .series(
                                                                                        cht.
↪serie(unitPriceColumn)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
```

(continues on next page)

(continued from previous page)

```
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new PieChartReport();
        }
}
```

1.17.16 Scatter Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

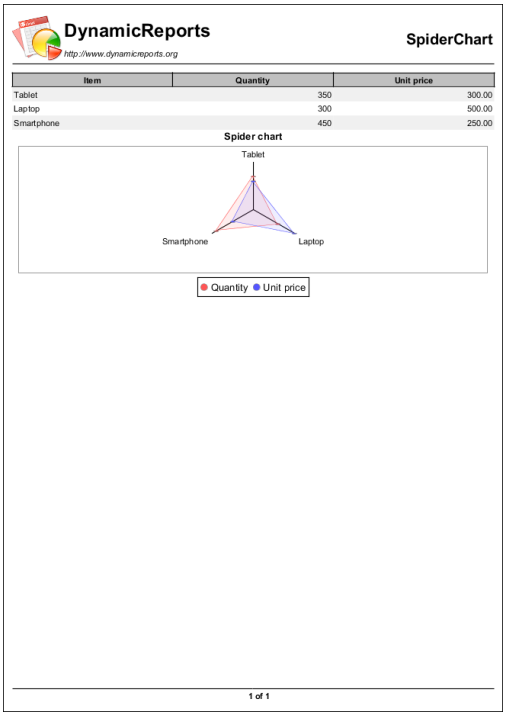
```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ScatterChartReport {

        public ScatterChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
                TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↪integerType());
                TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↪integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(xColumn, y1Column, y2Column)
                                        .title(Templates.createTitleComponent(
↪"ScatterChart"))
                                        .summary(
                                                        cht.scatterChart()
                                                                        .setTitle(
↪"Scatter chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setShowLines(false)
                                                                        .
↪setXValue(xColumn)
```

(continues on next page)

(continued from previous page)

```
                                                                        .series(
                                                                                cht.
↪xySerie(y1Column), cht.xySerie(y2Column))
                                                                        .
↪setXAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("X"))
                                                                        .
↪setYAxisFormat(
                                                                                cht.
↪axisFormat().setLabel("Y")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
                for (int i = -10; i < 10; i++) {
                        dataSource.add(i, i + (int) (Math.random() * 5), i + (int)␣
↪(Math.random() * 5));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new ScatterChartReport();
        }
}
```

1.17.17 Spider Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

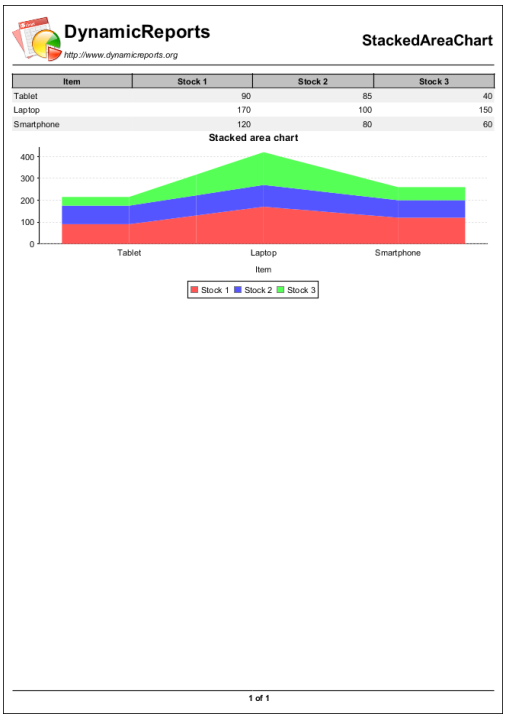
```
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SpiderChartReport {

        public SpiderChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("SpiderChart"))
                                        .summary(
                                                        cht.spiderChart()
                                                                        .setTitle("Spider chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setCategory(itemColumn)
                                                                        .series(cht.serie(quantityColumn), cht.serie(unitPriceColumn)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
```

(continues on next page)

(continued from previous page)

```
        }

        public static void main(String[] args) {
                new SpiderChartReport();
        }
}
```

1.17.18 Stacked Area Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
```

(continues on next page)

(continued from previous page)

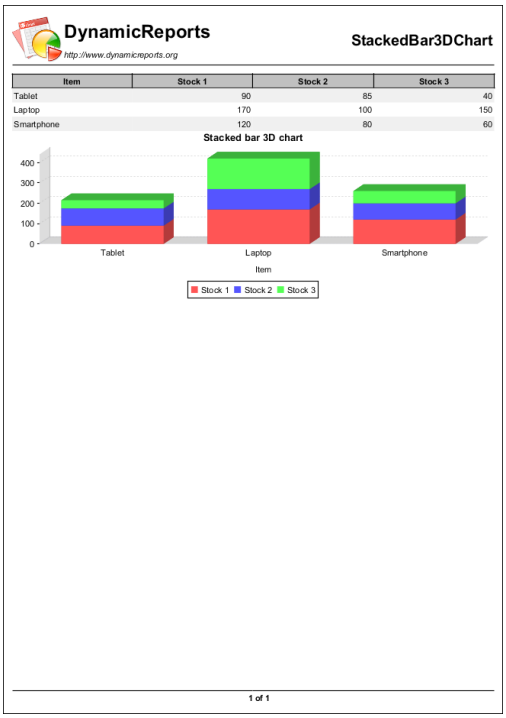
```
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class StackedAreaChartReport {

        public StackedAreaChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪"stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪"stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪"stock3", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, stock1Column,␣
↪stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent(
↪"StackedAreaChart"))
                                        .summary(
                                                        cht.stackedAreaChart()
                                                                        .setTitle(
↪"Stacked area chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setCategory(itemColumn)
                                                                        .series(
                                                                                        cht.
↪serie(stock1Column), cht.serie(stock2Column), cht.serie(stock3Column))
                                                                        .
↪setCategoryAxisFormat(
                                                                                        cht.
↪axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
```

(continues on next page)

(continued from previous page)

```
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↪ "stock3");
                dataSource.add("Tablet", 90, 85, 40);
                dataSource.add("Laptop", 170, 100, 150);
                dataSource.add("Smartphone", 120, 80, 60);
                return dataSource;
        }

        public static void main(String[] args) {
                new StackedAreaChartReport();
        }
}
```

1.17.19 Stacked Bar 3D Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

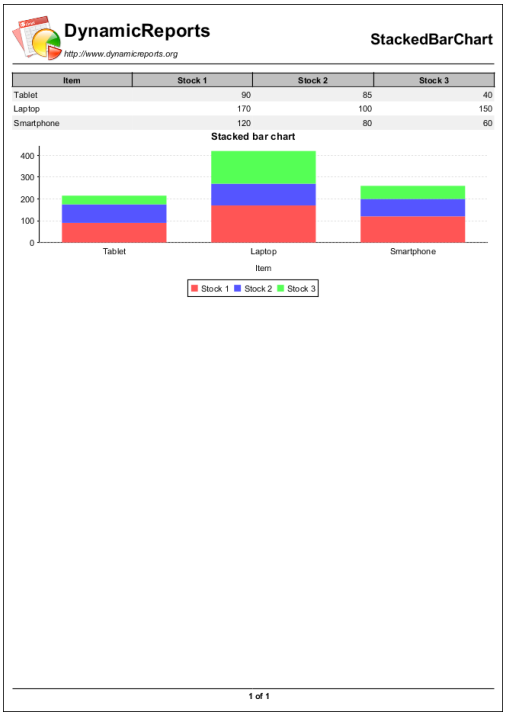
```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class StackedBar3DChartReport {

        public StackedBar3DChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪"stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪"stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪"stock3", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, stock1Column,␣
↪stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent(
↪"StackedBar3DChart"))
                                        .summary(
                                                        cht.stackedBar3DChart()
                                                                        .setTitle(
↪"Stacked bar 3D chart")
```

(continues on next page)

(continued from previous page)

```
58                                                                                .
↪setTitleFont(boldFont)
59                                                                                .
↪setCategory(itemColumn)
60                                                                                .series(
61                                                                                cht.
↪serie(stock1Column), cht.serie(stock2Column), cht.serie(stock3Column))
62                                                                                .
↪setCategoryAxisFormat(
63                                                                                cht.
↪axisFormat().setLabel("Item")))
64                                              .pageFooter(Templates.footerComponent)
65                                              .setDataSource(createDataSource())
66                                              .show();
67                  } catch (DREException e) {
68                          e.printStackTrace();
69                  }
70          }
71
72          private JRDataSource createDataSource() {
73                  DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↪ "stock3");
74                  dataSource.add("Tablet", 90, 85, 40);
75                  dataSource.add("Laptop", 170, 100, 150);
76                  dataSource.add("Smartphone", 120, 80, 60);
77                  return dataSource;
78          }
79
80          public static void main(String[] args) {
81                  new StackedBar3DChartReport();
82          }
83  }
```

1.17.20 Stacked Bar Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

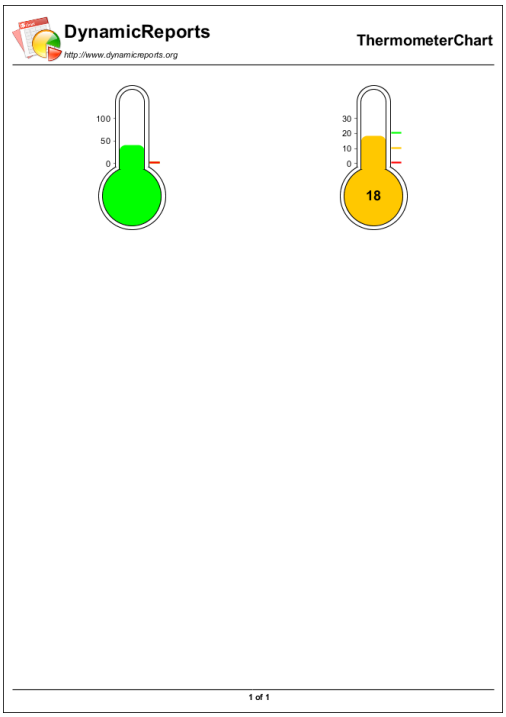
```
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class StackedBarChartReport {

        public StackedBarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1", "stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2", "stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3", "stock3", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, stock1Column, stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent("StackedBarChart"))
                                        .summary(
                                                        cht.stackedBarChart()
                                                                        .setTitle("Stacked bar chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setCategory(itemColumn)
                                                                        .series(cht.serie(stock1Column), cht.serie(stock2Column), cht.serie(stock3Column))
                                                                        .setCategoryAxisFormat(cht.axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2", "stock3");
                dataSource.add("Tablet", 90, 85, 40);
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Laptop", 170, 100, 150);
                dataSource.add("Smartphone", 120, 80, 60);
                return dataSource;
        }

        public static void main(String[] args) {
                new StackedBarChartReport();
        }
}
```

1.17.21 Thermometer Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
```

(continues on next page)

(continued from previous page)

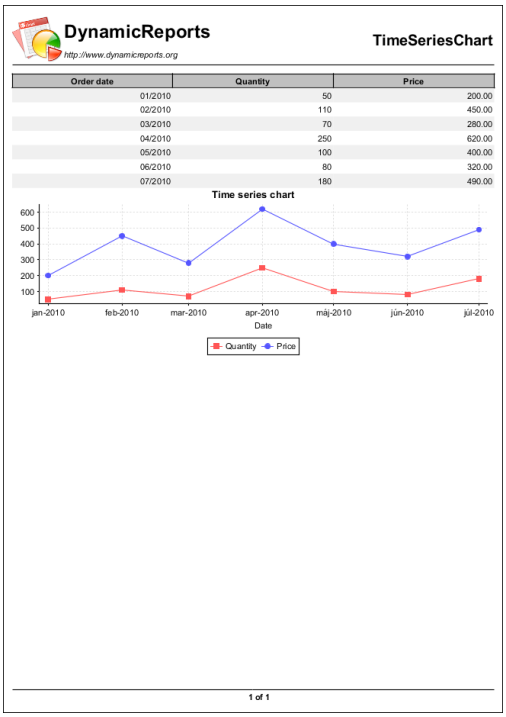
```
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.DynamicReports;
import net.sf.dynamicreports.report.builder.chart.ThermometerChartBuilder;
import net.sf.dynamicreports.report.constant.ValueLocation;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ThermometerChartReport {

        public ThermometerChartReport() {
                build();
        }

        private void build() {
                ThermometerChartBuilder chart1 = cht.thermometerChart()
                                .setValue(DynamicReports.<Number>field("value", type.
↪integerType()));

                ThermometerChartBuilder chart2 = cht.thermometerChart()
                                .setValue(18)
                                .setDataRangeHighExpression(30)
                                .setValueColor(Color.BLACK)
                                .setValueLocation(ValueLocation.BULB)
                                .setLowDataRangeLowExpression(0)
                                .setLowDataRangeHighExpression(10)
                                .setMediumDataRangeLowExpression(10)
                                .setMediumDataRangeHighExpression(20)
                                .setHighDataRangeLowExpression(20)
                                .setHighDataRangeHighExpression(30);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"ThermometerChart"))
                                        .summary(
                                                        cmp.horizontalList(chart1,␣
↪chart2))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
```

(continues on next page)

(continued from previous page)

```
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("value");
                dataSource.add(40);
                return dataSource;
        }

        public static void main(String[] args) {
                new ThermometerChartReport();
        }
}
```

1.17.22 Time-Series Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
```

(continues on next page)

(continued from previous page)

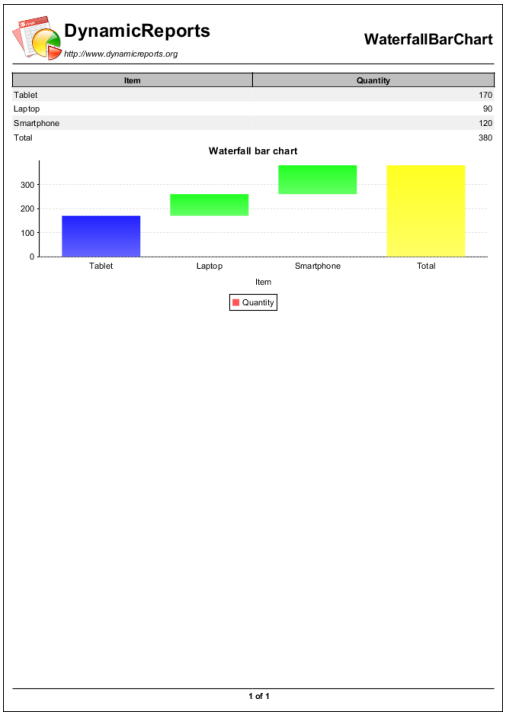
```
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.constant.TimePeriod;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TimeSeriesChartReport {

        public TimeSeriesChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateYearToMonthType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(orderDateColumn, quantityColumn,␣
↪priceColumn)
                                        .title(Templates.createTitleComponent(
↪"TimeSeriesChart"))
                                        .summary(
                                                        cht.timeSeriesChart()
                                                                        .setTitle(
↪"Time series chart")
                                                                        .
↪setTitleFont(boldFont)
```

(continues on next page)

(continued from previous page)

```
                                                                                .
↪setTimePeriod(orderDateColumn)
                                                                                .
↪setTimePeriodType(TimePeriod.MONTH)
                                                                                .series(
                                                                                                cht.
↪serie(quantityColumn), cht.serie(priceColumn))
                                                                                .
↪setTimeAxisFormat(
                                                                                                cht.
↪axisFormat().setLabel("Date")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("orderdate", "quantity",
↪"price");
                dataSource.add(toDate(2010, 1), 50, new BigDecimal(200));
                dataSource.add(toDate(2010, 2), 110, new BigDecimal(450));
                dataSource.add(toDate(2010, 3), 70, new BigDecimal(280));
                dataSource.add(toDate(2010, 4), 250, new BigDecimal(620));
                dataSource.add(toDate(2010, 5), 100, new BigDecimal(400));
                dataSource.add(toDate(2010, 6), 80, new BigDecimal(320));
                dataSource.add(toDate(2010, 7), 180, new BigDecimal(490));
                return dataSource;
        }

        private Date toDate(int year, int month) {
                Calendar c = Calendar.getInstance();
                c.clear();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                return c.getTime();
        }

        public static void main(String[] args) {
                new TimeSeriesChartReport();
        }
}
```

1.17.23 Waterfall Bar Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

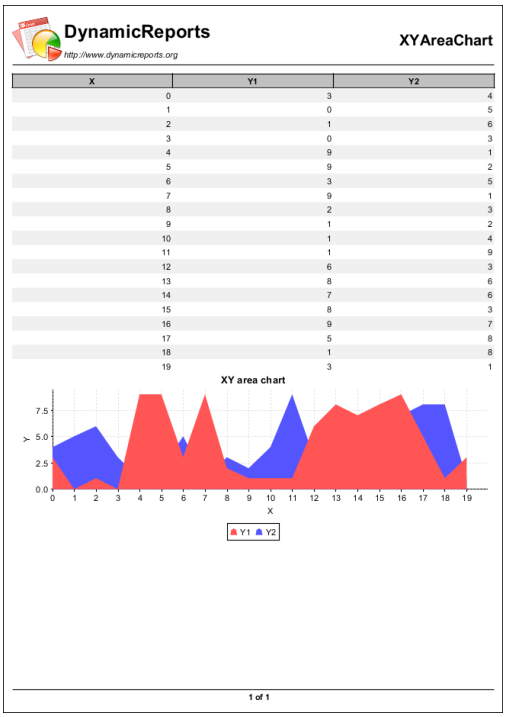
```
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class WaterfallBarChartReport {

        public WaterfallBarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn)
                                        .title(Templates.createTitleComponent("WaterfallBarChart"))
                                        .summary(
                                                        cht.waterfallBarChart()
                                                                        .setTitle("Waterfall bar chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setCategory(itemColumn)
                                                                        .series(cht.serie(quantityColumn))
                                                                        .setCategoryAxisFormat(cht.axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity");
                dataSource.add("Tablet", 170);
                dataSource.add("Laptop", 90);
                dataSource.add("Smartphone", 120);
                dataSource.add("Total", 380);
                return dataSource;
        }

        public static void main(String[] args) {
```

(continues on next page)

(continued from previous page)

```
                new WaterfallBarChartReport();
        }
}
```

1.17.24 XY-Area Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;
```

(continues on next page)

(continued from previous page)

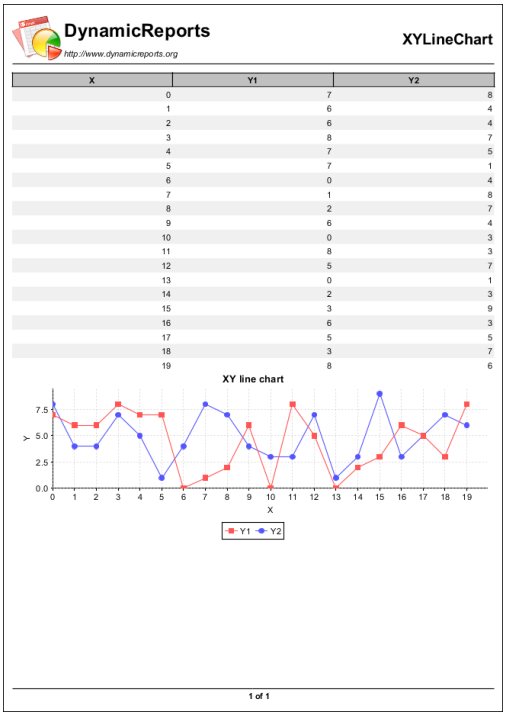
```
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28  import net.sf.dynamicreports.report.builder.style.FontBuilder;
29  import net.sf.dynamicreports.report.datasource.DRDataSource;
30  import net.sf.dynamicreports.report.exception.DRException;
31  import net.sf.jasperreports.engine.JRDataSource;
32
33  /**
34   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35   */
36  public class XYAreaChartReport {
37
38          public XYAreaChartReport() {
39                  build();
40          }
41
42          private void build() {
43                  FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45                  TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
    ↪integerType());
46                  TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
    ↪integerType());
47                  TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
    ↪integerType());
48
49                  try {
50                          report()
51                                          .setTemplate(Templates.reportTemplate)
52                                          .columns(xColumn, y1Column, y2Column)
53                                          .title(Templates.createTitleComponent(
    ↪"XYAreaChart"))
54                                          .summary(
55                                                          cht.xyAreaChart()
56                                                                          .setTitle("XY␣
    ↪area chart")
57                                                                          .
    ↪setTitleFont(boldFont)
58                                                                          .
    ↪setXValue(xColumn)
59                                                                          .series(
60                                                                                          cht.
    ↪xySerie(y1Column), cht.xySerie(y2Column))
61                                                                          .
    ↪setXAxisFormat(
62                                                                                          cht.
    ↪axisFormat().setLabel("X"))
63                                                                          .
    ↪setYAxisFormat(
64                                                                                          cht.
    ↪axisFormat().setLabel("Y")))
65                                          .pageFooter(Templates.footerComponent)
66                                          .setDataSource(createDataSource())
67                                          .show();
68                  } catch (DRException e) {
```

(continues on next page)

(continued from previous page)

```
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
                for (int i = 0; i < 20; i++) {
                        dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
↪random() * 10));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new XYAreaChartReport();
        }
}
```

1.17.25 XY-Line Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
```

(continues on next page)

(continued from previous page)

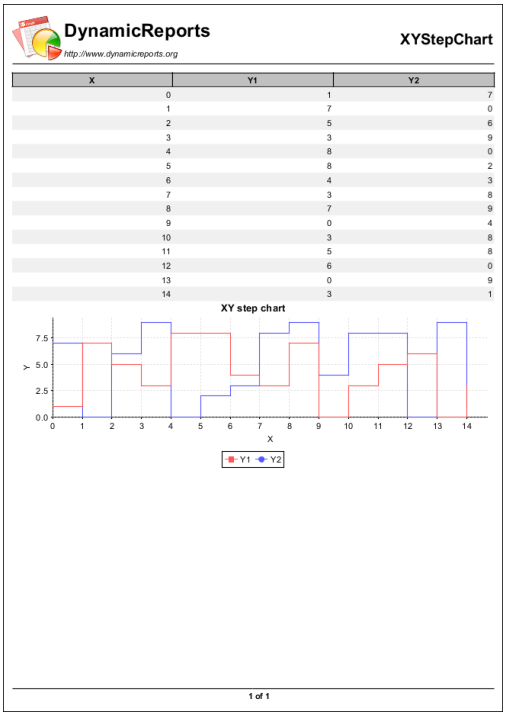
```
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class XYLineChartReport {

        public XYLineChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
                TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↪integerType());
                TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↪integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(xColumn, y1Column, y2Column)
                                        .title(Templates.createTitleComponent(
↪"XYLineChart"))
                                        .summary(
                                                        cht.xyLineChart()
                                                                        .setTitle("XY␣
↪line chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setXValue(xColumn)
                                                                        .series(
```

(continues on next page)

(continued from previous page)

```
                cht.xySerie(y1Column), cht.xySerie(y2Column))
                                                                .setXAxisFormat(
                cht.axisFormat().setLabel("X"))
                                                                .setYAxisFormat(
                cht.axisFormat().setLabel("Y")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
                for (int i = 0; i < 20; i++) {
                        dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
random() * 10));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new XYLineChartReport();
        }
}
```

1.17.26 XY-Step Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

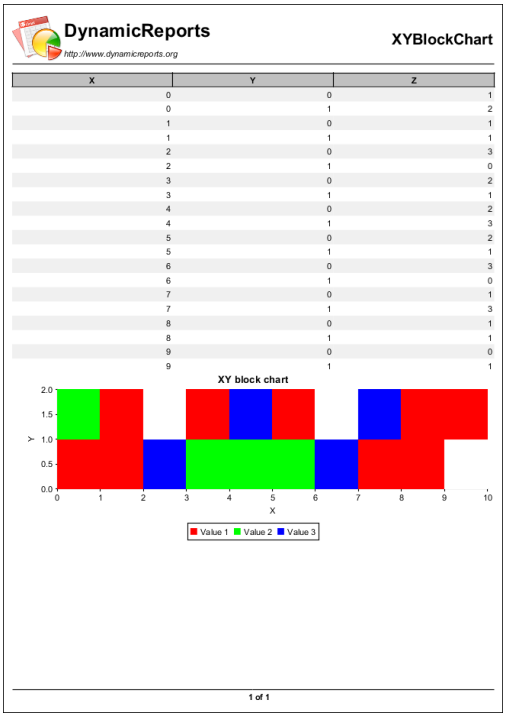
```
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class XYStepChartReport {

        public XYStepChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.integerType());
                TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.integerType());
                TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(xColumn, y1Column, y2Column)
                                        .title(Templates.createTitleComponent("XYStepChart"))
                                        .summary(
                                                        cht.xyStepChart()
                                                                        .setTitle("XY step chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setXValue(xColumn)
                                                                        .series(cht.xySerie(y1Column), cht.xySerie(y2Column))
                                                                        .setXAxisFormat(cht.axisFormat().setLabel("X"))
                                                                        .setYAxisFormat(cht.axisFormat().setLabel("Y")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
                for (int i = 0; i < 15; i++) {
```

(continues on next page)

(continued from previous page)

```
                        dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
↪random() * 10));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new XYStepChartReport();
        }
}
```

1.17.27 XY-Block Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

(continues on next page)

(continued from previous page)

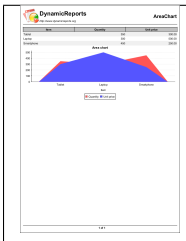
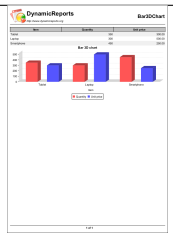
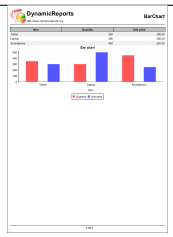
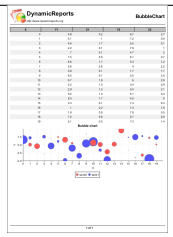
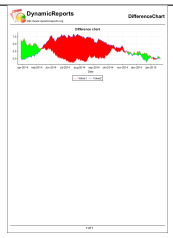
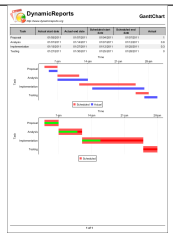
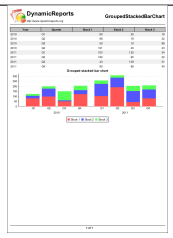
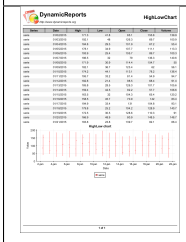
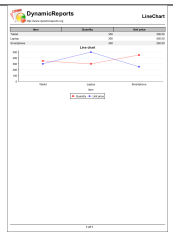
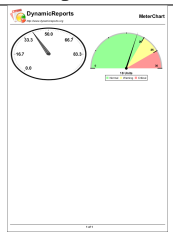
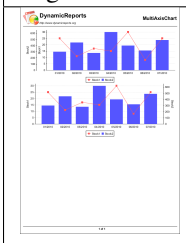
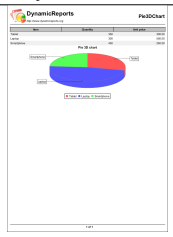
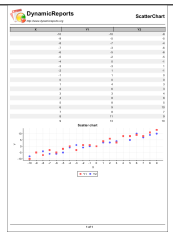
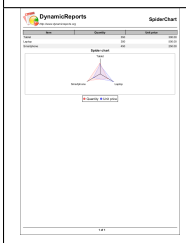
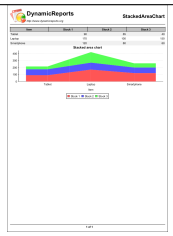
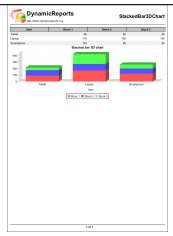
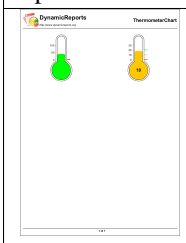
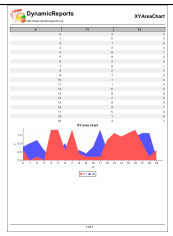
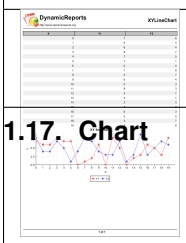
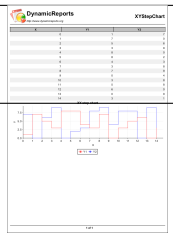
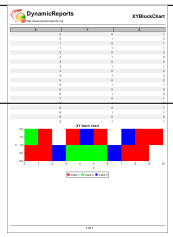
```
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.constant.RectangleAnchor;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class XyBlockChartReport {

        public XyBlockChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.integerType());
                TextColumnBuilder<Integer> yColumn = col.column("Y", "y", type.integerType());
                TextColumnBuilder<Integer> zColumn = col.column("Z", "z", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(xColumn, yColumn, zColumn)
                                        .title(Templates.createTitleComponent("XYBlockChart"))
                                        .summary(
                                                        cht.xyBlockChart(0, 5, Color.WHITE)
                                                                        .setTitle("XY block chart")
                                                                        .setTitleFont(boldFont)
                                                                        .setBlockAnchor(RectangleAnchor.BOTTOM_LEFT)
                                                                        .paintScales(
                                                                                        cht.paintScale("Value 1", 1, Color.RED),
```

(continues on next page)

(continued from previous page)

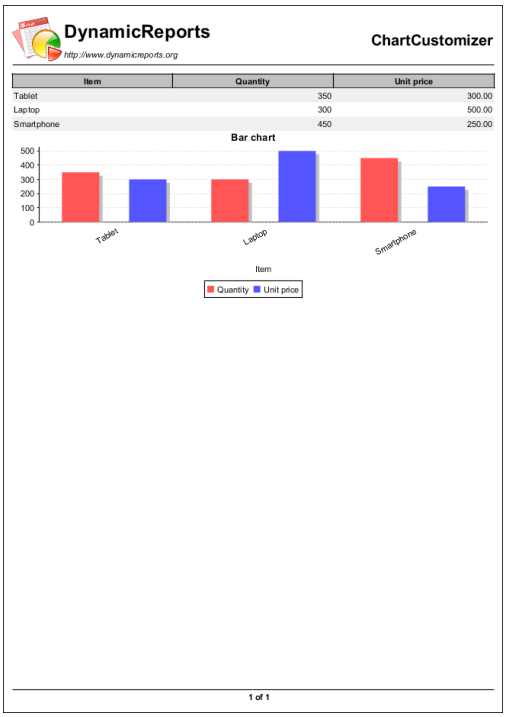
```
                                                                                        cht.paintScale("Value 2", 2, Color.GREEN),
                                                                                        cht.paintScale("Value 3", 3, Color.BLUE))
                                                                                .setXValue(xColumn)
                                                                                .series(
                                                                                        cht.xyzSerie().setYValue(yColumn).setZValue(zColumn))
                                                                                .setXAxisFormat(
                                                                                        cht.axisFormat().setLabel("X"))
                                                                                .setYAxisFormat(
                                                                                        cht.axisFormat().setLabel("Y")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("x", "y", "z");
                for (int i = 0; i < 10; i++) {
                        dataSource.add(i, 0, (int) (Math.random() * 4));
                        dataSource.add(i, 1, (int) (Math.random() * 4));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new XyBlockChartReport();
        }
}
```

Table 2: Chart Examples

 AreaChart	 Bar3DChart	 BarChart	 BubbleChart
 CandlestickChart	 DifferenceChart	 GanttChart	 GroupedStackedBar
 HighLowChart	 LayeredBarChart	 LineChart	 MeterChart
 MultiAxisChart	 Pie3DChart	 PieChart	 ScatterChart
 SpiderChart	 StackedAreaChart	 StackedBar3DChart	 StackedBarChart
 ThermometerChart	 TimeSeriesChart	 WaterfallBarChart	 XYAreaChart
			

1.18 Chart customization

1.18.1 Chart Customizer




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
```

(continues on next page)

(continued from previous page)

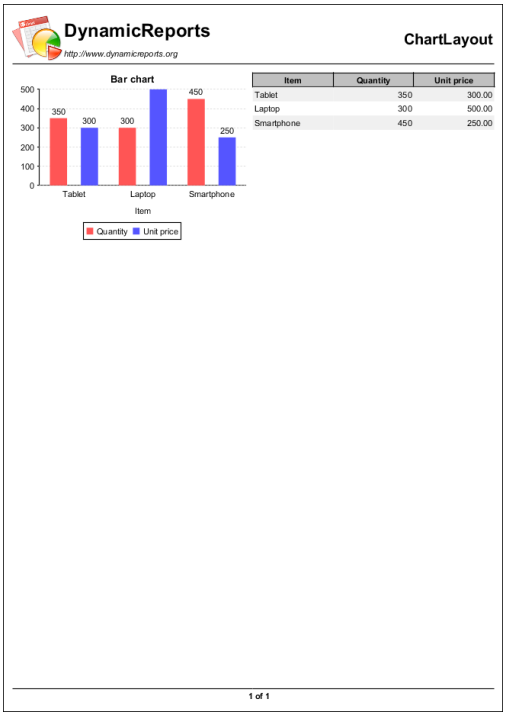
```
import java.io.Serializable;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.definition.chart.DRIChartCustomizer;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.CategoryAxis;
import org.jfree.chart.axis.CategoryLabelPositions;
import org.jfree.chart.renderer.category.BarRenderer;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartCustomizerReport {

        public ChartCustomizerReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn,␣
↪unitPriceColumn)
                                        .title(Templates.createTitleComponent(
↪"ChartCustomizer"))
                                        .summary(
                                                        cht.barChart()
                                                                        .
↪customizers(new ChartCustomizer())
                                                                        .setTitle(
↪"Bar chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setCategory(itemColumn)
                                                                        .series(
                                                                                        cht.
↪serie(quantityColumn), cht.serie(unitPriceColumn))
                                                                        .
↪setCategoryAxisFormat(
```

(continues on next page)

(continued from previous page)

```
                                                                                cht.
↪axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class ChartCustomizer implements DRIChartCustomizer, Serializable {
                private static final long serialVersionUID = 1L;

                @Override
                public void customize(JFreeChart chart, ReportParameters␣
↪reportParameters) {
                        BarRenderer renderer = (BarRenderer) chart.getCategoryPlot().
↪getRenderer();
                        renderer.setShadowPaint(Color.LIGHT_GRAY);
                        renderer.setShadowVisible(true);
                        CategoryAxis domainAxis = chart.getCategoryPlot().
↪getDomainAxis();
                        domainAxis.setCategoryLabelPositions(CategoryLabelPositions.
↪createUpRotationLabelPositions(Math.PI / 6.0));
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new ChartCustomizerReport();
        }
}
```

1.18.2 Chart Layout




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.Serializable;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

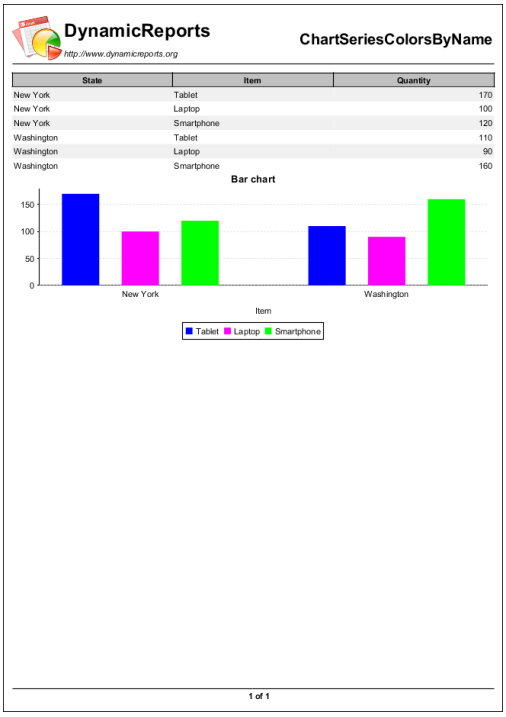
```
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.definition.chart.DRIChartCustomizer;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

import org.jfree.chart.JFreeChart;
import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
import org.jfree.chart.renderer.category.BarRenderer;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartLayoutReport {

        public ChartLayoutReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                JasperReportBuilder subreport = report();
                subreport
                                .setTemplate(Templates.reportTemplate)
                                .columns(itemColumn, quantityColumn, unitPriceColumn)
                                .setDataSource(createDataSource());

                BarChartBuilder chart = cht.barChart()
                                .customizers(new ChartCustomizer())
                                .setTitle("Bar chart")
                                .setTitleFont(boldFont)
                                .setCategory(itemColumn)
                                .series(
                                                cht.serie(quantityColumn), cht.
↪serie(unitPriceColumn))
                                .setValueAxisFormat(
                                                cht.axisFormat().
↪setRangeMaxValueExpression(500))
                                .setCategoryAxisFormat(
                                                cht.axisFormat().setLabel("Item"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"ChartLayout"))
```

(continues on next page)

(continued from previous page)

```
                                        .summary(
                                                cmp.horizontalList(chart, cmp.subreport(subreport)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class ChartCustomizer implements DRIChartCustomizer, Serializable {
                private static final long serialVersionUID = 1L;

                @Override
                public void customize(JFreeChart chart, ReportParameters reportParameters) {
                        BarRenderer renderer = (BarRenderer) chart.getCategoryPlot().getRenderer();
                        renderer.setBaseItemLabelGenerator(new StandardCategoryItemLabelGenerator());
                        renderer.setBaseItemLabelsVisible(true);
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                dataSource.add("Tablet", 350, new BigDecimal(300));
                dataSource.add("Laptop", 300, new BigDecimal(500));
                dataSource.add("Smartphone", 450, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new ChartLayoutReport();
        }
}
```

1.18.3 Chart Series Colors By Name




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.util.HashMap;
import java.util.Map;

```

(continues on next page)

(continued from previous page)

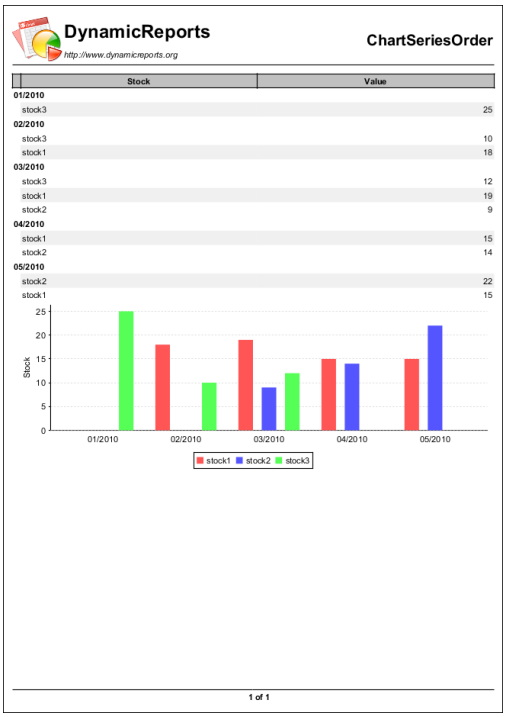
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartSeriesColorsByNameReport {

        public ChartSeriesColorsByNameReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> stateColumn = col.column("State", "state", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());

                Map<String, Color> seriesColors = new HashMap<String, Color>();
                seriesColors.put("Tablet", Color.BLUE);
                seriesColors.put("Smartphone", Color.GREEN);
                seriesColors.put("Laptop", Color.MAGENTA);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(stateColumn, itemColumn, quantityColumn)
                                        .title(Templates.createTitleComponent("ChartSeriesColorsByName"))
                                        .summary(
                                                        cht.barChart()
                                                                        .setTitle("Bar chart")
                                                                        .setTitleFont(boldFont)
                                                                        .seriesColorsByName(seriesColors)
                                                                        .setCategory(stateColumn)
                                                                        .series(
                                                                                        cht.serie(quantityColumn).setSeries(itemColumn))
                                                                        .setCategoryAxisFormat(
                                                                                        cht.axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
```

(continues on next page)

(continued from previous page)

```
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪");
                dataSource.add("New York", "Tablet", 170);
                dataSource.add("New York", "Laptop", 100);
                dataSource.add("New York", "Smartphone", 120);
                dataSource.add("Washington", "Tablet", 110);
                dataSource.add("Washington", "Laptop", 90);
                dataSource.add("Washington", "Smartphone", 160);
                return dataSource;
        }

        public static void main(String[] args) {
                new ChartSeriesColorsByNameReport();
        }
}
```

1.18.4 Chart Series Order




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

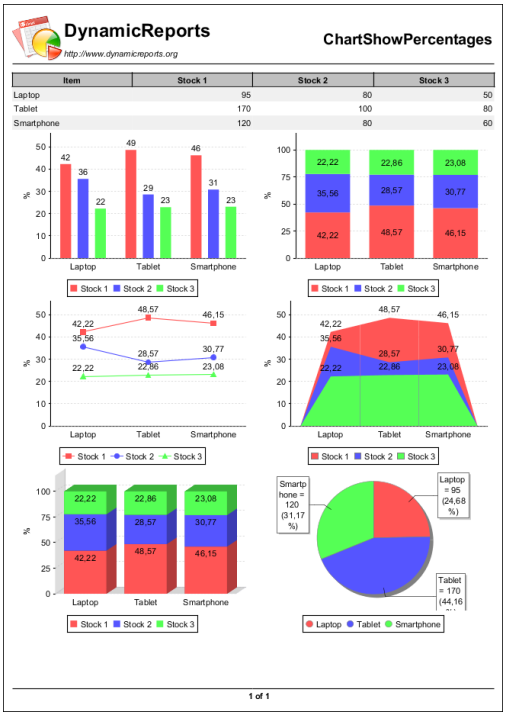
```
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.constant.OrderType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartSeriesOrderReport {

        public ChartSeriesOrderReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
↪dateYearToMonthType());
                TextColumnBuilder<String> stockColumn = col.column("Stock", "stock",␣
↪type.stringType());
                TextColumnBuilder<Integer> valueColumn = col.column("Value", "value",␣
↪type.integerType());

                BarChartBuilder chart = cht.barChart()
                                .setCategory(new CategoryExpression())
                                .series(cht.serie(valueColumn).setSeries(stockColumn))
                                .setSeriesOrderType(OrderType.ASCENDING)
                                .setValueAxisFormat(
```

(continues on next page)

(continued from previous page)

```
                                                cht.axisFormat().setLabel("Stock"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(dateColumn, stockColumn, valueColumn)
                                        .title(Templates.createTitleComponent(
↪"ChartSeriesOrder"))
                                        .groupBy(dateColumn)
                                        .summary(chart)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("date", "stock", "value");
                dataSource.add(toDate(2010, 1), "stock3", 25);
                dataSource.add(toDate(2010, 2), "stock3", 10);
                dataSource.add(toDate(2010, 2), "stock1", 18);
                dataSource.add(toDate(2010, 3), "stock3", 12);
                dataSource.add(toDate(2010, 3), "stock1", 19);
                dataSource.add(toDate(2010, 3), "stock2", 9);
                dataSource.add(toDate(2010, 4), "stock1", 15);
                dataSource.add(toDate(2010, 4), "stock2", 14);
                dataSource.add(toDate(2010, 5), "stock2", 22);
                dataSource.add(toDate(2010, 5), "stock1", 15);
                return dataSource;
        }

        private Date toDate(int year, int month) {
                Calendar c = Calendar.getInstance();
                c.clear();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                return c.getTime();
        }

        public static void main(String[] args) {
                new ChartSeriesOrderReport();
        }

        private class CategoryExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return type.dateYearToMonthType().valueToString("date",␣
↪reportParameters);
                }
        }
}
```

1.18.5 Chart Show Percentages




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.chart.AreaChartBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
import net.sf.dynamicreports.report.builder.chart.PieChartBuilder;
```

(continues on next page)

(continued from previous page)

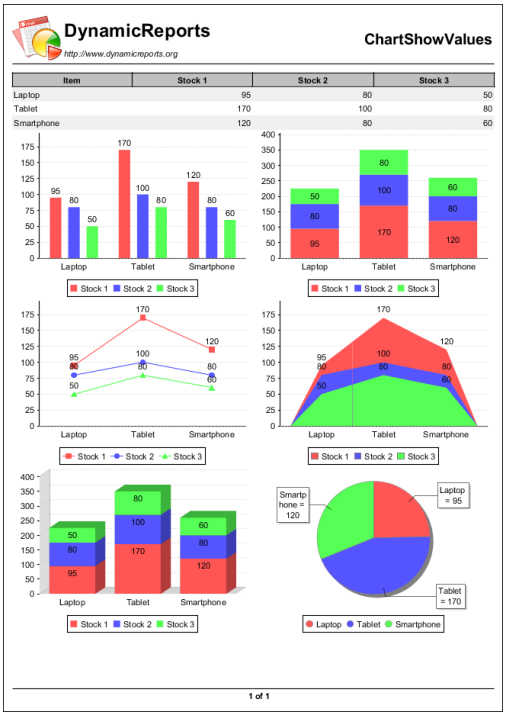
```
import net.sf.dynamicreports.report.builder.chart.StackedBar3DChartBuilder;
import net.sf.dynamicreports.report.builder.chart.StackedBarChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartShowPercentagesReport {

        public ChartShowPercentagesReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪"stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪"stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪"stock3", type.integerType());

                BarChartBuilder barChart = cht.barChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setPercentValuePattern("#,##0")
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                StackedBarChartBuilder stackedBarChart = cht.stackedBarChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                LineChartBuilder lineChart = cht.lineChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                AreaChartBuilder areaChart = cht.areaChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                StackedBar3DChartBuilder stackedBar3DChart = cht.stackedBar3DChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
```

(continues on next page)

(continued from previous page)

```
                PieChartBuilder pieChart = cht.pieChart()
                                .setShowPercentages(true)
                                .setShowValues(true)
                                .setKey(itemColumn)
                                .series(cht.serie(stock1Column));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, stock1Column, stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent("ChartShowPercentages"))
                                        .summary(
                                                        cmp.horizontalList(barChart, stackedBarChart),
                                                        cmp.horizontalList(lineChart, areaChart),
                                                        cmp.horizontalList(stackedBar3DChart, pieChart))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2", "stock3");
                dataSource.add("Laptop", 95, 80, 50);
                dataSource.add("Tablet", 170, 100, 80);
                dataSource.add("Smartphone", 120, 80, 60);
                return dataSource;
        }

        public static void main(String[] args) {
                new ChartShowPercentagesReport();
        }
}
```

1.18.6 Chart Show Values




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.chart.AreaChartBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
import net.sf.dynamicreports.report.builder.chart.PieChartBuilder;
```

(continues on next page)

(continued from previous page)

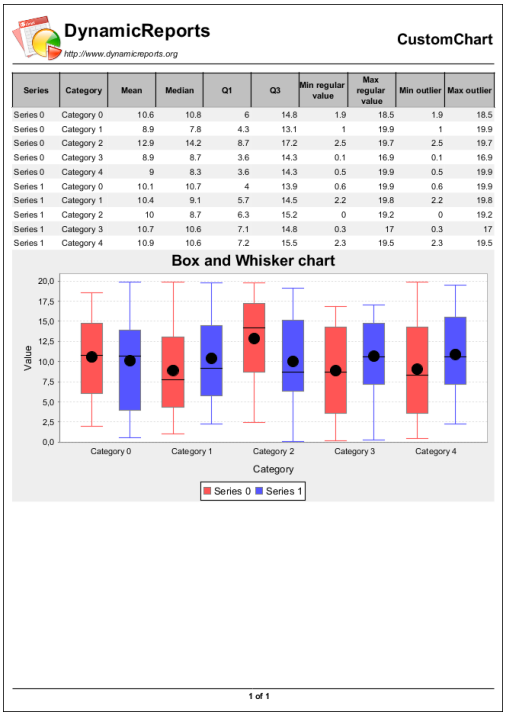
```
import net.sf.dynamicreports.report.builder.chart.StackedBar3DChartBuilder;
import net.sf.dynamicreports.report.builder.chart.StackedBarChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ChartShowValuesReport {

        public ChartShowValuesReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪"stock1", type.integerType());
                TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪"stock2", type.integerType());
                TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪"stock3", type.integerType());

                BarChartBuilder barChart = cht.barChart()
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                StackedBarChartBuilder stackedBarChart = cht.stackedBarChart()
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                LineChartBuilder lineChart = cht.lineChart()
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                AreaChartBuilder areaChart = cht.areaChart()
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                StackedBar3DChartBuilder stackedBar3DChart = cht.stackedBar3DChart()
                                .setShowValues(true)
                                .setCategory(itemColumn)
                                .series(cht.serie(stock1Column), cht.
↪serie(stock2Column), cht.serie(stock3Column));
                PieChartBuilder pieChart = cht.pieChart()
                                .setShowValues(true)
                                .setKey(itemColumn)
                                .series(cht.serie(stock1Column));

                try {
```

(continues on next page)

(continued from previous page)

```
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, stock1Column,␣
↪stock2Column, stock3Column)
                                        .title(Templates.createTitleComponent(
↪"ChartShowValues"))
                                        .summary(
                                                        cmp.horizontalList(barChart,␣
↪stackedBarChart),
                                                        cmp.horizontalList(lineChart,␣
↪areaChart),
                                                        cmp.
↪horizontalList(stackedBar3DChart, pieChart))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↪ "stock3");
                dataSource.add("Laptop", 95, 80, 50);
                dataSource.add("Tablet", 170, 100, 80);
                dataSource.add("Smartphone", 120, 80, 60);
                return dataSource;
        }

        public static void main(String[] args) {
                new ChartShowValuesReport();
        }
}
```

1.18.7 Custom Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.data.statistics.BoxAndWhiskerItem;
import org.jfree.data.statistics.DefaultBoxAndWhiskerCategoryDataset;
```

(continues on next page)

(continued from previous page)

```
31
32  import net.sf.dynamicreports.examples.Templates;
33  import net.sf.dynamicreports.report.base.AbstractScriptlet;
34  import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
35  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
36  import net.sf.dynamicreports.report.datasource.DRDataSource;
37  import net.sf.dynamicreports.report.definition.ReportParameters;
38  import net.sf.dynamicreports.report.exception.DRException;
39  import net.sf.jasperreports.charts.util.DrawChartRendererImpl;
40  import net.sf.jasperreports.engine.JRDataSource;
41  import net.sf.jasperreports.renderers.Renderable;
42
43  /**
44   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
45   */
46  public class CustomChartReport {
47          private DefaultBoxAndWhiskerCategoryDataset dataset = new␣
    ↪DefaultBoxAndWhiskerCategoryDataset();
48
49          public CustomChartReport() {
50                  build();
51          }
52
53          private void build() {
54                  TextColumnBuilder<String> seriesColumn = col.column("Series", "series
    ↪", type.stringType());
55                  TextColumnBuilder<String> categoryColumn = col.column("Category",
    ↪"category", type.stringType());
56                  TextColumnBuilder<Double> meanColumn = col.column("Mean", "mean",␣
    ↪type.doubleType());
57                  TextColumnBuilder<Double> medianColumn = col.column("Median", "median
    ↪", type.doubleType());
58                  TextColumnBuilder<Double> q1Column = col.column("Q1", "q1", type.
    ↪doubleType());
59                  TextColumnBuilder<Double> q3Column = col.column("Q3", "q3", type.
    ↪doubleType());
60                  TextColumnBuilder<Double> minRegularValueColumn = col.column("Min␣
    ↪regular value", "minregularvalue", type.doubleType());
61                  TextColumnBuilder<Double> maxRegularValueColumn = col.column("Max␣
    ↪regular value", "maxregularvalue", type.doubleType());
62                  TextColumnBuilder<Double> minOutlierColumn = col.column("Min outlier",
    ↪ "minoutlier", type.doubleType());
63                  TextColumnBuilder<Double> maxOutlierColumn = col.column("Max outlier",
    ↪ "maxoutlier", type.doubleType());
64
65                  try {
66                          report()
67                                          .setTemplate(Templates.reportTemplate)
68                                          .scriptlets(new ReportScriptlet())
69                                          .columns(seriesColumn, categoryColumn,␣
    ↪meanColumn, medianColumn, q1Column, q3Column, minRegularValueColumn,␣
    ↪maxRegularValueColumn, minOutlierColumn,
70                                                                          maxOutlierColumn)
71                                          .title(Templates.createTitleComponent(
    ↪"CustomChart"))
72                                          .summary(
73                                                          cmp.image(new␣
    ↪ChartExpression()).setFixedHeight(300))
```

(continues on next page)

(continued from previous page)

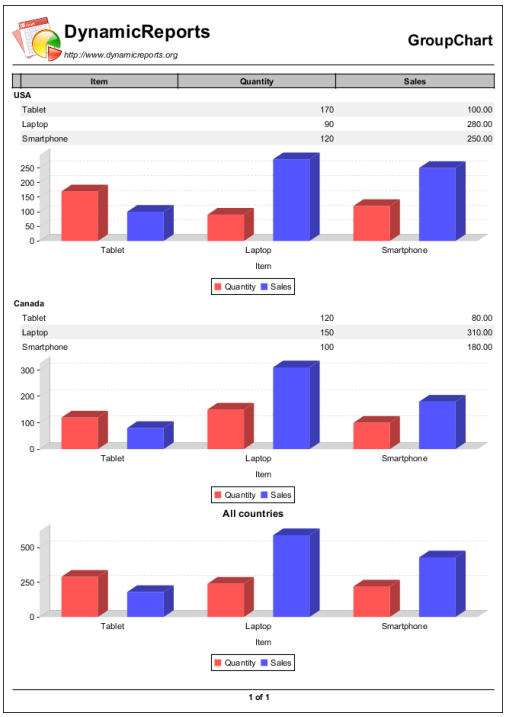
```
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("series", "category", "mean", "median", "q1", "q3", "minregularvalue", "maxregularvalue", "minoutlier",
                                "maxoutlier");
                dataSource.add("Series 0", "Category 0", 10.55d, 10.75d, 6.05d, 14.76d, 1.93d, 18.51d, 1.93d, 18.51d);
                dataSource.add("Series 0", "Category 1", 8.92d, 7.78d, 4.32d, 13.07d, 1.01d, 19.89d, 1.01d, 19.89d);
                dataSource.add("Series 0", "Category 2", 12.88d, 14.19d, 8.72d, 17.23d, 2.48d, 19.74d, 2.48d, 19.74d);
                dataSource.add("Series 0", "Category 3", 8.87d, 8.68d, 3.55d, 14.26d, 0.13d, 16.87d, 0.13d, 16.87d);
                dataSource.add("Series 0", "Category 4", 9.05d, 8.33d, 3.61d, 14.28d, 0.46d, 19.86d, 0.46d, 19.86d);
                dataSource.add("Series 1", "Category 0", 10.11d, 10.69d, 3.96d, 13.93d, 0.58d, 19.9d, 0.58d, 19.9d);
                dataSource.add("Series 1", "Category 1", 10.43d, 9.13d, 5.72d, 14.46d, 2.25d, 19.79d, 2.25d, 19.79d);
                dataSource.add("Series 1", "Category 2", 10.04d, 8.71d, 6.32d, 15.15d, 0.04d, 19.15d, 0.04d, 19.15d);
                dataSource.add("Series 1", "Category 3", 10.67d, 10.56d, 7.14d, 14.76d, 0.27d, 16.99d, 0.27d, 16.99d);
                dataSource.add("Series 1", "Category 4", 10.91d, 10.63d, 7.18d, 15.51d, 2.3d, 19.53d, 2.3d, 19.53d);
                return dataSource;
        }

        public static void main(String[] args) {
                new CustomChartReport();
        }

        private class ReportScriptlet extends AbstractScriptlet {

                @Override
                public void afterDetailEval(ReportParameters reportParameters) {
                        super.afterDetailEval(reportParameters);
                        String series = reportParameters.getValue("series");
                        String category = reportParameters.getValue("category");
                        Double mean = reportParameters.getValue("mean");
                        Double median = reportParameters.getValue("median");
                        Double q1 = reportParameters.getValue("q1");
                        Double q3 = reportParameters.getValue("q3");
                        Double minRegularValue = reportParameters.getValue("minregularvalue");
                        Double maxRegularValue = reportParameters.getValue("maxregularvalue");
                        Double minOutlier = reportParameters.getValue("minoutlier");
                        Double maxOutlier = reportParameters.getValue("maxoutlier");
                        BoxAndWhiskerItem item = new BoxAndWhiskerItem(mean, median, q1, q3, minRegularValue, maxRegularValue, minOutlier, maxOutlier, null);
```

(continues on next page)

(continued from previous page)

```
                        dataset.add(item, series, category);
                }
        }

        private class ChartExpression extends AbstractSimpleExpression<Renderable> {
                private static final long serialVersionUID = 1L;

                @Override
                public Renderable evaluate(ReportParameters reportParameters) {
                        JFreeChart chart = ChartFactory.createBoxAndWhiskerChart("Box and Whisker chart", "Category", "Value", dataset, true);
                        return new DrawChartRendererImpl(chart, null);
                }
        }

}
```

1.18.8 Group Chart




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

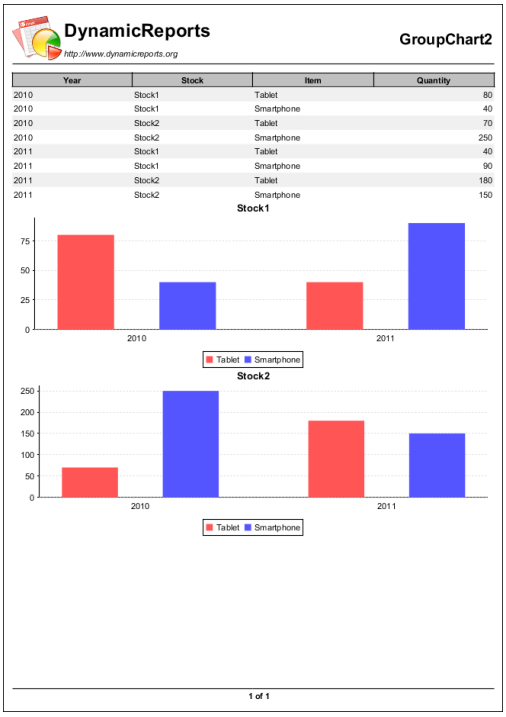
```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.chart.Bar3DChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupChartReport {

        public GroupChartReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> salesColumn = col.column("Sales", "sales
↪", type.bigDecimalType());

                Bar3DChartBuilder chart = cht.bar3DChart()
                                .setFixedHeight(180)
                                .setCategory(itemColumn)
                                .series(cht.serie(quantityColumn), cht.
↪serie(salesColumn))
                                .setCategoryAxisFormat(
                                                cht.axisFormat().setLabel("Item"));

                ColumnGroupBuilder countryGroup = grp.group(countryColumn)
                                .footer(chart);

                try {
```

(continues on next page)

(continued from previous page)

```
63                          report()
64                                          .setTemplate(Templates.reportTemplate)
65                                          .columns(countryColumn, itemColumn,␣
↪quantityColumn, salesColumn)
66                                          .title(Templates.createTitleComponent(
↪"GroupChart"))
67                                          .groupBy(countryGroup)
68                                          .summary(
69                                                          cmp.text("All countries").
↪setStyle(Templates.bold12CenteredStyle),
70                                                          chart)
71                                          .pageFooter(Templates.footerComponent)
72                                          .setDataSource(createDataSource())
73                                          .show();
74                  } catch (DRException e) {
75                          e.printStackTrace();
76                  }
77          }
78
79          private JRDataSource createDataSource() {
80                  DRDataSource dataSource = new DRDataSource("country", "item",
↪"quantity", "sales");
81                  dataSource.add("USA", "Tablet", 170, new BigDecimal(100));
82                  dataSource.add("USA", "Laptop", 90, new BigDecimal(280));
83                  dataSource.add("USA", "Smartphone", 120, new BigDecimal(250));
84                  dataSource.add("Canada", "Tablet", 120, new BigDecimal(80));
85                  dataSource.add("Canada", "Laptop", 150, new BigDecimal(310));
86                  dataSource.add("Canada", "Smartphone", 100, new BigDecimal(180));
87                  return dataSource;
88          }
89
90          public static void main(String[] args) {
91                  new GroupChartReport();
92          }
93  }
```

1.18.9 Group Chart 2




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.List;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupChartReport2 {

        public GroupChartReport2() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> yearColumn = col.column("Year", "year", type.stringType());
                TextColumnBuilder<String> stockColumn = col.column("Stock", "stock", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());

                BarChartBuilder chart1 = cht.barChart()
                                .setTitle(new ChartTitleExpression(stockColumn))
                                .setTitleFont(boldFont)
                                .setCategory(yearColumn)
                                .series(
                                                cht.serie(quantityColumn).setSeries(itemColumn));

                ColumnGroupBuilder stockGroup = grp.group(stockColumn)
                                .setHeaderLayout(GroupHeaderLayout.EMPTY)
                                .footer(chart1);

                JasperReportBuilder subReport = report()
                                .sortBy(stockColumn)
                                .groupBy(stockGroup)
                                .setDataSource(createDataSource());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(yearColumn, stockColumn, itemColumn, quantityColumn)
                                        .title(Templates.createTitleComponent("GroupChart2"))
                                        .summary(cmp.subreport(subReport))
```

(continues on next page)

(continued from previous page)

```
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("year", "stock", "item",
↪"quantity");
                dataSource.add("2010", "Stock1", "Tablet", 80);
                dataSource.add("2010", "Stock1", "Smartphone", 40);
                dataSource.add("2010", "Stock2", "Tablet", 70);
                dataSource.add("2010", "Stock2", "Smartphone", 250);
                dataSource.add("2011", "Stock1", "Tablet", 40);
                dataSource.add("2011", "Stock1", "Smartphone", 90);
                dataSource.add("2011", "Stock2", "Tablet", 180);
                dataSource.add("2011", "Stock2", "Smartphone", 150);
                return dataSource;
        }

        public static void main(String[] args) {
                new GroupChartReport2();
        }

        private class ChartTitleExpression extends AbstractComplexExpression<String> {
                private static final long serialVersionUID = 1L;

                public ChartTitleExpression(TextColumnBuilder<String> stockColumn) {
                        addExpression(stockColumn);
                }

                @Override
                public String evaluate(List<?> values, ReportParameters␣
↪reportParameters) {
                        return (String) values.get(0);
                }
        }
}
```

1.18.10 Group Count Chart


```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.VariableBuilder;
import net.sf.dynamicreports.report.builder.chart.Bar3DChartBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.constant.Evaluation;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupCountChartReport {

        public GroupCountChartReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());

                Bar3DChartBuilder chart1 = cht.bar3DChart()
                                .setFixedHeight(180)
                                .setCategory(countryColumn)
                                .series(cht.serie(exp.number(1)).setLabel("Items␣
↪(group count)"))
                                .setCategoryAxisFormat(
                                                cht.axisFormat().setLabel("Country"));

                VariableBuilder<Integer> itemVariable = variable(itemColumn,␣
↪Calculation.DISTINCT_COUNT);
                itemVariable.setResetType(Evaluation.FIRST_GROUP);

                Bar3DChartBuilder chart2 = cht.bar3DChart()
                                .setFixedHeight(180)
                                .setCategory(countryColumn)
                                .series(cht.serie(itemVariable).setLabel("Items␣
↪(group distinct count)"))
                                .setCategoryAxisFormat(
                                                cht.axisFormat().setLabel("Country"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(countryColumn, itemColumn,␣
↪quantityColumn)
                                        .title(Templates.createTitleComponent(
↪"GroupCountChart"))
                                        .groupBy(grp.group(countryColumn))
                                        .summary(cmp.horizontalList(chart1, chart2))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
```

(continues on next page)

(continued from previous page)

```
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("country", "item",
↪"quantity");
                dataSource.add("USA", "Tablet", 170);
                dataSource.add("USA", "Tablet", 80);
                dataSource.add("USA", "Laptop", 90);
                dataSource.add("USA", "Smartphone", 120);
                dataSource.add("Canada", "Tablet", 120);
                dataSource.add("Canada", "Laptop", 150);
                return dataSource;
        }

        public static void main(String[] args) {
                new GroupCountChartReport();
        }
}
```

1.18.11 Series Bar Chart


```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

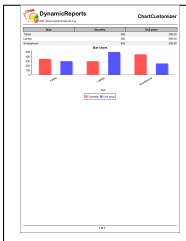
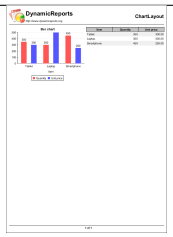
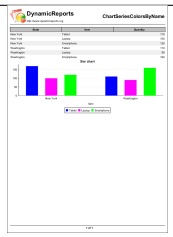
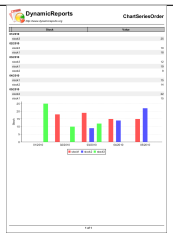
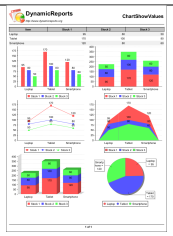
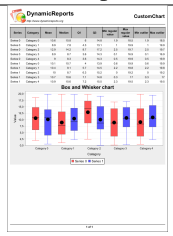
```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.chartcustomization;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SeriesBarChartReport {

        public SeriesBarChartReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                TextColumnBuilder<String> stateColumn = col.column("State", "state",␣
↪type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(stateColumn, itemColumn,␣
↪quantityColumn)
                                        .title(Templates.createTitleComponent(
↪"SeriesBarChart"))
                                        .summary(
                                                        cht.barChart()
                                                                        .setTitle(
↪"Bar chart")
                                                                        .
↪setTitleFont(boldFont)
                                                                        .
↪setCategory(stateColumn)
```

(continues on next page)

(continued from previous page)

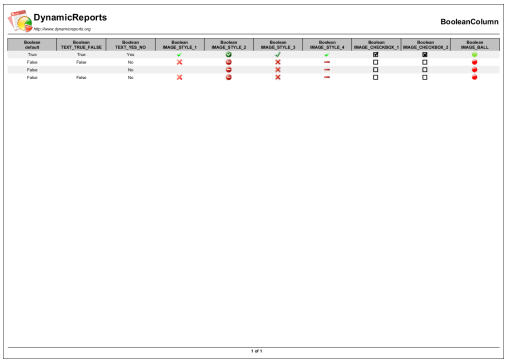
```
                                                                        .series(
                                                                                                                cht.
↪serie(quantityColumn).setSeries(itemColumn))
                                                                        .
↪setCategoryAxisFormat(
                                                                                                                cht.
↪axisFormat().setLabel("Item")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪");
                dataSource.add("New York", "Tablet", 170);
                dataSource.add("New York", "Laptop", 100);
                dataSource.add("New York", "Smartphone", 120);
                dataSource.add("Washington", "Tablet", 110);
                dataSource.add("Washington", "Laptop", 90);
                dataSource.add("Washington", "Smartphone", 160);
                return dataSource;
        }

        public static void main(String[] args) {
                new SeriesBarChartReport();
        }
}
```

Table 3: Chart customization Examples

 ChartCustomizerReport	 ChartLayoutReport	 ChartSeriesColorsByNameReport	 ChartSeriesOrderReport
 ChartShowPercentagesReport	 ChartShowValuesReport	 CustomChartReport	 GroupChartReport
 GroupChartReport2	 GroupCountChartReport	 SeriesBarChartReport	

1.19 Column

1.19.1 Boolean Column



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

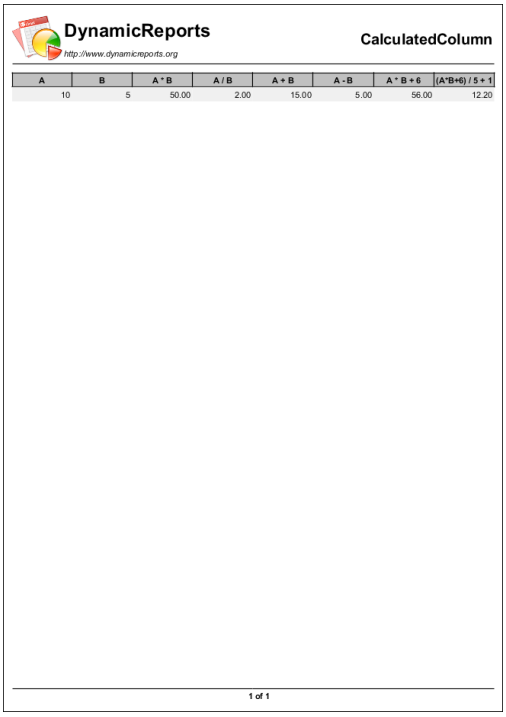
```
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.constant.BooleanComponentType;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class BooleanColumnReport {

        public BooleanColumnReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setPageFormat(PageType.A3, PageOrientation.
↪LANDSCAPE)
                                        .columns(
                                                        col.booleanColumn(
↪"Boolean\ndefault", "boolean"),
                                                        col.booleanColumn(
↪"Boolean\nTEXT_TRUE_FALSE", "boolean").setComponentType(BooleanComponentType.TEXT_
↪TRUE_FALSE).setEmptyWhenNullValue(true),
                                                        col.booleanColumn(
↪"Boolean\nTEXT_YES_NO", "boolean").setComponentType(BooleanComponentType.TEXT_YES_
↪NO),
                                                        col.booleanColumn(
↪"Boolean\nIMAGE_STYLE_1", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪STYLE_1).setEmptyWhenNullValue(true),
                                                        col.booleanColumn(
↪"Boolean\nIMAGE_STYLE_2", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪STYLE_2),
```

(continues on next page)

(continued from previous page)

```
54                                                          col.booleanColumn(
↪"Boolean\nIMAGE_STYLE_3", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪STYLE_3),
55                                                          col.booleanColumn(
↪"Boolean\nIMAGE_STYLE_4", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪STYLE_4),
56                                                          col.booleanColumn(
↪"Boolean\nIMAGE_CHECKBOX_1", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪CHECKBOX_1),
57                                                          col.booleanColumn(
↪"Boolean\nIMAGE_CHECKBOX_2", "boolean").setComponentType(BooleanComponentType.IMAGE_
↪CHECKBOX_2),
58                                                          col.booleanColumn(
↪"Boolean\nIMAGE_BALL", "boolean").setComponentType(BooleanComponentType.IMAGE_BALL))
59                                              .title(Templates.createTitleComponent(
↪"BooleanColumn"))
60                                              .pageFooter(Templates.footerComponent)
61                                              .setDataSource(createDataSource())
62                                              .show();
63                  } catch (DREException e) {
64                          e.printStackTrace();
65                  }
66          }
67
68          private JRDataSource createDataSource() {
69                  DRDataSource dataSource = new DRDataSource("boolean");
70                  dataSource.add(true);
71                  dataSource.add(false);
72                  dataSource.add();
73                  dataSource.add(false);
74                  return dataSource;
75          }
76
77          public static void main(String[] args) {
78                  new BooleanColumnReport();
79          }
80  }
```

1.19.2 Calculated Column



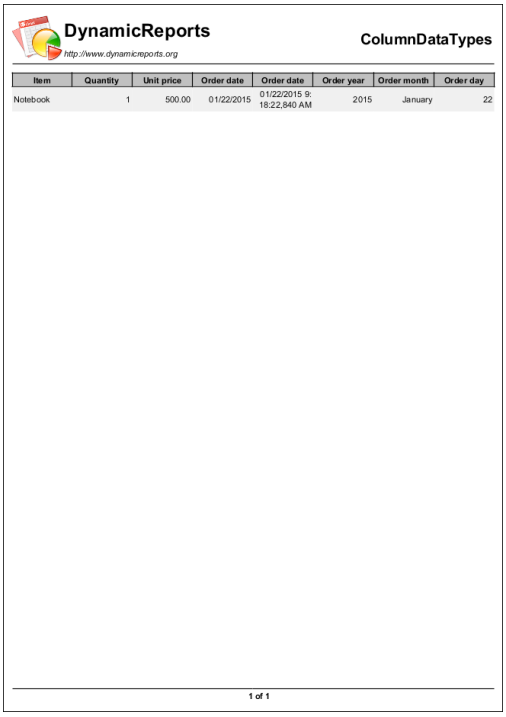

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CalculatedColumnReport {

        public CalculatedColumnReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<Integer> column1 = col.column("A", "field1", type.integerType());
                TextColumnBuilder<Integer> column2 = col.column("B", "field2", type.integerType());
                TextColumnBuilder<BigDecimal> column3 = column1.multiply(column2).setTitle("A * B");
                TextColumnBuilder<BigDecimal> column4 = column1.divide(2, column2).setTitle("A / B");
                TextColumnBuilder<BigDecimal> column5 = column1.add(column2).setTitle("A + B");
                TextColumnBuilder<BigDecimal> column6 = column1.subtract(column2).setTitle("A - B");
                TextColumnBuilder<BigDecimal> column7 = column3.add(6).setTitle("A * B + 6");
                TextColumnBuilder<BigDecimal> column8 = column7.divide(2, 5).add(1).setTitle("(A*B+6) / 5 + 1");

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        column1, column2, column3, column4, column5, column6, column7, column8)
                                        .title(Templates.createTitleComponent("CalculatedColumn"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("field1", "field2");
                dataSource.add(10, 5);
                return dataSource;
        }

        public static void main(String[] args) {
                new CalculatedColumnReport();
        }
}
```

1.19.3 Column Data Types



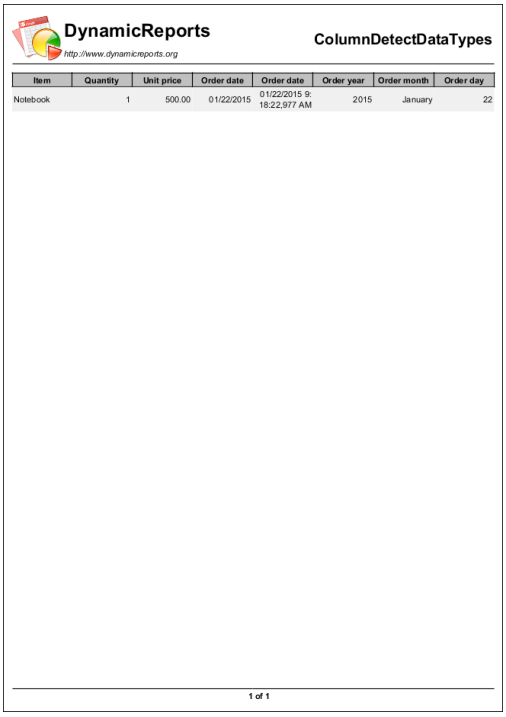

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnDataTypesReport {

        public ColumnDataTypesReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()),
                                                        col.column("Order date", "orderdate", type.dateType()),
                                                        col.column("Order date", "orderdate", type.dateYearToFractionType()),
                                                        col.column("Order year", "orderdate", type.dateYearType()),
                                                        col.column("Order month", "orderdate", type.dateMonthType()),
                                                        col.column("Order day", "orderdate", type.dateDayType()))
                                        .title(Templates.createTitleComponent("ColumnDataTypes"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate", "quantity", "unitprice");
                dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
                return dataSource;
        }

        public static void main(String[] args) {
                new ColumnDataTypesReport();
        }
}
```

1.19.4 Column Detect Data Type




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

(continues on next page)

(continued from previous page)

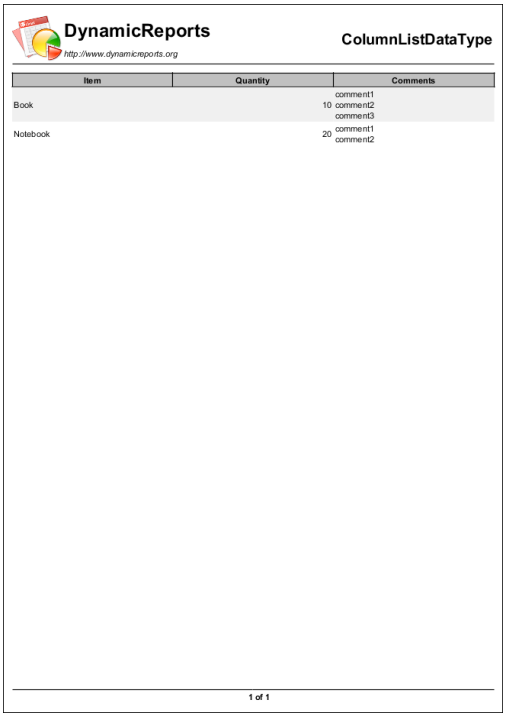
```
31
32  import net.sf.dynamicreports.examples.Templates;
33  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
34  import net.sf.dynamicreports.report.datasource.DRDataSource;
35  import net.sf.dynamicreports.report.definition.datatype.DRIDataType;
36  import net.sf.dynamicreports.report.exception.DRException;
37  import net.sf.jasperreports.engine.JRDataSource;
38
39  /**
40   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41   */
42  public class ColumnDetectDataTypeReport {
43
44          public ColumnDetectDataTypeReport() {
45                  build();
46          }
47
48          @SuppressWarnings({ "rawtypes", "unchecked" })
49          private void build() {
50                  try {
51                          JasperReportBuilder report = report();
52                          for (Column column : createColumns()) {
53                                  report.addColumn(col.column(column.title, column.
    ↪field, (DRIDataType) type.detectType(column.dataType)));
54                          }
55                          report
56                                          .setTemplate(Templates.reportTemplate)
57                                          .title(Templates.createTitleComponent(
    ↪"ColumnDetectDataTypes"))
58                                          .pageFooter(Templates.footerComponent)
59                                          .setDataSource(createDataSource())
60                                          .show();
61                  } catch (DRException e) {
62                          e.printStackTrace();
63                  }
64          }
65
66          private JRDataSource createDataSource() {
67                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪"quantity", "unitprice");
68                  dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
69                  return dataSource;
70          }
71
72          private List<Column> createColumns() {
73                  List<Column> columns = new ArrayList<Column>();
74                  columns.add(new Column("Item", "item", "string"));// dataType =
    ↪"String", "STRING", "java.lang.String", "text"
75                  columns.add(new Column("Quantity", "quantity", "integer"));//␣
    ↪dataType = "Integer", "INTEGER", "java.lang.Integer"
76                  columns.add(new Column("Unit price", "unitprice", "bigDecimal"));//␣
    ↪dataType = "bigdecimal", "BIGDECIMAL", "java.math.BigDecimal"
77                  columns.add(new Column("Order date", "orderdate", "date"));//␣
    ↪dataType = "Date", "DATE", "java.util.Date"
78                  columns.add(new Column("Order date", "orderdate", "dateYearToFraction
    ↪"));// dataType = "dateyeartofraction", "DATEYEARTOFRACTION"
79                  columns.add(new Column("Order year", "orderdate", "dateYear"));//␣
    ↪dataType = "DateYear", "dateyear", "DATEYEAR"
```

(continues on next page)

(continued from previous page)

```
80                  columns.add(new Column("Order month", "orderdate", "dateMonth"));//␣
↪dataType = "DateMonth", "datemonth", "DATEMONTH"
81                  columns.add(new Column("Order day", "orderdate", "dateDay"));//␣
↪dataType = "DateDay", "dateday", "DATEDAY"
82                  return columns;
83          }
84
85          private class Column {
86                  private String title;
87                  private String field;
88                  private String dataType;
89
90                  private Column(String title, String field, String dataType) {
91                          this.title = title;
92                          this.field = field;
93                          this.dataType = dataType;
94                  }
95          }
96
97          public static void main(String[] args) {
98                  new ColumnDetectDataTypeReport();
99          }
100 }
```

1.19.5 Column List Data Type




```
1   /**
2    * DynamicReports - Free Java reporting library for creating reports dynamically
3    *
```

(continues on next page)

(continued from previous page)

```
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.ArrayList;
import java.util.List;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnListDataTypeReport {

        public ColumnListDataTypeReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Comments", "comments", type.listType()))
                                        .title(Templates.createTitleComponent("ColumnListDataType"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
```

(continues on next page)

(continued from previous page)

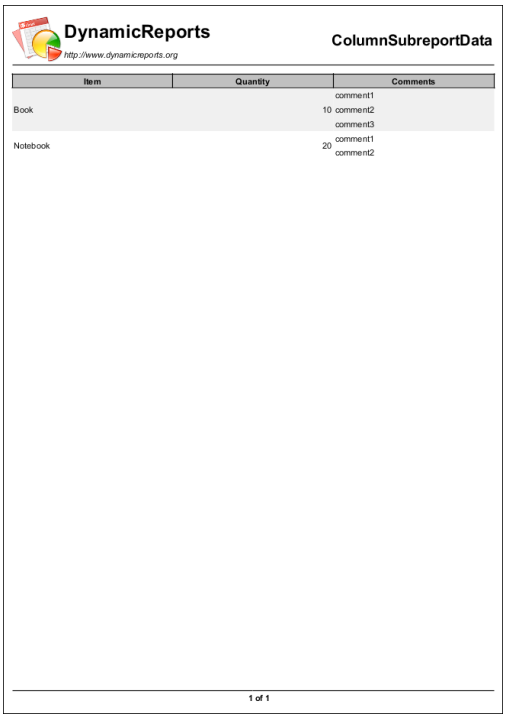
```
                e.printStackTrace();
            }
        }

        private JRDataSource createDataSource() {
            List<ReportData> datasource = new ArrayList<ReportData>();

            ReportData data = new ReportData();
            List<String> comments = new ArrayList<String>();
            comments.add("comment1");
            comments.add("comment2");
            comments.add("comment3");
            data.setItem("Book");
            data.setQuantity(new Integer(10));
            data.setComments(comments);
            datasource.add(data);

            data = new ReportData();
            comments = new ArrayList<String>();
            comments.add("comment1");
            comments.add("comment2");
            data.setItem("Notebook");
            data.setQuantity(new Integer(20));
            data.setComments(comments);
            datasource.add(data);

            return new JRBeanCollectionDataSource(datasource);
        }

        public class ReportData {
            private String item;
            private Integer quantity;
            private List<String> comments;

            public String getItem() {
                return item;
            }

            public void setItem(String item) {
                this.item = item;
            }

            public Integer getQuantity() {
                return quantity;
            }

            public void setQuantity(Integer quantity) {
                this.quantity = quantity;
            }

            public List<String> getComments() {
                return comments;
            }

            public void setComments(List<String> comments) {
                this.comments = comments;
            }
```

(continues on next page)

(continued from previous page)

```
114         }
115
116         public static void main(String[] args) {
117                 new ColumnListDataTypeReport();
118         }
119 }
```

1.19.6 Column Subreport Data




```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
```

(continues on next page)

(continued from previous page)

```
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import net.sf.jasperreports.engine.data.JRMapCollectionDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnSubreportDataReport {

        public ColumnSubreportDataReport() {
                build();
        }

        private void build() {
                SubreportBuilder subreport = cmp.subreport(new SubreportDesign())
                                .setDataSource(new SubreportData());
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(field("comments", List.class))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.componentColumn("Comments
↪", subreport))
                                        .title(Templates.createTitleComponent(
↪"ColumnSubreportData"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class SubreportDesign extends AbstractSimpleExpression
↪<JasperReportBuilder> {
```

(continues on next page)

(continued from previous page)

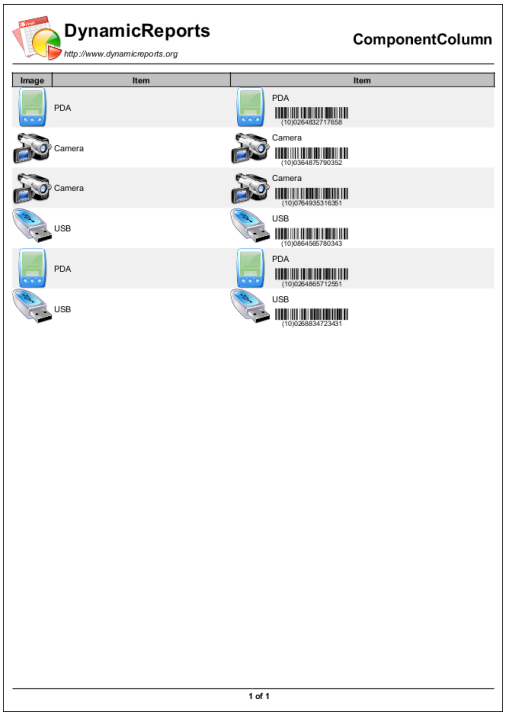
```
73                 private static final long serialVersionUID = 1L;
74
75                 @Override
76                 public JasperReportBuilder evaluate(ReportParameters␣
↪reportParameters) {
77                         JasperReportBuilder report = report()
78                                         .columns(col.column("comment", type.
↪stringType()));
79                         return report;
80                 }
81         }
82
83         private class SubreportData extends AbstractSimpleExpression<JRDataSource> {
84                 private static final long serialVersionUID = 1L;
85
86                 @Override
87                 public JRDataSource evaluate(ReportParameters reportParameters) {
88                         Collection<Map<String, ?>> value = reportParameters.getValue(
↪"comments");
89                         return new JRMapCollectionDataSource(value);
90                 }
91         }
92
93         private JRDataSource createDataSource() {
94                 List<ReportData> datasource = new ArrayList<ReportData>();
95
96                 ReportData data = new ReportData();
97                 List<Map<String, Object>> comments = new ArrayList<Map<String, Object>
↪>();
98                 Map<String, Object> values = new HashMap<String, Object>();
99                 values.put("comment", "comment1");
100                comments.add(values);
101                values = new HashMap<String, Object>();
102                values.put("comment", "comment2");
103                comments.add(values);
104                values = new HashMap<String, Object>();
105                values.put("comment", "comment3");
106                comments.add(values);
107                data.setItem("Book");
108                data.setQuantity(new Integer(10));
109                data.setComments(comments);
110                datasource.add(data);
111
112                data = new ReportData();
113                comments = new ArrayList<Map<String, Object>>();
114                values = new HashMap<String, Object>();
115                values.put("comment", "comment1");
116                comments.add(values);
117                values = new HashMap<String, Object>();
118                values.put("comment", "comment2");
119                comments.add(values);
120                data.setItem("Notebook");
121                data.setQuantity(new Integer(20));
122                data.setComments(comments);
123                datasource.add(data);
124
125                return new JRBeanCollectionDataSource(datasource);
```

(continues on next page)

(continued from previous page)

```
126         }
127
128         public class ReportData {
129                 private String item;
130                 private Integer quantity;
131                 private List<Map<String, Object>> comments;
132
133                 public String getItem() {
134                         return item;
135                 }
136
137                 public void setItem(String item) {
138                         this.item = item;
139                 }
140
141                 public Integer getQuantity() {
142                         return quantity;
143                 }
144
145                 public void setQuantity(Integer quantity) {
146                         this.quantity = quantity;
147                 }
148
149                 public List<Map<String, Object>> getComments() {
150                         return comments;
151                 }
152
153                 public void setComments(List<Map<String, Object>> comments) {
154                         this.comments = comments;
155                 }
156         }
157
158         public static void main(String[] args) {
159                 new ColumnSubreportDataReport();
160         }
161 }
```

1.19.7 Component Column




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
```

(continues on next page)

(continued from previous page)

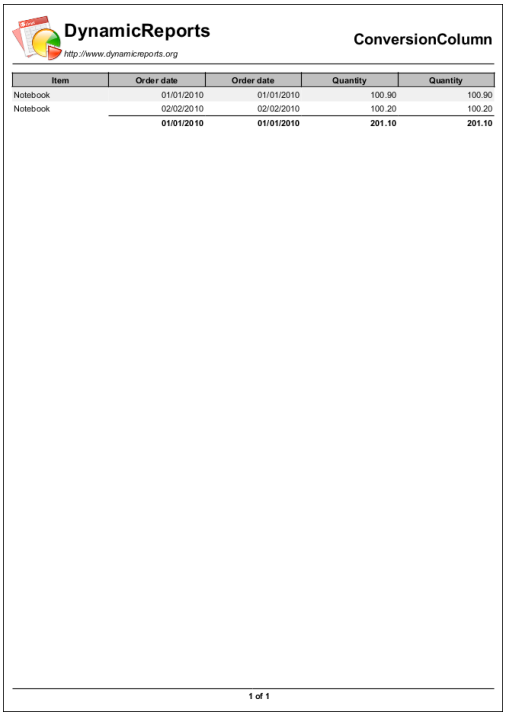
```
import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
import net.sf.dynamicreports.report.builder.component.ImageBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ComponentColumnReport {

        public ComponentColumnReport() {
                build();
        }

        private void build() {
                try {
                        ImageBuilder image = cmp.image(new ImageExpression()).
↪setFixedDimension(48, 48);
                        HorizontalListBuilder itemComponent = cmp.horizontalList(
                                        image,
                                        cmp.verticalList(
                                                        cmp.text(new␣
↪ItemExpression()),
                                                        bcode.ean128(new␣
↪BarcodeExpression()).setFixedHeight(24)));

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("image", String.class),
                                                        field("barcode", String.
↪class))
                                        .columns(
                                                        col.componentColumn("Image",␣
↪image),
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.componentColumn("Item",␣
↪itemComponent))
                                        .title(Templates.createTitleComponent(
↪"ComponentColumn"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        public class ImageExpression extends AbstractSimpleExpression<InputStream> {
                private static final long serialVersionUID = 1L;

                @Override
                public InputStream evaluate(ReportParameters reportParameters) {
                        return Templates.class.getResourceAsStream("images/" +␣
↪reportParameters.getValue("image") + ".png");
```

(continues on next page)

(continued from previous page)

```
                }
        }

        public class ItemExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return reportParameters.getValue("item");
                }
        }

        public class BarcodeExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return reportParameters.getValue("barcode");
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "image", "barcode
↪");
                dataSource.add("PDA", "pda", "100264832717658");
                dataSource.add("Camera", "camera", "100364875790352");
                dataSource.add("Camera", "camera", "100764935316351");
                dataSource.add("USB", "usb", "100864565780343");
                dataSource.add("PDA", "pda", "100264865712551");
                dataSource.add("USB", "usb", "100268834723431");
                return dataSource;
        }

        public static void main(String[] args) {
                new ComponentColumnReport();
        }
}
```

1.19.8 Conversion Column




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
import java.text.ParseException;
```

(continues on next page)

(continued from previous page)

```
import java.text.SimpleDateFormat;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ConversionColumnReport {

        public ConversionColumnReport() {
                build();
        }

        private void build() {
                try {
                        TextColumnBuilder<Date> orderDateColumn1 = col.column("Order␣
↪date", new OrderDateColumn1())
                                        .setDataType(type.dateType());
                        TextColumnBuilder<Date> orderDateColumn2 = col.column("Order␣
↪date", new OrderDateColumn2())
                                        .setDataType(type.dateType());
                        TextColumnBuilder<BigDecimal> quantityColumn1 = col.column(
↪"Quantity", new QuantityColumn1())
                                        .setDataType(type.bigDecimalType());
                        TextColumnBuilder<BigDecimal> quantityColumn2 = col.column(
↪"Quantity", new QuantityColumn2())
                                        .setDataType(type.bigDecimalType());

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("orderdate", String.
↪class),
                                                        field("quantity", String.
↪class))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        orderDateColumn1,␣
↪orderDateColumn2, quantityColumn1, quantityColumn2)
                                        .title(Templates.createTitleComponent(
↪"ConversionColumn"))
                                        .pageFooter(Templates.footerComponent)
                                        .subtotalsAtSummary(
                                                        sbt.min(orderDateColumn1),␣
↪sbt.min(orderDateColumn2), sbt.sum(quantityColumn1), sbt.sum(quantityColumn2))
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
```

(continues on next page)

(continued from previous page)

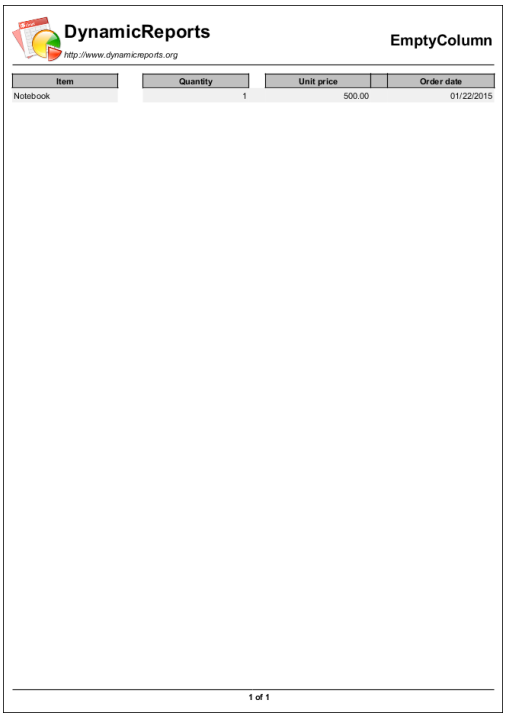
```
78                  }
79          }
80
81          private JRDataSource createDataSource() {
82                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪"quantity");
83                  dataSource.add("Notebook", "1/1/2010", "100.9");
84                  dataSource.add("Notebook", "2/2/2010", "100.2");
85                  return dataSource;
86          }
87
88          public static void main(String[] args) {
89                  new ConversionColumnReport();
90          }
91
92          private class OrderDateColumn1 extends AbstractSimpleExpression<Date> {
93                  private static final long serialVersionUID = 1L;
94
95                  @Override
96                  public Date evaluate(ReportParameters reportParameters) {
97                          String value = reportParameters.getValue("orderdate");
98                          try {
99                                  return new SimpleDateFormat("MM/dd/yyyy",␣
    ↪reportParameters.getLocale()).parse(value);
100                         } catch (ParseException e) {
101                                 e.printStackTrace();
102                         }
103                         return null;
104                 }
105         }
106
107         private class OrderDateColumn2 extends AbstractSimpleExpression<Date> {
108                 private static final long serialVersionUID = 1L;
109
110                 @Override
111                 public Date evaluate(ReportParameters reportParameters) {
112                         try {
113                                 return type.dateType().stringToValue("orderdate",␣
    ↪reportParameters);
114                         } catch (DRException e) {
115                                 e.printStackTrace();
116                         }
117                         return null;
118                 }
119         }
120
121         private class QuantityColumn1 extends AbstractSimpleExpression<BigDecimal> {
122                 private static final long serialVersionUID = 1L;
123
124                 @Override
125                 public BigDecimal evaluate(ReportParameters reportParameters) {
126                         String value = reportParameters.getValue("quantity");
127                         try {
128                                 Number number = new DecimalFormat("#,###.##", new␣
    ↪DecimalFormatSymbols(reportParameters.getLocale())).parse(value);
129                                 return new BigDecimal(number.doubleValue());
130                         } catch (ParseException e) {
```

(continues on next page)

(continued from previous page)

```
                                e.printStackTrace();
                        }
                        return null;
                }
        }

        private class QuantityColumn2 extends AbstractSimpleExpression<BigDecimal> {
                private static final long serialVersionUID = 1L;

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        try {
                                return type.bigDecimalType().stringToValue("quantity",
↪ reportParameters);
                        } catch (DRException e) {
                                e.printStackTrace();
                        }
                        return null;
                }
        }
}
```

1.19.9 Empty Column




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
```

(continues on next page)

(continued from previous page)

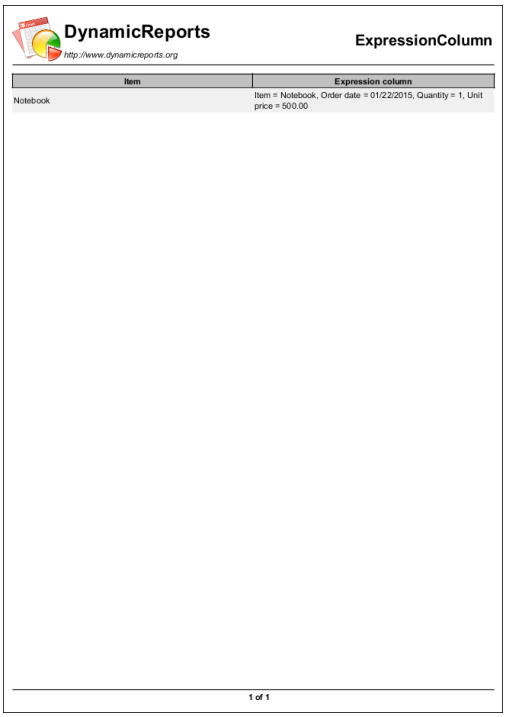
```
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class EmptyColumnReport {

        public EmptyColumnReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.emptyColumn().setFixedWidth(30),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.emptyColumn(false, true).setFixedWidth(20),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()),
                                                        col.emptyColumn(true, true).setFixedWidth(20),
                                                        col.column("Order date", "orderdate", type.dateType()))
```

(continues on next page)

(continued from previous page)

```
                                        .title(Templates.createTitleComponent(
↪"EmptyColumn"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
                return dataSource;
        }

        public static void main(String[] args) {
                new EmptyColumnReport();
        }
}
```

1.19.10 Expression Column




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
```

(continues on next page)

(continued from previous page)

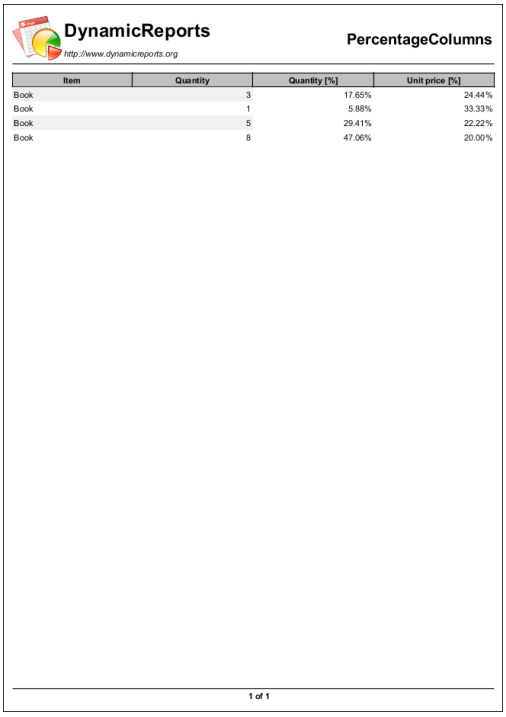
```
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ExpressionColumnReport {
        private FieldBuilder<Integer> quantityField;

        public ExpressionColumnReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("orderdate", Date.
↪class),
                                                        quantityField = field(
↪"quantity", Integer.class),
                                                        field("unitprice", BigDecimal.
↪class))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Expression column
↪", new ExpressionColumn()))
```

(continues on next page)

(continued from previous page)

```
59                                          .title(Templates.createTitleComponent(
↪"ExpressionColumn"))
60                                          .pageFooter(Templates.footerComponent)
61                                          .setDataSource(createDataSource())
62                                          .show();
63                  } catch (DRException e) {
64                          e.printStackTrace();
65                  }
66          }
67
68          private JRDataSource createDataSource() {
69                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
70                  dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
71                  return dataSource;
72          }
73
74          public static void main(String[] args) {
75                  new ExpressionColumnReport();
76          }
77
78          private class ExpressionColumn extends AbstractSimpleExpression<String> {
79                  private static final long serialVersionUID = 1L;
80
81                  @Override
82                  public String evaluate(ReportParameters reportParameters) {
83                          return "Item = " + reportParameters.getValue("item") + ", " +
84                                          "Order date = " + type.dateType().
↪valueToString("orderdate", reportParameters) + ", " +
85                                          "Quantity = " + type.integerType().
↪valueToString(quantityField, reportParameters) + ", " +
86                                          "Unit price = " + type.bigDecimalType().
↪valueToString("unitprice", reportParameters);
87                  }
88          }
89  }
```

1.19.11 Percentage Columns




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
```

(continues on next page)

(continued from previous page)

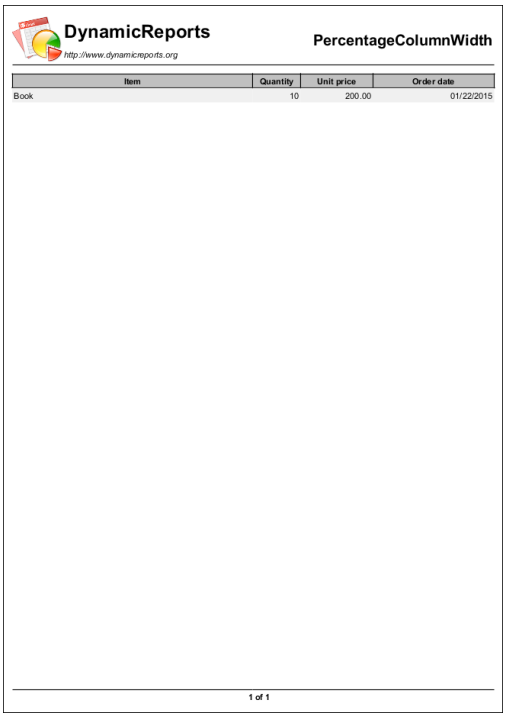
```
import net.sf.dynamicreports.report.builder.column.PercentageColumnBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PercentageColumnsReport {
        private FieldBuilder<BigDecimal> unitPriceField;

        public PercentageColumnsReport() {
                build();
        }

        private void build() {
                try {
                        unitPriceField = field("unitprice", BigDecimal.class);

                        TextColumnBuilder<String> itemColumn = col.column("Item",
↪"item", type.stringType());
                        TextColumnBuilder<Integer> quantityColumn = col.column(
↪"Quantity", "quantity", type.integerType());
                        PercentageColumnBuilder quantityPercColumn = col.
↪percentageColumn("Quantity [%]", quantityColumn);
                        PercentageColumnBuilder unitPricePercColumn = col.
↪percentageColumn("Unit price [%]", unitPriceField);

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        unitPriceField)
                                        .columns(
                                                        itemColumn, quantityColumn,␣
↪quantityPercColumn, unitPricePercColumn)
                                        .title(Templates.createTitleComponent(
↪"PercentageColumns"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Book", 3, new BigDecimal(11));
                dataSource.add("Book", 1, new BigDecimal(15));
                dataSource.add("Book", 5, new BigDecimal(10));
                dataSource.add("Book", 8, new BigDecimal(9));
                return dataSource;
        }

        public static void main(String[] args) {
```

(continues on next page)

(continued from previous page)

```
                new PercentageColumnsReport();
        }
}
```

1.19.12 Percentage Column Width



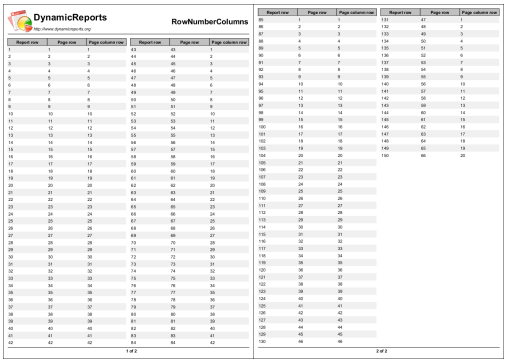

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;
```

(continues on next page)

(continued from previous page)

```
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.Date;
29
30  import net.sf.dynamicreports.examples.Templates;
31  import net.sf.dynamicreports.report.datasource.DRDataSource;
32  import net.sf.dynamicreports.report.exception.DRException;
33  import net.sf.jasperreports.engine.JRDataSource;
34
35  /**
36   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37   */
38  public class PercentageColumnWidthReport {
39
40          public PercentageColumnWidthReport() {
41                  build();
42          }
43
44          private void build() {
45                  try {
46                          report()
47                                          .setTemplate(Templates.reportTemplate)
48                                          .columns(
49                                                          col.column("Item", "item",␣
    ↪type.stringType()).setWidth(50), // 50%
50                                                          col.column("Quantity",
    ↪"quantity", type.integerType()).setWidth(10), // 10%
51                                                          col.column("Unit price",
    ↪"unitprice", type.bigDecimalType()).setWidth(15), // 15%
52                                                          col.column("Order date",
    ↪"orderdate", type.dateType()).setWidth(25)) // 25%
53                                          .title(Templates.createTitleComponent(
    ↪"PercentageColumnWidth"))
54                                          .pageFooter(Templates.footerComponent)
55                                          .setDataSource(createDataSource())
56                                          .show();
57                  } catch (DRException e) {
58                          e.printStackTrace();
59                  }
60          }
61
62          private JRDataSource createDataSource() {
63                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪"quantity", "unitprice");
64                  dataSource.add("Book", new Date(), 10, new BigDecimal(200));
65                  return dataSource;
66          }
67
68          public static void main(String[] args) {
69                  new PercentageColumnWidthReport();
70          }
71  }
```

1.19.13 Row Number Columns



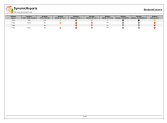
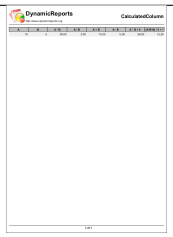
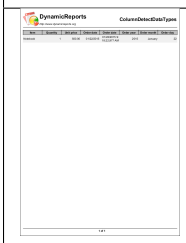
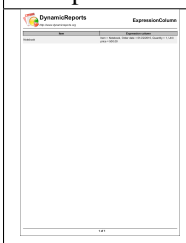
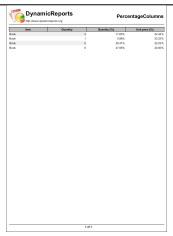
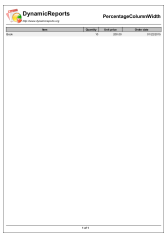
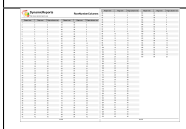

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.column;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JREmptyDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class RowNumberColumnsReport {

        public RowNumberColumnsReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
```

(continues on next page)

(continued from previous page)

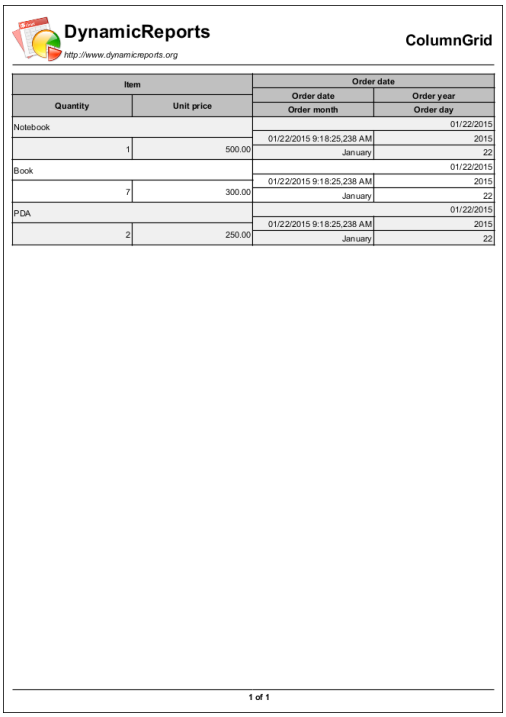
```
43                                          .setPageColumnsPerPage(2)
44                                          .setPageColumnSpace(10)
45                                          .columns(
46                                                  col.reportRowNumberColumn(
    ↪"Report row"),
47                                                  col.pageRowNumberColumn("Page␣
    ↪row"),
48                                                  col.columnRowNumberColumn(
    ↪"Page column row"))
49                                          .title(Templates.createTitleComponent(
    ↪"RowNumberColumns"))
50                                          .pageFooter(Templates.footerComponent)
51                                          .setDataSource(new JREmptyDataSource(150))
52                                          .show();
53                  } catch (DRException e) {
54                          e.printStackTrace();
55                  }
56          }
57
58          public static void main(String[] args) {
59                  new RowNumberColumnsReport();
60          }
61  }
```

Table 4: Column Examples

 BooleanColumnReport	 CalculatedColumnReport	 ColumnDataTypesReport
 ColumnDetectDataTypeReport	 ColumnListDataTypeReport	 ColumnSubreportDataReport
 ComponentColumnReport	 ConversionColumnReport	 EmptyColumnReport
 ExpressionColumnReport	 PercentageColumnsReport	 PercentageColumnWidthReport
 RowNumberColumnsReport		

1.20 Column Grid

1.20.1 Column Grid



DynamicReports
http://www.dynamicreports.org

ColumnGrid

Item		Order date	
Quantity	Unit price	Order date Order month	Order year Order day
Notebook			01/22/2015
1	500.00	01/22/2015 9:18:25,238 AM January	2015 22
Book			01/22/2015
7	300.00	01/22/2015 9:18:25,238 AM January	2015 22
PDA			01/22/2015
2	250.00	01/22/2015 9:18:25,238 AM January	2015 22

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
```

(continues on next page)

(continued from previous page)

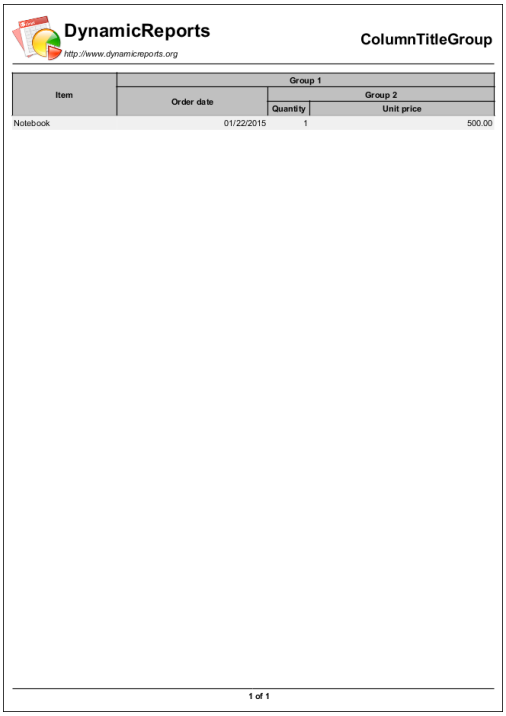
```
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnGridReport {

        public ColumnGridReport() {
                build();
        }

        private void build() {
                StyleBuilder textStyle = stl.style(Templates.columnStyle)
                                .setBorder(stl.pen1Point());

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());
                TextColumnBuilder<Date> orderDateColumn = col.column("Order date", "orderdate", type.dateType());
                TextColumnBuilder<Date> orderDateFColumn = col.column("Order date", "orderdate", type.dateYearToFractionType());
                TextColumnBuilder<Date> orderYearColumn = col.column("Order year", "orderdate", type.dateYearType());
                TextColumnBuilder<Date> orderMonthColumn = col.column("Order month", "orderdate", type.dateMonthType());
                TextColumnBuilder<Date> orderDayColumn = col.column("Order day", "orderdate", type.dateDayType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setColumnStyle(textStyle)
                                        .columns(
                                                        itemColumn, quantityColumn, unitPriceColumn, orderDateColumn,
                                                        orderDateFColumn, orderYearColumn, orderMonthColumn, orderDayColumn)
                                        .columnGrid(
                                                        grid.verticalColumnGridList(
                                                                        itemColumn,
                                                                        grid.horizontalColumnGridList(quantityColumn, unitPriceColumn)),
                                                        grid.verticalColumnGridList(
                                                                        orderDateColumn,
                                                                        grid.horizontalColumnGridList(orderDateFColumn, orderYearColumn),
```

(continues on next page)

(continued from previous page)

```
                                                                    grid.
↪horizontalColumnGridList(orderMonthColumn, orderDayColumn)))
                                        .title(Templates.createTitleComponent(
↪"ColumnGrid"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
                dataSource.add("Book", new Date(), 7, new BigDecimal(300));
                dataSource.add("PDA", new Date(), 2, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new ColumnGridReport();
        }
}
```

1.20.2 Column Title Group



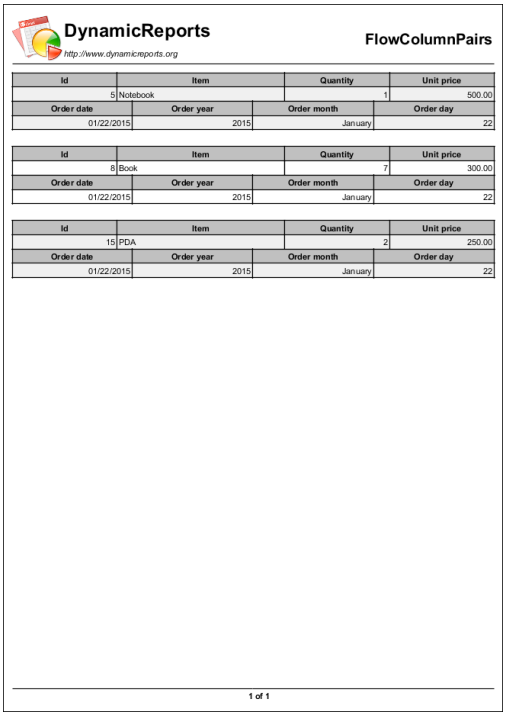
```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.grid.ColumnTitleGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnTitleGroupReport {

        public ColumnTitleGroupReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType()).setFixedWidth(50);
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                ColumnTitleGroupBuilder titleGroup2 = grid.titleGroup("Group 2",␣
↪quantityColumn, unitPriceColumn);
```

(continues on next page)

(continued from previous page)

```
53                 ColumnTitleGroupBuilder titleGroup1 = grid.titleGroup("Group 1",␣
↪orderDateColumn, titleGroup2).setTitleFixedWidth(450);
54
55                 try {
56                         report()
57                                         .setTemplate(Templates.reportTemplate)
58                                         .columnGrid(
59                                                         itemColumn, titleGroup1)
60                                         .columns(
61                                                         itemColumn, orderDateColumn,␣
↪quantityColumn, unitPriceColumn)
62                                         .title(Templates.createTitleComponent(
↪"ColumnTitleGroup"))
63                                         .pageFooter(Templates.footerComponent)
64                                         .setDataSource(createDataSource())
65                                         .show();
66                 } catch (DRException e) {
67                         e.printStackTrace();
68                 }
69         }
70
71         private JRDataSource createDataSource() {
72                 DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
73                 dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
74                 return dataSource;
75         }
76
77         public static void main(String[] args) {
78                 new ColumnTitleGroupReport();
79         }
80 }
```

1.20.3 Flow Column Pairs




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

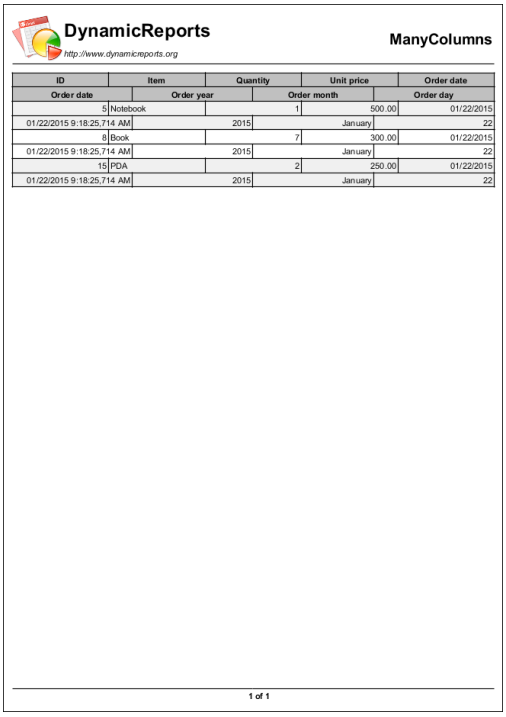
```
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.ListType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class FlowColumnPairsReport {

        public FlowColumnPairsReport() {
                build();
        }

        private void build() {
                StyleBuilder textStyle = stl.style(Templates.columnStyle)
                                .setBorder(stl.pen1Point());

                FieldBuilder<Integer> idField = field("id", type.integerType());
                FieldBuilder<String> itemField = field("item", type.stringType());
                FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
                FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());
                FieldBuilder<Date> orderDateField = field("orderdate", type.
↪dateType());
                FieldBuilder<Date> orderYearField = field("orderdate", type.
↪dateYearType());
                FieldBuilder<Date> orderMonthField = field("orderdate", type.
↪dateMonthType());
                FieldBuilder<Date> orderDayField = field("orderdate", type.
↪dateDayType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setColumnStyle(textStyle)
                                        .columnGrid(ListType.HORIZONTAL_FLOW)
                                        .fields(idField, itemField, quantityField,␣
↪unitPriceField, orderDateField, orderYearField, orderMonthField, orderDayField)
                                        .columns(
                                                        col.
↪componentColumn(columnPair("Id", idField)),
                                                        col.
↪componentColumn(columnPair("Item", itemField).setFixedWidth(200)),
                                                        col.
↪componentColumn(columnPair("Quantity", quantityField)),
                                                        col.
↪componentColumn(columnPair("Unit price", unitPriceField)),
                                                        col.
↪componentColumn(columnPair("Order date", orderDateField)),
                                                        col.
↪componentColumn(columnPair("Order year", orderYearField)),
```

(continues on next page)

(continued from previous page)

```
                                                        col.
↪componentColumn(columnPair("Order month", orderMonthField)),
                                                        col.
↪componentColumn(columnPair("Order day", orderDayField)))
                                        .title(Templates.createTitleComponent(
↪"FlowColumnPairs"))
                                        .detailFooter(cmp.verticalGap(20))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private VerticalListBuilder columnPair(String title, FieldBuilder<?> value) {
                TextFieldBuilder<String> titleCmp = cmp.text(title)
                                .setStyle(Templates.columnTitleStyle);
                TextFieldBuilder<?> valueCmp = cmp.text(value);
                return cmp.verticalList(titleCmp, valueCmp);
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("id", "item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));
                dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
                dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new FlowColumnPairsReport();
        }
}
```

1.20.4 Many Columns




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

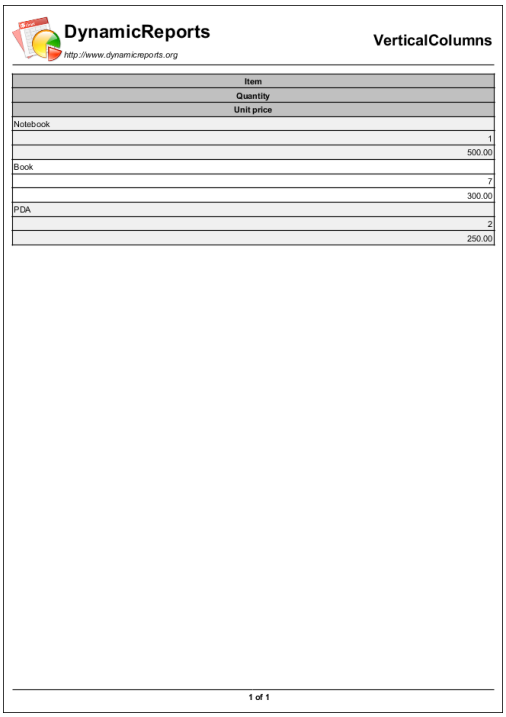
```
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.ListType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ManyColumnsReport {

        public ManyColumnsReport() {
                build();
        }

        private void build() {
                StyleBuilder textStyle = stl.style(Templates.columnStyle)
                                .setBorder(stl.pen1Point());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setColumnStyle(textStyle)
                                        .columnGrid(ListType.HORIZONTAL_FLOW)
                                        .columns(
                                                        col.column("ID", "id", type.integerType()),
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()),
                                                        col.column("Order date", "orderdate", type.dateType()),
                                                        col.column("Order date", "orderdate", type.dateYearToFractionType()),
                                                        col.column("Order year", "orderdate", type.dateYearType()),
                                                        col.column("Order month", "orderdate", type.dateMonthType()),
                                                        col.column("Order day", "orderdate", type.dateDayType()))
                                        .title(Templates.createTitleComponent("ManyColumns"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("id", "item", "orderdate", "quantity", "unitprice");
                dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
                dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new ManyColumnsReport();
        }
}
```

1.20.5 Vertical Columns




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
```

(continues on next page)

(continued from previous page)

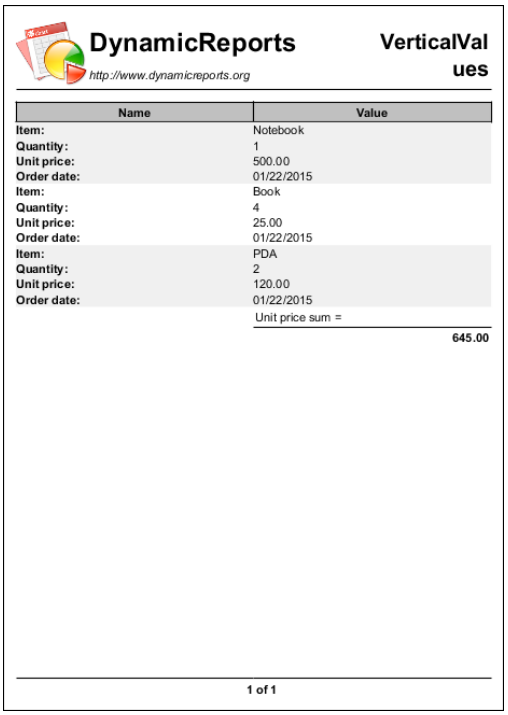
```
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.ListType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class VerticalColumnsReport {

        public VerticalColumnsReport() {
                build();
        }

        private void build() {
                StyleBuilder textStyle = stl.style(Templates.columnStyle)
                                .setBorder(stl.pen1Point());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setColumnStyle(textStyle)
                                        .columnGrid(ListType.VERTICAL)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()))
                                        .title(Templates.createTitleComponent("VerticalColumns"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                dataSource.add("Notebook", 1, new BigDecimal(500));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Book", 7, new BigDecimal(300));
                dataSource.add("PDA", 2, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new VerticalColumnsReport();
        }
}
```

1.20.6 Vertical Values




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
```

(continues on next page)

(continued from previous page)

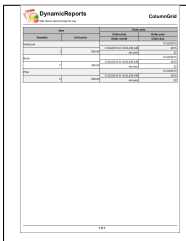
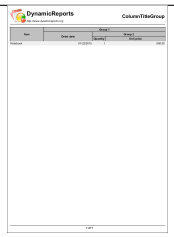
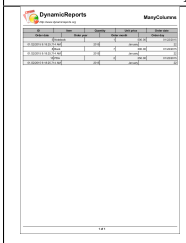
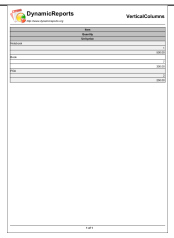
```
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.columngrid;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.column.ComponentColumnBuilder;
import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class VerticalValuesReport {

        public VerticalValuesReport() {
                build();
        }

        private void build() {
                StyleBuilder nameStyle = stl.style().bold();
                StyleBuilder valueStyle = stl.style().
↪setHorizontalTextAlignment(HorizontalTextAlignment.LEFT);

                FieldBuilder<String> itemField = field("item", type.stringType());
                FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
                FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());
                FieldBuilder<Date> orderDateField = field("orderdate", type.
↪dateType());

                VerticalListBuilder nameList = cmp.verticalList(
                                cmp.text("Item:").setStyle(nameStyle),
                                cmp.text("Quantity:").setStyle(nameStyle),
                                cmp.text("Unit price:").setStyle(nameStyle),
                                cmp.text("Order date:").setStyle(nameStyle));
                VerticalListBuilder valueList = cmp.verticalList(
                                cmp.text(itemField).setStyle(valueStyle),
                                cmp.text(quantityField).setStyle(valueStyle),
                                cmp.text(unitPriceField).setStyle(valueStyle),
                                cmp.text(orderDateField).setStyle(valueStyle));

```

(continues on next page)

(continued from previous page)

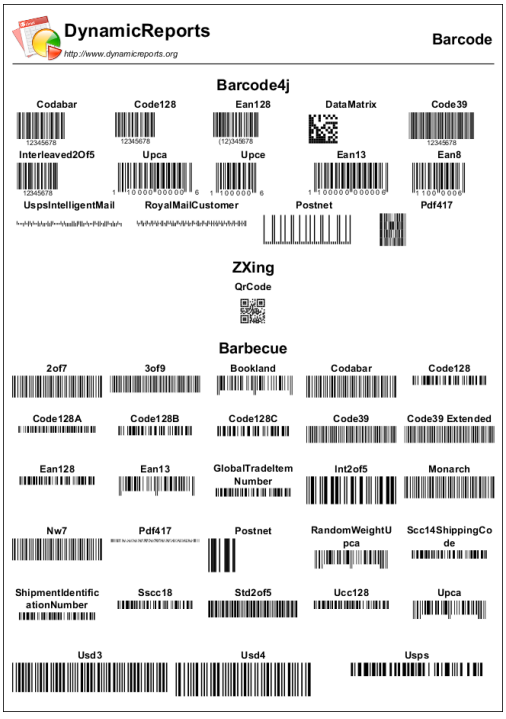
```
71                 ComponentColumnBuilder nameColumn = col.componentColumn("Name",␣
↪nameList);
72                 ComponentColumnBuilder valueColumn = col.componentColumn("Value",␣
↪valueList);
73
74                 AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.
↪sum(unitPriceField, valueColumn)
75                                 .setLabel("Unit price sum =");
76
77                 try {
78                         report()
79                                         .setTemplate(Templates.reportTemplate)
80                                         .setPageFormat(PageType.A5)
81                                         .fields(itemField, quantityField,␣
↪unitPriceField, orderDateField)
82                                         .columns(nameColumn, valueColumn)
83                                         .subtotalsAtSummary(unitPriceSum)
84                                         .title(Templates.createTitleComponent(
↪"VerticalValues"))
85                                         .pageFooter(Templates.footerComponent)
86                                         .setDataSource(createDataSource())
87                                         .show();
88                 } catch (DRException e) {
89                         e.printStackTrace();
90                 }
91         }
92
93         private JRDataSource createDataSource() {
94                 DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
95                 dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
96                 dataSource.add("Book", new Date(), 4, new BigDecimal(25));
97                 dataSource.add("PDA", new Date(), 2, new BigDecimal(120));
98                 return dataSource;
99         }
100
101        public static void main(String[] args) {
102                new VerticalValuesReport();
103        }
104 }
```

Table 5: Chart Examples

 ColumnGridReport	 ColumnTitleGroupReport	 FlowColumnPairsReport
 ManyColumnsReport	 VerticalColumnsReport	 VerticalValuesReport

1.21 Component

1.21.1 Barcode




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

```
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.component;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
import net.sf.dynamicreports.report.builder.component.DimensionComponentBuilder;
import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class BarcodeReport {

        public BarcodeReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(template().setBarcodeHeight(40))
                                        .title(
                                                        Templates.
↪createTitleComponent("Barcode"),
                                                        cmp.text("Barcode4j").
↪setStyle(Templates.bold18CenteredStyle),
                                                        barcode4j(),
                                                        cmp.verticalGap(10),
                                                        cmp.text("ZXing").
↪setStyle(Templates.bold18CenteredStyle),
                                                        barcode("QrCode", bcode.
↪qrCode("12345678")),
                                                        cmp.verticalGap(10),
                                                        cmp.text("Barbecue").
↪setStyle(Templates.bold18CenteredStyle),
                                                        barbecue())
                                        .show();
                } catch (DRException e) {
```

(continues on next page)

(continued from previous page)

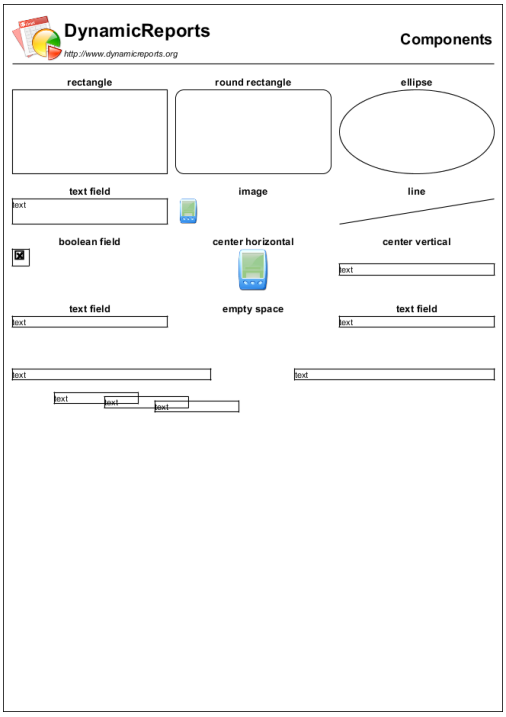
```
                        e.printStackTrace();
                }
        }

        private ComponentBuilder<?, ?> barcode4j() {
                HorizontalListBuilder list = cmp.horizontalFlowList();
                list.setGap(10);
                list.add(
                                barcode("Codabar", bcode.codabar("12345678")),
                                barcode("Code128", bcode.code128("12345678")),
                                barcode("Ean128", bcode.ean128("12345678")),
                                barcode("DataMatrix", bcode.dataMatrix("12345678")),
                                barcode("Code39", bcode.code39("12345678")),
                                barcode("Interleaved2Of5", bcode.interleaved2Of5("12345678")),
                                barcode("Upca", bcode.upca("11000000000")),
                                barcode("Upce", bcode.upce("1100000")),
                                barcode("Ean13", bcode.ean13("110000000000")),
                                barcode("Ean8", bcode.ean8("1100000")),
                                barcode("UspsIntelligentMail", bcode.uspsIntelligentMail("34160265194042788110")),
                                barcode("RoyalMailCustomer", bcode.royalMailCustomer("34160265194042788110")),
                                barcode("Postnet", bcode.postnet("12345678")),
                                barcode("Pdf417", bcode.pdf417("12345678")));
                return list;
        }

        private ComponentBuilder<?, ?> barbecue() {
                HorizontalListBuilder list = cmp.horizontalFlowList();
                list.setGap(10);
                list.add(
                                barcode("2of7", bcode.barbecue_2of7("12345678")),
                                barcode("3of9", bcode.barbecue_3of9("12345678")),
                                barcode("Bookland", bcode.barbecue_bookland("1234567890")),
                                barcode("Codabar", bcode.barbecue_codabar("12345678")),
                                barcode("Code128", bcode.barbecue_code128("12345678")),
                                barcode("Code128A", bcode.barbecue_code128A("12345678")),
                                barcode("Code128B", bcode.barbecue_code128B("12345678")),
                                barcode("Code128C", bcode.barbecue_code128C("12345678")),
                                barcode("Code39", bcode.barbecue_code39("12345678")),
                                barcode("Code39 Extended", bcode.barbecue_code39Extended("12345678")),
                                barcode("Ean128", bcode.barbecue_ean128("12345678")),
                                barcode("Ean13", bcode.barbecue_ean13("123456789012")),
                                barcode("GlobalTradeItemNumber", bcode.barbecue_globalTradeItemNumber("12345678")),
                                barcode("Int2of5", bcode.barbecue_int2of5("12345678")),
                                barcode("Monarch", bcode.barbecue_monarch("12345678")),
```

(continues on next page)

(continued from previous page)

```
                                barcode("Nw7", bcode.barbecue_nw7("12345678")),
                                barcode("Pdf417", bcode.barbecue_pdf417("12345678")),
                                barcode("Postnet", bcode.barbecue_postnet("12345678
↪")),
                                barcode("RandomWeightUpca", bcode.barbecue_
↪randomWeightUpca("12345678901")),
                                barcode("Scc14ShippingCode", bcode.barbecue_
↪scc14ShippingCode("12345678")),
                                barcode("ShipmentIdentificationNumber", bcode.
↪barbecue_shipmentIdentificationNumber("12345678")),
                                barcode("Sscc18", bcode.barbecue_sscc18("12345678")),
                                barcode("Std2of5", bcode.barbecue_std2of5("12345678
↪")),
                                barcode("Ucc128", bcode.barbecue_ucc128("12345678").
↪setApplicationIdentifierExpression("123")),
                                barcode("Upca", bcode.barbecue_upca("12345678901")),
                                barcode("Usd3", bcode.barbecue_usd3("12345678")),
                                barcode("Usd4", bcode.barbecue_usd4("12345678")),
                                barcode("Usps", bcode.barbecue_usps("12345678")));
                return list;
        }

        private ComponentBuilder<?, ?> barcode(String label, DimensionComponentBuilder
↪<?, ?> barcode) {
                return cmp.verticalList(cmp.text(label).setStyle(Templates.
↪bold12CenteredStyle), barcode);
        }

        public static void main(String[] args) {
                new BarcodeReport();
        }
}
```

1.21.2 Components




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.component;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.net.URL;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
```

(continues on next page)

(continued from previous page)

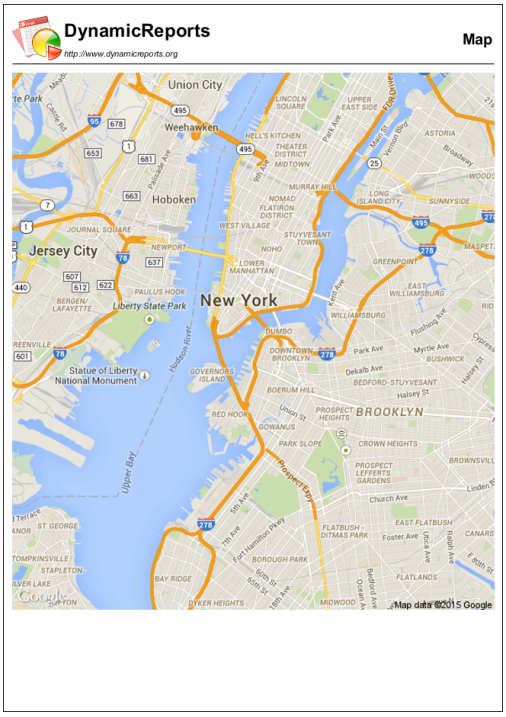
```
import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.constant.BooleanComponentType;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ComponentsReport {

        public ComponentsReport() {
                build();
        }

        private void build() {
                URL image = Templates.class.getResource("images/pda.png");

                try {
                        report()
                                        .setTemplate(template().setBarcodeHeight(50))
                                        .setTextStyle(stl.style(stl.pen1Point()))
                                        .title(
                                                        Templates.
↪createTitleComponent("Components"),
                                                        components(
                                                                        "rectangle",␣
↪cmp.rectangle(),
                                                                        "round␣
↪rectangle", cmp.roundRectangle(10),
                                                                        "ellipse",␣
↪cmp.ellipse()),
                                                        cmp.verticalGap(10),
                                                        components(
                                                                        "text field",␣
↪cmp.text("text"),
                                                                        "image", cmp.
↪image(image).setFixedDimension(30, 30),
                                                                        "line", cmp.
↪line()),
                                                        cmp.verticalGap(10),
                                                        components(
                                                                        "boolean field
↪", cmp.booleanField(true).setComponentType(BooleanComponentType.IMAGE_CHECKBOX_2).
↪setFixedDimension(20, 20),
                                                                        "center␣
↪horizontal", cmp.centerHorizontal(cmp.image(image).setFixedDimension(50, 50)),
                                                                        "center␣
↪vertical", cmp.centerVertical(cmp.text("text").setFixedRows(1))),
                                                        cmp.verticalGap(10),
                                                        components(
                                                                        "text field",␣
↪cmp.text("text"),
                                                                        "empty space",
↪ cmp.filler(),
                                                                        "text field",␣
↪cmp.text("text")),
                                                        cmp.verticalGap(50),
```

(continues on next page)

(continued from previous page)

```
74                                                              cmp.horizontalList(cmp.text(
↪"text"), cmp.horizontalGap(100), cmp.text("text")),
75                                                              cmp.xyList()
76                                                                      .add(50, 15,␣
↪cmp.text("text"))
77                                                                      .add(110, 20,␣
↪cmp.text("text"))
78                                                                      .add(170, 25,␣
↪cmp.text("text")))
79                                              .show();
80                  } catch (DRException e) {
81                          e.printStackTrace();
82                  }
83          }
84
85          private ComponentBuilder<?, ?> components(String label1, ComponentBuilder<?, ?
↪> component1, String label2, ComponentBuilder<?, ?> component2, String label3,
86                          ComponentBuilder<?, ?> component3) {
87                  HorizontalListBuilder list = cmp.horizontalList()
88                                  .setGap(10);
89                  list.add(component(label1, component1));
90                  list.add(component(label2, component2));
91                  list.add(component(label3, component3));
92                  return list;
93          }
94
95          private ComponentBuilder<?, ?> component(String label, ComponentBuilder<?, ?>␣
↪component) {
96                  TextFieldBuilder<String> labelField = cmp.text(label)
97                                  .setFixedRows(1)
98                                  .setStyle(Templates.bold12CenteredStyle);
99                  return cmp.verticalList(labelField, component);
100         }
101
102         public static void main(String[] args) {
103                 new ComponentsReport();
104         }
105 }
```

1.21.3 Map




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.component;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
import net.sf.dynamicreports.report.exception.DRException;

/**
```

(continues on next page)

(continued from previous page)

```
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class MapReport {

        public MapReport() {
                build();
        }

        private void build() {
                try {
                        JasperHtmlExporterBuilder htmlExporter = export.htmlExporter("c:/report.html")
                                                .setImagesDirName("c:/images")
                                                .setOutputImagesToDir(true);

                        report()
                                                .setTemplate(template())
                                                .title(
                                                                Templates.createTitleComponent("Map"),
                                                                cmp.map(40.7f, -74f, 12).setFixedHeight(750))
                                                .toHtml(htmlExporter);
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new MapReport();
        }
}
```

1.21.4 Multi Page List


```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
```

(continues on next page)

(continued from previous page)

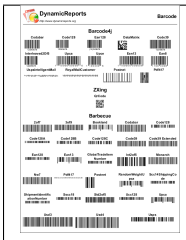
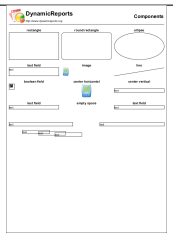
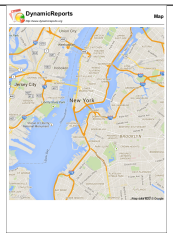
```
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.component;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.component.MultiPageListBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class MultiPageListReport {

        public MultiPageListReport() {
                build();
        }

        private void build() {
                StyleBuilder style = stl.style(stl.pen1Point())
                                .setTextAlignment(HorizontalTextAlignment.CENTER,␣
↪VerticalTextAlignment.MIDDLE);

                MultiPageListBuilder multiPageList = cmp.multiPageList();
                for (int i = 0; i < 10; i++) {
                        TextFieldBuilder<String> textField = cmp.text("Title␣
↪component " + (i + 1))
                                        .setFixedHeight(100)
                                        .setStyle(style);
                        multiPageList.add(textField);
                }

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"MultiPageList"))
                                        .summary(multiPageList)
                                        .pageFooter(Templates.footerComponent)
```

(continues on next page)

(continued from previous page)

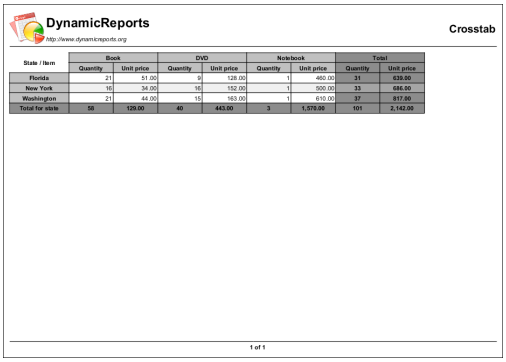
```
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new MultiPageListReport();
        }
}
```

Table 6: Component Examples

 BarcodeReport	 ComponentsReport	 MapReport
 MultiPageListReport		

1.22 Crosstab

1.22.1 Crosstab




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

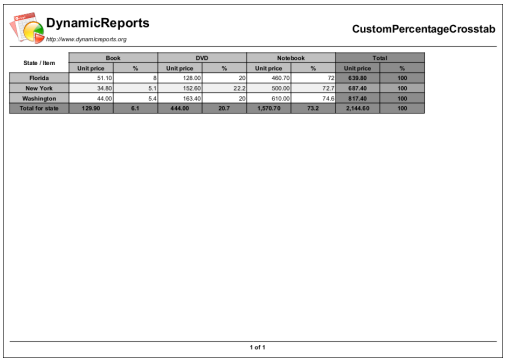
```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.crosstab;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CrosstabReport {

        public CrosstabReport() {
                build();
        }

        private void build() {
                CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",␣
↪String.class)
                                .setTotalHeader("Total for state");

                CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↪"item", String.class);

                CrosstabBuilder crosstab = ctab.crosstab()
                                .headerCell(cmp.text("State / Item").
↪setStyle(Templates.boldCenteredStyle))
                                .rowGroups(rowGroup)
                                .columnGroups(columnGroup)
                                .measures(
                                                ctab.measure("Quantity", "quantity",␣
↪Integer.class, Calculation.SUM),
```

(continues on next page)

(continued from previous page)

```
                                                ctab.measure("Unit price", "unitprice
↪", BigDecimal.class, Calculation.SUM));

                try {
                        report()
                                        .setPageFormat(PageType.A4, PageOrientation.
↪LANDSCAPE)
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"Crosstab"))
                                        .summary(crosstab)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪", "unitprice");
                dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
                dataSource.add("New York", "DVD", 5, new BigDecimal(30));
                dataSource.add("New York", "DVD", 2, new BigDecimal(45));
                dataSource.add("New York", "DVD", 4, new BigDecimal(36));
                dataSource.add("New York", "DVD", 5, new BigDecimal(41));
                dataSource.add("New York", "Book", 2, new BigDecimal(11));
                dataSource.add("New York", "Book", 8, new BigDecimal(9));
                dataSource.add("New York", "Book", 6, new BigDecimal(14));

                dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
                dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
                dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
                dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
                dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
                dataSource.add("Washington", "Book", 3, new BigDecimal(12));
                dataSource.add("Washington", "Book", 9, new BigDecimal(8));
                dataSource.add("Washington", "Book", 4, new BigDecimal(14));
                dataSource.add("Washington", "Book", 5, new BigDecimal(10));

                dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
                dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
                dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
                dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
                dataSource.add("Florida", "Book", 4, new BigDecimal(11));
                dataSource.add("Florida", "Book", 8, new BigDecimal(6));
                dataSource.add("Florida", "Book", 6, new BigDecimal(16));
                dataSource.add("Florida", "Book", 3, new BigDecimal(18));
                return dataSource;
        }

        public static void main(String[] args) {
                new CrosstabReport();
        }
}
```

1.22.2 Custom Percentage Crosstab




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.crosstab;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.List;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

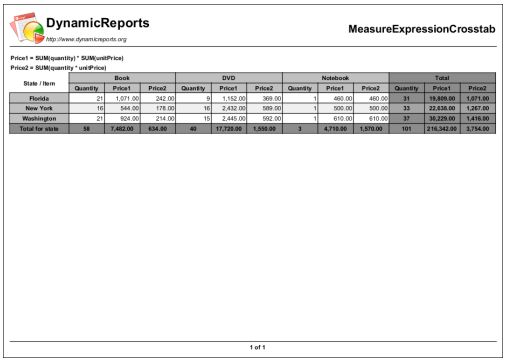
```
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CustomPercentageCrosstabReport {

        public CustomPercentageCrosstabReport() {
                build();
        }

        private void build() {
                CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state", String.class)
                                .setTotalHeader("Total for state");

                CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup("item", String.class);

                FieldBuilder<BigDecimal> quantityField = field("unitprice", BigDecimal.class);

                CrosstabMeasureBuilder<BigDecimal> unitPriceMeasure = ctab.measure("Unit price", quantityField, Calculation.SUM);
                CrosstabMeasureBuilder<BigDecimal> percentageMeasure = ctab.measure("%", new PercentageExpression(unitPriceMeasure, columnGroup));
                percentageMeasure.setDataType(type.doubleType());
                CrosstabBuilder crosstab = ctab.crosstab()
                                .headerCell(cmp.text("State / Item").setStyle(Templates.boldCenteredStyle))
                                .rowGroups(rowGroup)
                                .columnGroups(columnGroup)
                                .measures(unitPriceMeasure, percentageMeasure);

                try {
                        report()
                                        .setPageFormat(PageType.A4, PageOrientation.LANDSCAPE)
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent("CustomPercentageCrosstab"))
                                        .summary(crosstab)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "unitprice");
                dataSource.add("New York", "Notebook", new BigDecimal(500));
                dataSource.add("New York", "DVD", new BigDecimal(30));
                dataSource.add("New York", "DVD", new BigDecimal(45.6));
                dataSource.add("New York", "DVD", new BigDecimal(36));
```

(continues on next page)

(continued from previous page)

```
91                  dataSource.add("New York", "DVD", new BigDecimal(41));
92                  dataSource.add("New York", "Book", new BigDecimal(11));
93                  dataSource.add("New York", "Book", new BigDecimal(9));
94                  dataSource.add("New York", "Book", new BigDecimal(14.8));
95
96                  dataSource.add("Washington", "Notebook", new BigDecimal(610));
97                  dataSource.add("Washington", "DVD", new BigDecimal(40));
98                  dataSource.add("Washington", "DVD", new BigDecimal(35));
99                  dataSource.add("Washington", "DVD", new BigDecimal(46.4));
100                 dataSource.add("Washington", "DVD", new BigDecimal(42));
101                 dataSource.add("Washington", "Book", new BigDecimal(12));
102                 dataSource.add("Washington", "Book", new BigDecimal(8));
103                 dataSource.add("Washington", "Book", new BigDecimal(14));
104                 dataSource.add("Washington", "Book", new BigDecimal(10));
105
106                 dataSource.add("Florida", "Notebook", new BigDecimal(460.7));
107                 dataSource.add("Florida", "DVD", new BigDecimal(49));
108                 dataSource.add("Florida", "DVD", new BigDecimal(32));
109                 dataSource.add("Florida", "DVD", new BigDecimal(47));
110                 dataSource.add("Florida", "Book", new BigDecimal(11));
111                 dataSource.add("Florida", "Book", new BigDecimal(6.1));
112                 dataSource.add("Florida", "Book", new BigDecimal(16));
113                 dataSource.add("Florida", "Book", new BigDecimal(18));
114                 return dataSource;
115         }
116
117         public static void main(String[] args) {
118                 new CustomPercentageCrosstabReport();
119         }
120
121         private class PercentageExpression extends AbstractComplexExpression
    ↪<BigDecimal> {
122                 private static final long serialVersionUID = 1L;
123
124                 private PercentageExpression(CrosstabMeasureBuilder<BigDecimal>␣
    ↪unitPriceMeasure, CrosstabColumnGroupBuilder<String> columnGroup) {
125                         addExpression(unitPriceMeasure);
126                         addExpression(exp.crosstabValue(unitPriceMeasure,␣
    ↪columnGroup));
127                 }
128
129                 @Override
130                 public BigDecimal evaluate(List<?> values, ReportParameters␣
    ↪reportParameters) {
131                         BigDecimal unitPrice = (BigDecimal) values.get(0);
132                         BigDecimal unitPriceTotal = (BigDecimal) values.get(1);
133                         return unitPrice.divide(unitPriceTotal, 4, BigDecimal.ROUND_
    ↪HALF_UP).multiply(new BigDecimal(100));
134                 }
135         }
136 }
```

1.22.3 Measure Expression Crosstab




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.crosstab;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabVariableBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
```

(continues on next page)

(continued from previous page)

```
43
44  /**
45   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46   */
47  public class MeasureExpressionCrosstabReport {
48          private CrosstabMeasureBuilder<Integer> quantityMeasure;
49          private CrosstabVariableBuilder<BigDecimal> unitPriceVariable;
50          private CrosstabVariableBuilder<BigDecimal> priceVariable;
51
52          public MeasureExpressionCrosstabReport() {
53                  build();
54          }
55
56          private void build() {
57                  CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",␣
    ↪String.class)
58                                  .setTotalHeader("Total for state");
59
60                  CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
    ↪"item", String.class);
61
62                  unitPriceVariable = ctab.variable("unitprice", BigDecimal.class,␣
    ↪Calculation.SUM);
63                  priceVariable = ctab.variable(new PriceExpression(), Calculation.SUM);
64                  quantityMeasure = ctab.measure("Quantity", "quantity", Integer.class,␣
    ↪Calculation.SUM);
65                  // price1 = sum(unitprice) * sum(quantity)
66                  CrosstabMeasureBuilder<BigDecimal> priceMeasure1 = ctab.measure(
    ↪"Price1", new PriceMeasure1Expression());
67                  priceMeasure1.setDataType(type.bigDecimalType());
68                  // price2 = sum(unitprice * quantity)
69                  CrosstabMeasureBuilder<BigDecimal> priceMeasure2 = ctab.measure(
    ↪"Price2", new PriceMeasure2Expression());
70                  priceMeasure2.setDataType(type.bigDecimalType());
71
72                  CrosstabBuilder crosstab = ctab.crosstab()
73                                  .headerCell(cmp.text("State / Item").
    ↪setStyle(Templates.boldCenteredStyle))
74                                  .setCellWidth(180)
75                                  .rowGroups(rowGroup)
76                                  .columnGroups(columnGroup)
77                                  .variables(unitPriceVariable, priceVariable)
78                                  .measures(quantityMeasure, priceMeasure1,␣
    ↪priceMeasure2);
79
80                  try {
81                          report()
82                                          .setPageFormat(PageType.A4, PageOrientation.
    ↪LANDSCAPE)
83                                          .setTemplate(Templates.reportTemplate)
84                                          .title(
85                                                          Templates.
    ↪createTitleComponent("MeasureExpressionCrosstab"),
86                                                          cmp.text("Price1 =␣
    ↪SUM(quantity) * SUM(unitPrice)").setStyle(Templates.boldStyle),
87                                                          cmp.text("Price2 =␣
    ↪SUM(quantity * unitPrice)").setStyle(Templates.boldStyle))
```

(continues on next page)

(continued from previous page)

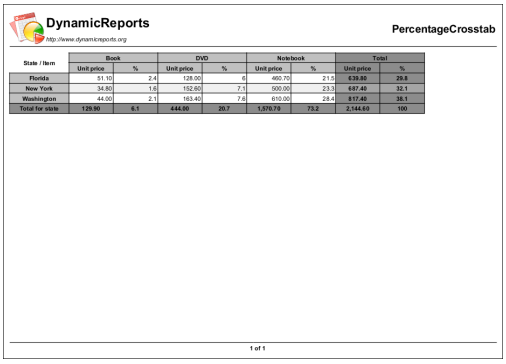
```
                                        .summary(crosstab)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private class PriceExpression extends AbstractSimpleExpression<BigDecimal> {
                private static final long serialVersionUID = 1L;

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        Integer quantity = reportParameters.getValue("quantity");
                        BigDecimal unitPrice = reportParameters.getValue("unitprice");
                        return unitPrice.multiply(new BigDecimal(quantity));
                }
        }

        private class PriceMeasure1Expression extends AbstractSimpleExpression
↪<BigDecimal> {
                private static final long serialVersionUID = 1L;

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        Integer quantity = reportParameters.getValue(quantityMeasure);
                        BigDecimal unitPrice = reportParameters.
↪getValue(unitPriceVariable);
                        return unitPrice.multiply(new BigDecimal(quantity));
                }
        }

        private class PriceMeasure2Expression extends AbstractSimpleExpression
↪<BigDecimal> {
                private static final long serialVersionUID = 1L;

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        return reportParameters.getValue(priceVariable);
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪", "unitprice");
                dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
                dataSource.add("New York", "DVD", 5, new BigDecimal(30));
                dataSource.add("New York", "DVD", 2, new BigDecimal(45));
                dataSource.add("New York", "DVD", 4, new BigDecimal(36));
                dataSource.add("New York", "DVD", 5, new BigDecimal(41));
                dataSource.add("New York", "Book", 2, new BigDecimal(11));
                dataSource.add("New York", "Book", 8, new BigDecimal(9));
                dataSource.add("New York", "Book", 6, new BigDecimal(14));

                dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
                dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
                dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
                dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
                dataSource.add("Washington", "Book", 3, new BigDecimal(12));
                dataSource.add("Washington", "Book", 9, new BigDecimal(8));
                dataSource.add("Washington", "Book", 4, new BigDecimal(14));
                dataSource.add("Washington", "Book", 5, new BigDecimal(10));

                dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
                dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
                dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
                dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
                dataSource.add("Florida", "Book", 4, new BigDecimal(11));
                dataSource.add("Florida", "Book", 8, new BigDecimal(6));
                dataSource.add("Florida", "Book", 6, new BigDecimal(16));
                dataSource.add("Florida", "Book", 3, new BigDecimal(18));
                return dataSource;
        }

        public static void main(String[] args) {
                new MeasureExpressionCrosstabReport();
        }
}
```

1.22.4 Percentage Crosstab




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

(continues on next page)

(continued from previous page)

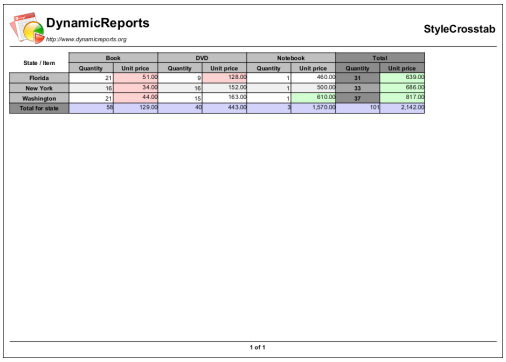
```
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.crosstab;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.CrosstabPercentageType;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PercentageCrosstabReport {

        public PercentageCrosstabReport() {
                build();
        }

        private void build() {
                CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state", String.class)
                                .setTotalHeader("Total for state");

                CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup("item", String.class);

                FieldBuilder<BigDecimal> unitPriceField = field("unitprice", BigDecimal.class);

                CrosstabBuilder crosstab = ctab.crosstab()
                                .headerCell(cmp.text("State / Item").setStyle(Templates.boldCenteredStyle))
                                .rowGroups(rowGroup)
                                .columnGroups(columnGroup)
                                .measures(
                                                ctab.measure("Unit price", unitPriceField, Calculation.SUM),
                                                ctab.measure("%", unitPriceField, Calculation.SUM).setPercentageType(CrosstabPercentageType.GRAND_TOTAL));

```

(continues on next page)

(continued from previous page)

```
                try {
                        report()
                                        .setPageFormat(PageType.A4, PageOrientation.
↪LANDSCAPE)
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent(
↪"PercentageCrosstab"))
                                        .summary(crosstab)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("state", "item", "unitprice
↪");
                dataSource.add("New York", "Notebook", new BigDecimal(500));
                dataSource.add("New York", "DVD", new BigDecimal(30));
                dataSource.add("New York", "DVD", new BigDecimal(45.6));
                dataSource.add("New York", "DVD", new BigDecimal(36));
                dataSource.add("New York", "DVD", new BigDecimal(41));
                dataSource.add("New York", "Book", new BigDecimal(11));
                dataSource.add("New York", "Book", new BigDecimal(9));
                dataSource.add("New York", "Book", new BigDecimal(14.8));

                dataSource.add("Washington", "Notebook", new BigDecimal(610));
                dataSource.add("Washington", "DVD", new BigDecimal(40));
                dataSource.add("Washington", "DVD", new BigDecimal(35));
                dataSource.add("Washington", "DVD", new BigDecimal(46.4));
                dataSource.add("Washington", "DVD", new BigDecimal(42));
                dataSource.add("Washington", "Book", new BigDecimal(12));
                dataSource.add("Washington", "Book", new BigDecimal(8));
                dataSource.add("Washington", "Book", new BigDecimal(14));
                dataSource.add("Washington", "Book", new BigDecimal(10));

                dataSource.add("Florida", "Notebook", new BigDecimal(460.7));
                dataSource.add("Florida", "DVD", new BigDecimal(49));
                dataSource.add("Florida", "DVD", new BigDecimal(32));
                dataSource.add("Florida", "DVD", new BigDecimal(47));
                dataSource.add("Florida", "Book", new BigDecimal(11));
                dataSource.add("Florida", "Book", new BigDecimal(6.1));
                dataSource.add("Florida", "Book", new BigDecimal(16));
                dataSource.add("Florida", "Book", new BigDecimal(18));
                return dataSource;
        }

        public static void main(String[] args) {
                new PercentageCrosstabReport();
        }
}
```

1.22.5 Style Crosstab




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.crosstab;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.builder.style.ConditionalStyleBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
```

(continues on next page)

(continued from previous page)

```
43
44  /**
45   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46   */
47  public class StyleCrosstabReport {
48
49          public StyleCrosstabReport() {
50                  build();
51          }
52
53          private void build() {
54                  CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",␣
    ↪String.class)
55                                  .setTotalHeader("Total for state");
56
57                  CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
    ↪"item", String.class);
58
59                  CrosstabMeasureBuilder<Integer> quantityMeasure = ctab.measure(
    ↪"Quantity", "quantity", Integer.class, Calculation.SUM);
60                  CrosstabMeasureBuilder<BigDecimal> unitPriceMeasure = ctab.measure(
    ↪"Unit price", "unitprice", BigDecimal.class, Calculation.SUM);
61
62                  ConditionalStyleBuilder condition1 = stl.conditionalStyle(cnd.
    ↪greater(unitPriceMeasure, 600))
63                                  .setBackgroundColor(new Color(210, 255, 210))
64                                  .setBorder(stl.pen1Point());
65                  ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.
    ↪smaller(unitPriceMeasure, 150))
66                                  .setBackgroundColor(new Color(255, 210, 210))
67                                  .setBorder(stl.pen1Point());
68
69                  StyleBuilder unitPriceStyle = stl.style()
70                                  .conditionalStyles(condition1, condition2)
71                                  .setBorder(stl.pen1Point());
72                  StyleBuilder totalCellStyle = stl.style()
73                                  .setBackgroundColor(new Color(200, 200, 255))
74                                  .setBorder(stl.pen1Point());
75
76                  unitPriceMeasure.setStyle(unitPriceStyle)
77                                  .setStyle(totalCellStyle, rowGroup)
78                                  .setStyle(totalCellStyle, rowGroup, columnGroup);
79                  quantityMeasure.setStyle(totalCellStyle, rowGroup)
80                                  .setStyle(totalCellStyle, rowGroup, columnGroup);
81
82                  CrosstabBuilder crosstab = ctab.crosstab()
83                                  .headerCell(cmp.text("State / Item").
    ↪setStyle(Templates.boldCenteredStyle))
84                                  .rowGroups(rowGroup)
85                                  .columnGroups(columnGroup)
86                                  .measures(quantityMeasure, unitPriceMeasure);
87
88                  try {
89                          report()
90                                          .setPageFormat(PageType.A4, PageOrientation.
    ↪LANDSCAPE)
91                                          .setTemplate(Templates.reportTemplate)
```

(continues on next page)

(continued from previous page)

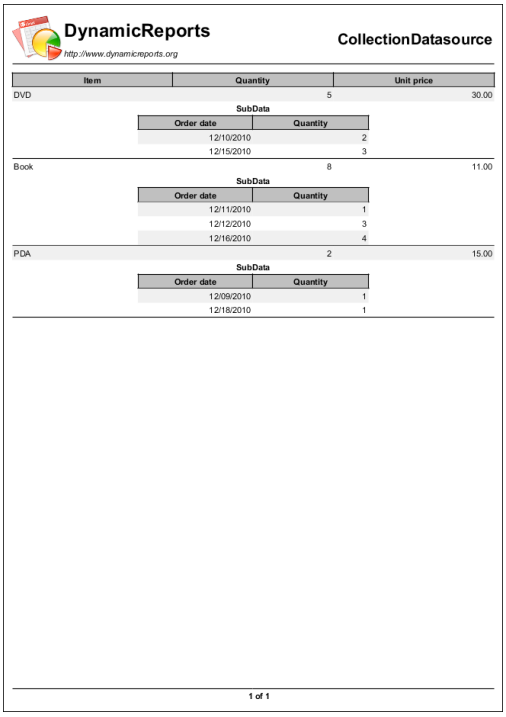
```
92                                          .title(Templates.createTitleComponent(
↪"StyleCrosstab"))
93                                          .summary(crosstab)
94                                          .pageFooter(Templates.footerComponent)
95                                          .setDataSource(createDataSource())
96                                          .show();
97                  } catch (DRException e) {
98                          e.printStackTrace();
99                  }
100         }
101
102         private JRDataSource createDataSource() {
103                 DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪", "unitprice");
104                 dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
105                 dataSource.add("New York", "DVD", 5, new BigDecimal(30));
106                 dataSource.add("New York", "DVD", 2, new BigDecimal(45));
107                 dataSource.add("New York", "DVD", 4, new BigDecimal(36));
108                 dataSource.add("New York", "DVD", 5, new BigDecimal(41));
109                 dataSource.add("New York", "Book", 2, new BigDecimal(11));
110                 dataSource.add("New York", "Book", 8, new BigDecimal(9));
111                 dataSource.add("New York", "Book", 6, new BigDecimal(14));
112
113                 dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
114                 dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
115                 dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
116                 dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
117                 dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
118                 dataSource.add("Washington", "Book", 3, new BigDecimal(12));
119                 dataSource.add("Washington", "Book", 9, new BigDecimal(8));
120                 dataSource.add("Washington", "Book", 4, new BigDecimal(14));
121                 dataSource.add("Washington", "Book", 5, new BigDecimal(10));
122
123                 dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
124                 dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
125                 dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
126                 dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
127                 dataSource.add("Florida", "Book", 4, new BigDecimal(11));
128                 dataSource.add("Florida", "Book", 8, new BigDecimal(6));
129                 dataSource.add("Florida", "Book", 6, new BigDecimal(16));
130                 dataSource.add("Florida", "Book", 3, new BigDecimal(18));
131                 return dataSource;
132         }
133
134         public static void main(String[] args) {
135                 new StyleCrosstabReport();
136         }
137 }
```

Table 7: Column Examples

 CrosstabReport	 CustomPercentageCrosstabReport	 MeasureExpressionCrosstabReport
 PercentageCrosstabReport	 StyleCrosstabReport	

1.23 Datasource

1.23.1 Collection Datasource




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CollectionDatasourceReport {

        public CollectionDatasourceReport() {
                build();
        }

        private void build() {
                SubreportBuilder subreport = cmp.subreport(createSubreport())
                                .setDataSource(exp.subDatasourceBeanCollection(
↪"subData"));

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitPrice", type.bigDecimalType()))
                                        .title(Templates.createTitleComponent(
↪"CollectionDatasource"))
```

(continues on next page)

(continued from previous page)

```
                                        .detailFooter(
                                                cmp.horizontalList(cmp.horizontalGap(150), subreport, cmp.horizontalGap(150)),
                                                cmp.line())
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JasperReportBuilder createSubreport() {
                JasperReportBuilder report = report();
                report
                                .setTemplate(Templates.reportTemplate)
                                .title(cmp.text("SubData").setStyle(Templates.boldCenteredStyle))
                                .columns(
                                                col.column("Order date", "orderDate", type.dateType()),
                                                col.column("Quantity", "quantity", type.integerType()));

                return report;
        }

        private JRDataSource createDataSource() {
                List<Data> data = new ArrayList<Data>();

                List<SubData> subData = new ArrayList<SubData>();
                subData.add(new SubData(toDate(2011, 0, 10), 2));
                subData.add(new SubData(toDate(2011, 0, 15), 3));
                data.add(new Data("DVD", 5, new BigDecimal(30), subData));

                subData = new ArrayList<SubData>();
                subData.add(new SubData(toDate(2011, 0, 11), 1));
                subData.add(new SubData(toDate(2011, 0, 12), 3));
                subData.add(new SubData(toDate(2011, 0, 16), 4));
                data.add(new Data("Book", 8, new BigDecimal(11), subData));

                subData = new ArrayList<SubData>();
                subData.add(new SubData(toDate(2011, 0, 9), 1));
                subData.add(new SubData(toDate(2011, 0, 18), 1));
                data.add(new Data("PDA", 2, new BigDecimal(15), subData));

                return new JRBeanCollectionDataSource(data);
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

```

(continues on next page)

(continued from previous page)

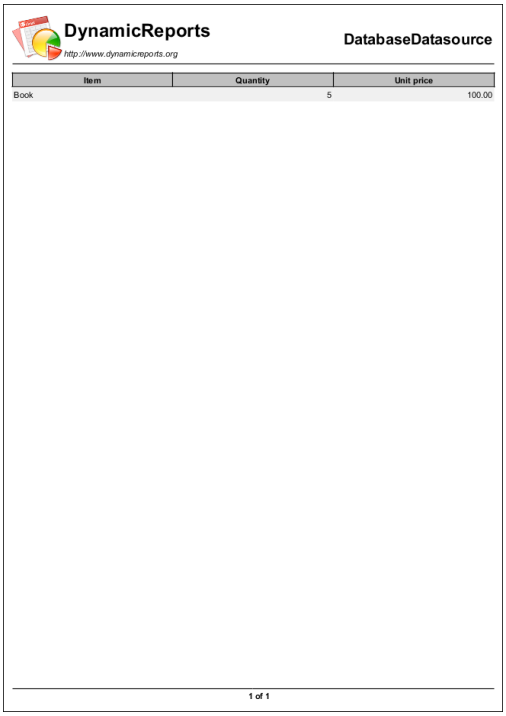
```
114         public static void main(String[] args) {
115                 new CollectionDatasourceReport();
116         }
117
118         public class Data {
119                 private String item;
120                 private Integer quantity;
121                 private BigDecimal unitPrice;
122                 private List<SubData> subData;
123
124                 public Data(String item, Integer quantity, BigDecimal unitPrice, List
    ↪<SubData> subData) {
125                         this.item = item;
126                         this.quantity = quantity;
127                         this.unitPrice = unitPrice;
128                         this.subData = subData;
129                 }
130
131                 public String getItem() {
132                         return item;
133                 }
134
135                 public void setItem(String item) {
136                         this.item = item;
137                 }
138
139                 public Integer getQuantity() {
140                         return quantity;
141                 }
142
143                 public void setQuantity(Integer quantity) {
144                         this.quantity = quantity;
145                 }
146
147                 public BigDecimal getUnitPrice() {
148                         return unitPrice;
149                 }
150
151                 public void setUnitPrice(BigDecimal unitPrice) {
152                         this.unitPrice = unitPrice;
153                 }
154
155                 public List<SubData> getSubData() {
156                         return subData;
157                 }
158
159                 public void setSubData(List<SubData> subData) {
160                         this.subData = subData;
161                 }
162         }
163
164         public class SubData {
165                 private Date orderDate;
166                 private Integer quantity;
167
168                 public SubData(Date orderDate, Integer quantity) {
169                         this.orderDate = orderDate;
```

(continues on next page)

(continued from previous page)

```
                this.quantity = quantity;
        }

        public Date getOrderDate() {
                return orderDate;
        }

        public void setOrderDate(Date orderDate) {
                this.orderDate = orderDate;
        }

        public Integer getQuantity() {
                return quantity;
        }

        public void setQuantity(Integer quantity) {
                this.quantity = quantity;
        }
    }
}
```

1.23.2 Database Datasource




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
```

(continues on next page)

(continued from previous page)

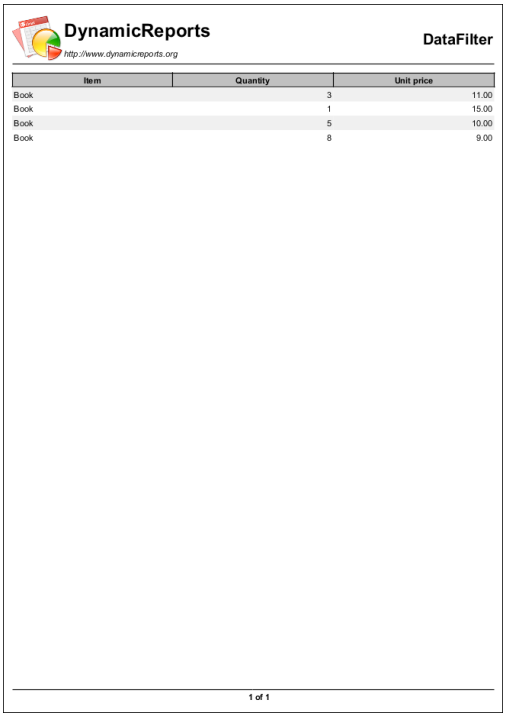
```
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DatabaseDatasourceReport {
        private Connection connection;

        public DatabaseDatasourceReport() {
                try {
                        Class.forName("org.hsqldb.jdbcDriver");
                        connection = DriverManager.getConnection("jdbc:hsqldb:mem:test
↪");
                        createTable();
                        build();
                } catch (SQLException e) {
                        e.printStackTrace();
                } catch (ClassNotFoundException e) {
                        e.printStackTrace();
                }
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
```

(continues on next page)

(continued from previous page)

```
                                                col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .title(Templates.createTitleComponent(
↪"DatabaseDatasource"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource("SELECT * FROM sales",␣
↪connection)
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private void createTable() throws SQLException {
                Statement st = connection.createStatement();
                st.execute("CREATE TABLE sales (item VARCHAR(50), quantity INTEGER,␣
↪unitprice DECIMAL)");
                st.execute("INSERT INTO sales VALUES ('Book', 5, 100)");
        }

        public static void main(String[] args) {
                new DatabaseDatasourceReport();
        }
}
```

1.23.3 Data Filter




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
```

(continues on next page)

(continued from previous page)

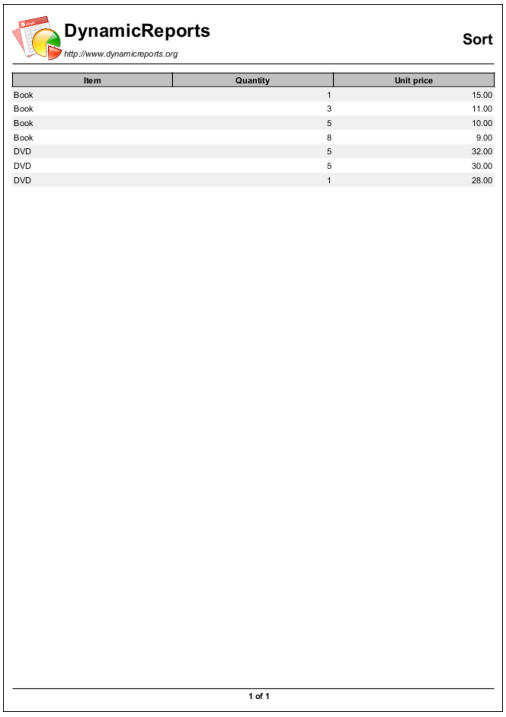
```
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DataFilterReport {

        public DataFilterReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()))
                                        .title(Templates.createTitleComponent("DataFilter"))
                                        .pageFooter(Templates.footerComponent)
                                        .setFilterExpression(new FilterExpression())
```

(continues on next page)

(continued from previous page)

```
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("DVD", 5, new BigDecimal(30));
                dataSource.add("DVD", 1, new BigDecimal(28));
                dataSource.add("DVD", 5, new BigDecimal(32));
                dataSource.add("Book", 3, new BigDecimal(11));
                dataSource.add("Book", 1, new BigDecimal(15));
                dataSource.add("Book", 5, new BigDecimal(10));
                dataSource.add("Book", 8, new BigDecimal(9));
                return dataSource;
        }

        public static void main(String[] args) {
                new DataFilterReport();
        }

        private class FilterExpression extends AbstractSimpleExpression<Boolean> {
                private static final long serialVersionUID = 1L;

                @Override
                public Boolean evaluate(ReportParameters reportParameters) {
                        return reportParameters.getValue("item").equals("Book");
                }
        }
}
```

1.23.4 Sort



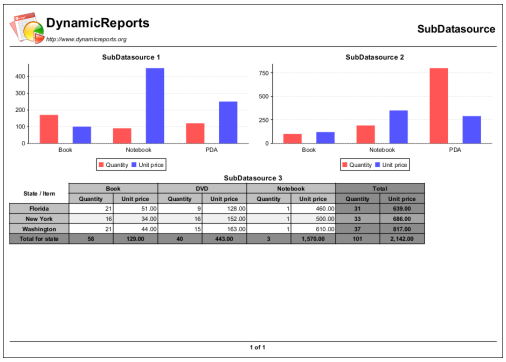

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SortReport {

        public SortReport() {
                build();
        }

        private void build() {
                try {
                        TextColumnBuilder<String> itemColumn = col.column("Item",
↪"item", type.stringType());
                        TextColumnBuilder<Integer> quantityColumn = col.column(
↪"Quantity", "quantity", type.integerType());
                        TextColumnBuilder<BigDecimal> unitPriceColumn = col.column(
↪"Unit price", "unitprice", type.bigDecimalType());
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(itemColumn, quantityColumn,␣
↪unitPriceColumn)
                                        .title(Templates.createTitleComponent("Sort"))
                                        .pageFooter(Templates.footerComponent)
                                        .sortBy(
                                                        asc(itemColumn),␣
↪desc(unitPriceColumn))
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Book", 1, new BigDecimal(15));
                dataSource.add("DVD", 5, new BigDecimal(30));
                dataSource.add("DVD", 1, new BigDecimal(28));
                dataSource.add("Book", 5, new BigDecimal(10));
                dataSource.add("DVD", 5, new BigDecimal(32));
                dataSource.add("Book", 3, new BigDecimal(11));
                dataSource.add("Book", 8, new BigDecimal(9));
                return dataSource;
        }

        public static void main(String[] args) {
                new SortReport();
        }
}
```

1.23.5 Sub Datasource




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.chart.CategoryChartSerieBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
```

(continues on next page)

(continued from previous page)

```

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SubDatasourceReport {

        public SubDatasourceReport() {
                build();
        }

        private void build() {
                FontBuilder boldFont = stl.fontArialBold().setFontSize(12);

                FieldBuilder<String> itemField = field("item", type.stringType());
                FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
                FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());

                CategoryChartSerieBuilder quantitySerie = cht.serie(quantityField).
↪setLabel("Quantity");
                CategoryChartSerieBuilder unitPriceSerie = cht.serie(unitPriceField).
↪setLabel("Unit price");

                BarChartBuilder chart1 = cht.barChart()
                                .setDataSource(createDataSource1())
                                .setTitle("SubDatasource 1")
                                .setTitleFont(boldFont)
                                .setCategory(itemField)
                                .series(
                                                quantitySerie, unitPriceSerie);

                BarChartBuilder chart2 = cht.barChart()
                                .setDataSource(createDataSource2())
                                .setTitle("SubDatasource 2")
                                .setTitleFont(boldFont)
                                .setCategory(itemField)
                                .series(
                                                quantitySerie, unitPriceSerie);

                CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",␣
↪String.class)
                                .setTotalHeader("Total for state");

                CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↪"item", String.class);

                CrosstabBuilder crosstab = ctab.crosstab()
                                .setDataSource(createDataSource3())
                                .headerCell(cmp.text("State / Item").
↪setStyle(Templates.boldCenteredStyle))
                                .rowGroups(rowGroup)
                                .columnGroups(columnGroup)
                                .measures(
                                                ctab.measure("Quantity", "quantity",␣
↪Integer.class, Calculation.SUM),
                                                ctab.measure("Unit price", "unitprice
↪", BigDecimal.class, Calculation.SUM));
```

(continues on next page)

(continued from previous page)

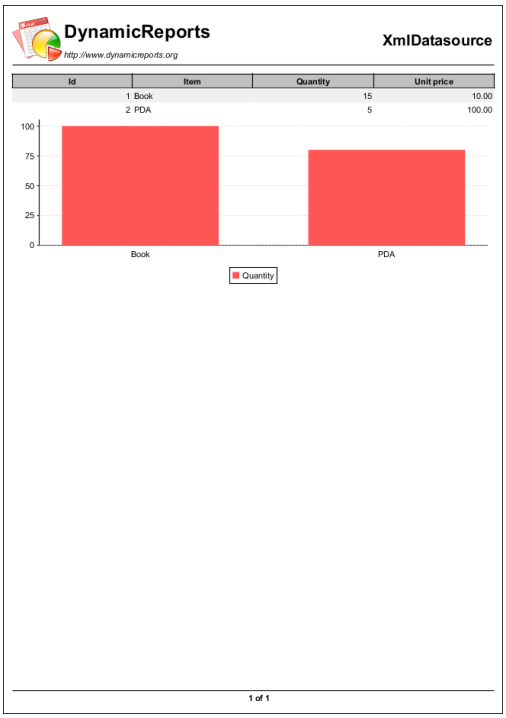
```
92
93                 try {
94                         report()
95                                         .setPageFormat(PageType.A4, PageOrientation.
    ↪LANDSCAPE)
96                                         .setTemplate(Templates.reportTemplate)
97                                         .title(
98                                                         Templates.
    ↪createTitleComponent("SubDatasource"),
99                                                         cmp.horizontalList(chart1,␣
    ↪chart2),
100                                                        cmp.text("SubDatasource 3").
    ↪setStyle(Templates.bold12CenteredStyle),
101                                                        crosstab)
102                                        .pageFooter(Templates.footerComponent)
103                                        .show();
104                 } catch (DRException e) {
105                         e.printStackTrace();
106                 }
107         }
108
109         private JRDataSource createDataSource1() {
110                 DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
111                 dataSource.add("Book", 170, new BigDecimal(100));
112                 dataSource.add("Notebook", 90, new BigDecimal(450));
113                 dataSource.add("PDA", 120, new BigDecimal(250));
114                 return dataSource;
115         }
116
117         private JRDataSource createDataSource2() {
118                 DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
119                 dataSource.add("Book", 100, new BigDecimal(120));
120                 dataSource.add("Notebook", 190, new BigDecimal(350));
121                 dataSource.add("PDA", 800, new BigDecimal(290));
122                 return dataSource;
123         }
124
125         private JRDataSource createDataSource3() {
126                 DRDataSource dataSource = new DRDataSource("state", "item", "quantity
    ↪", "unitprice");
127                 dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
128                 dataSource.add("New York", "DVD", 5, new BigDecimal(30));
129                 dataSource.add("New York", "DVD", 2, new BigDecimal(45));
130                 dataSource.add("New York", "DVD", 4, new BigDecimal(36));
131                 dataSource.add("New York", "DVD", 5, new BigDecimal(41));
132                 dataSource.add("New York", "Book", 2, new BigDecimal(11));
133                 dataSource.add("New York", "Book", 8, new BigDecimal(9));
134                 dataSource.add("New York", "Book", 6, new BigDecimal(14));
135
136                 dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
137                 dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
138                 dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
139                 dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
140                 dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
141                 dataSource.add("Washington", "Book", 3, new BigDecimal(12));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Washington", "Book", 9, new BigDecimal(8));
                dataSource.add("Washington", "Book", 4, new BigDecimal(14));
                dataSource.add("Washington", "Book", 5, new BigDecimal(10));

                dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
                dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
                dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
                dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
                dataSource.add("Florida", "Book", 4, new BigDecimal(11));
                dataSource.add("Florida", "Book", 8, new BigDecimal(6));
                dataSource.add("Florida", "Book", 6, new BigDecimal(16));
                dataSource.add("Florida", "Book", 3, new BigDecimal(18));
                return dataSource;
        }

        public static void main(String[] args) {
                new SubDatasourceReport();
        }
}
```

1.23.6 Xml Datasource




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

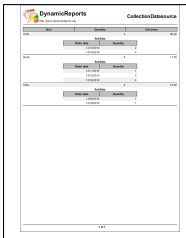
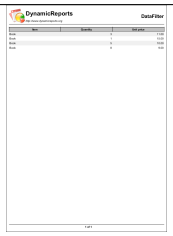
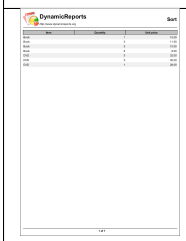
```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.datasource;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.data.JRXmlDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class XmlDatasourceReport {

        public XmlDatasourceReport() {
                build();
        }

        private void build() {
                try {
                        JRXmlDataSource dataSource = new␣
↪JRXmlDataSource(XmlDatasourceReport.class.getResourceAsStream("sales.xml"), "/sales/
↪item");
                        JRXmlDataSource chartDataSource = dataSource.dataSource("/
↪sales/chart/item");

                        FieldBuilder<Object> idField = field("id", type.integerType())
                                        .setDescription("@id");
                        FieldBuilder<String> itemField = field("item", type.
↪stringType());
                        FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
                        FieldBuilder<BigDecimal> unitPriceField = field("unitprice",␣
↪type.bigDecimalType());

                        BarChartBuilder barChart = cht.barChart()
                                        .setDataSource(chartDataSource)
                                        .setCategory(itemField)
```

(continues on next page)

(continued from previous page)

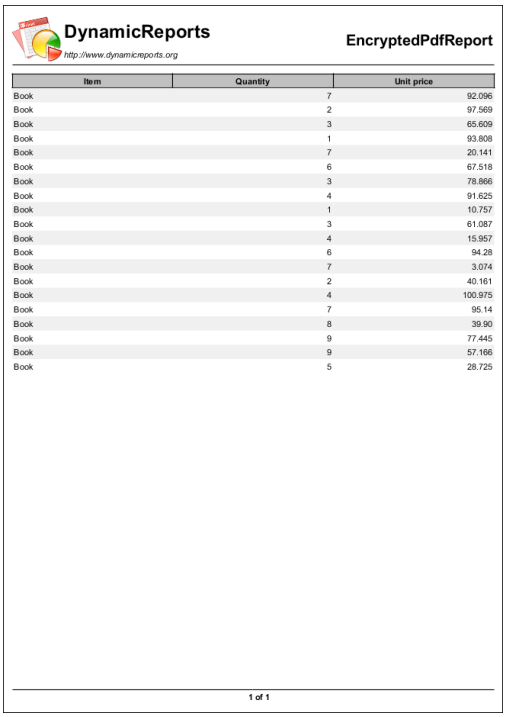
```
                                        .series(
                                                cht.serie(quantityField).
↪setLabel("Quantity"));

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                col.column("Id", idField),
                                                col.column("Item", itemField),
                                                col.column("Quantity",␣
↪quantityField),
                                                col.column("Unit price",␣
↪unitPriceField))
                                        .title(Templates.createTitleComponent(
↪"XmlDatasource"))
                                        .summary(barChart)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(dataSource)
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                } catch (JRException e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new XmlDatasourceReport();
        }
}
```

Table 8: Column Examples

 CollectionDatasourceReport	 DatabaseDatasourceReport	 DataFilterReport
 SortReport	 SubDatasourceReport	 XmlDatasourceReport

1.24 Exporter

1.24.1 Encrypted Pdf



DynamicReports
http://www.dynamicreports.org

EncryptedPdfReport

Item	Quantity	Unit price
Book	7	92.096
Book	2	97.569
Book	3	65.609
Book	1	93.808
Book	7	20.141
Book	6	67.518
Book	3	78.866
Book	4	91.625
Book	1	10.757
Book	3	61.087
Book	4	15.957
Book	6	94.28
Book	7	3.074
Book	2	40.161
Book	4	100.975
Book	7	95.14
Book	8	39.90
Book	9	77.445
Book	9	57.166
Book	5	28.725

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.exporter;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
```

(continues on next page)

(continued from previous page)

```
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.jasper.builder.export.JasperPdfExporterBuilder;
31  import net.sf.dynamicreports.report.datasource.DRDataSource;
32  import net.sf.dynamicreports.report.exception.DRException;
33  import net.sf.jasperreports.engine.JRDataSource;
34
35  /**
36   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37   */
38  public class EncryptedPdfReport {
39
40          public EncryptedPdfReport() {
41                  build();
42          }
43
44          private void build() {
45                  try {
46                          JasperPdfExporterBuilder pdfExporter = export.pdfExporter("c:/
    ↪report.pdf")
47                                                  .setEncrypted(true)
48                                                  .setUserPassword("1234");
49
50                          report()
51                                                  .setTemplate(Templates.reportTemplate)
52                                                  .columns(
53                                                                  col.column("Item", "item",␣
    ↪type.stringType()),
54                                                                  col.column("Quantity",
    ↪"quantity", type.integerType()),
55                                                                  col.column("Unit price",
    ↪"unitprice", type.bigDecimalType()))
56                                                  .title(Templates.createTitleComponent(
    ↪"EncryptedPdfReport"))
57                                                  .pageFooter(Templates.footerComponent)
58                                                  .setDataSource(createDataSource())
59                                                  .toPdf(pdfExporter);
60                  } catch (DRException e) {
61                          e.printStackTrace();
62                  }
63          }
64
65          private JRDataSource createDataSource() {
66                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
67                  for (int i = 0; i < 20; i++) {
68                          dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
    ↪BigDecimal(Math.random() * 100 + 1));
69                  }
70                  return dataSource;
71          }
72
73          public static void main(String[] args) {
74                  new EncryptedPdfReport();
75          }
76  }
```

1.24.2 Excel 1

Item	Quantity	Unit price
Book	4	24,238
Book	2	91,841
Book	3	39,703
Book	6	31,667
Book	2	68,379
Book	6	72,355
Book	8	28,255
Book	4	63,426
Book	5	73,515
Book	10	27,053
Book	2	8,264
Book	9	29,859
Book	1	7,125
Book	5	85,24
Book	4	46,506
Book	2	50,246
Book	10	21,545
Book	2	66,284
Book	5	48,059
Book	1	81,291
Book	3	26,244
Book	10	82,535
Book	10	73,472
Book	3	63,822
Book	7	17,185
Book	2	48,646
Book	10	75,732
Book	2	91,706
Book	6	44,922
Book	3	85,189
Book	10	51,142
Book	2	3,389
Book	9	28,057
Book	8	29,451
Book	9	23,268
Book	1	21,425
Book	10	43,70
Book	5	91,311
Book	8	4,085
Book	6	90,414
Book	7	51,592
Book	10	17,074
Book	7	1,471
Book	2	88,715
Book	3	33,508
Book	10	49,169
Book	10	67,184
Book	5	14,346
Book	7	54,586
Book	6	94,589

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.exporter;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.export.JasperXlsExporterBuilder;
import net.sf.dynamicreports.jasper.constant.JasperProperty;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ExcelReport1 {

        public ExcelReport1() {
                build();
        }

        private void build() {
                try {
                        JasperXlsExporterBuilder xlsExporter = export.xlsExporter("c:/
↪report.xls")
                                        .setDetectCellType(true)
                                        .setIgnorePageMargins(true)
                                        .setWhitePageBackground(false)
                                        .setRemoveEmptySpaceBetweenColumns(true);

                        report()
                                        .setColumnTitleStyle(Templates.
↪columnTitleStyle)
                                        .addProperty(JasperProperty.EXPORT_XLS_FREEZE_
↪ROW, "2")
                                        .ignorePageWidth()
                                        .ignorePagination()
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
```

(continues on next page)

(continued from previous page)

```
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .setDataSource(createDataSource())
                                        .toXls(xlsExporter);
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 50; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new ExcelReport1();
        }
}
```

1.24.3 Excel 2

Item	Quantity	Unit price
Very	5	4,462
Very	2	70,434
Very	10	78,238
Very	1	39,965
Very	3	79,392

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
```

(continues on next page)

(continued from previous page)

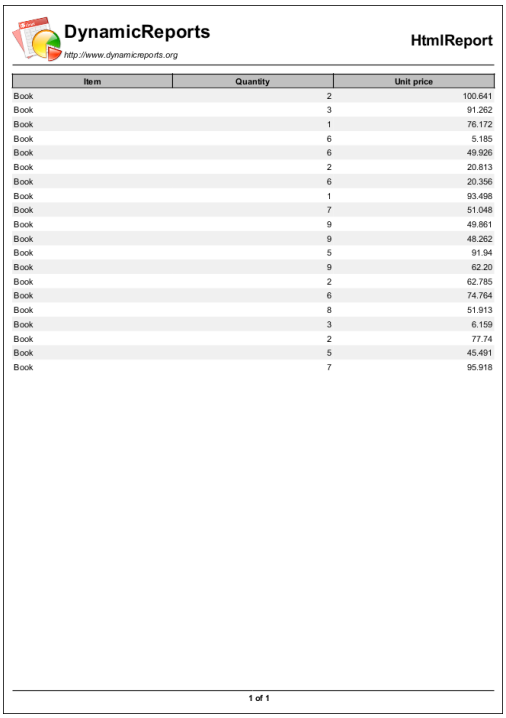
```
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.exporter;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.export.JasperXlsExporterBuilder;
import net.sf.dynamicreports.jasper.constant.JasperProperty;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ExcelReport2 {

        public ExcelReport2() {
                build();
        }

        private void build() {
                try {
                        JasperXlsExporterBuilder xlsExporter = export.xlsExporter("c:/
↪report.xls")
                                        .setDetectCellType(true)
                                        .setIgnorePageMargins(true)
                                        .setWhitePageBackground(false)
                                        .setRemoveEmptySpaceBetweenColumns(true);

                        TextColumnBuilder<String> itemColumn = col.column("Item",
↪"item", type.stringType())
                                        .setFixedWidth(30)
                                        .setStretchWithOverflow(false)
                                        .addProperty(JasperProperty.PRINT_KEEP_FULL_
↪TEXT, "true");

                        report()
                                        .setColumnTitleStyle(Templates.
↪columnTitleStyle)
                                        .addProperty(JasperProperty.EXPORT_XLS_FREEZE_
↪ROW, "2")
                                        .ignorePageWidth()
                                        .ignorePagination()
                                        .columns(
                                                        itemColumn,
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .setDataSource(createDataSource())
                                        .toXls(xlsExporter);
```

(continues on next page)

(continued from previous page)

```
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 5; i++) {
                        dataSource.add("Very long book name", (int) (Math.random() *␣
↪10) + 1, new BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new ExcelReport2();
        }
}
```

1.24.4 Html




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

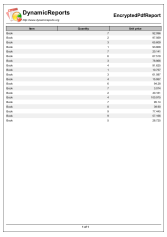
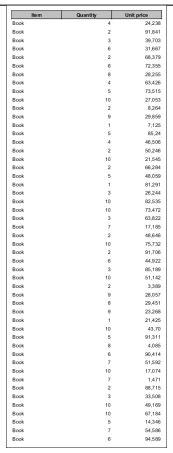
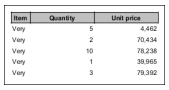
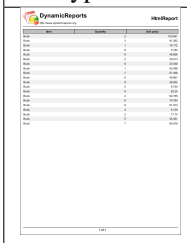
```
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.exporter;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class HtmlReport {

        public HtmlReport() {
                build();
        }

        private void build() {
                try {
                        JasperHtmlExporterBuilder htmlExporter = export.htmlExporter("c:/report.html")
                                        .setImagesDirName("c:/images")
                                        .setOutputImagesToDir(true);

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Item", "item", type.stringType()),
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()))
                                        .title(Templates.createTitleComponent("HtmlReport"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .toHtml(htmlExporter);
```

(continues on next page)

(continued from previous page)

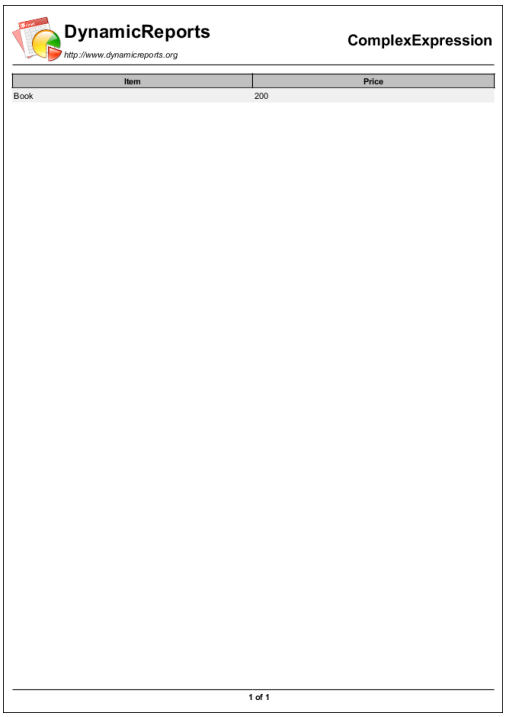
```
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 20; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new HtmlReport();
        }
}
```

Table 9: Exporter Examples

 EncryptedPdfReport	 ExcelReport1	 ExcelReport2
 HtmlReport		

1.25 Expression

1.25.1 Complex Expression




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.expression;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
```

(continues on next page)

(continued from previous page)

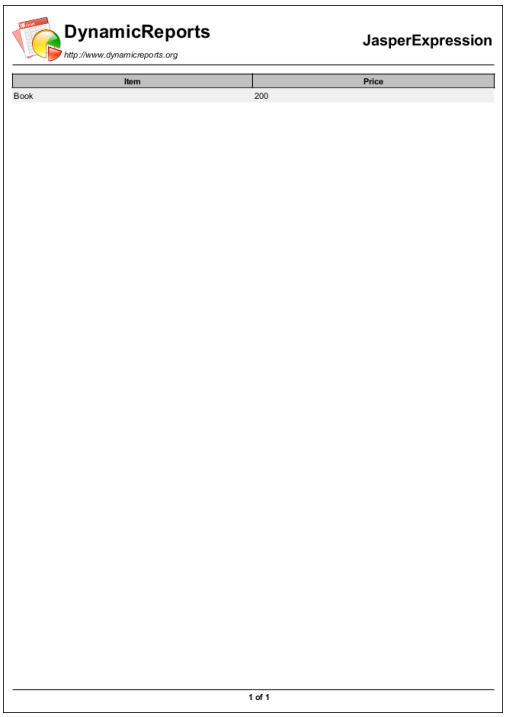
```
28  import java.util.List;
29
30  import net.sf.dynamicreports.examples.Templates;
31  import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
32  import net.sf.dynamicreports.report.datasource.DRDataSource;
33  import net.sf.dynamicreports.report.definition.ReportParameters;
34  import net.sf.dynamicreports.report.exception.DRException;
35  import net.sf.jasperreports.engine.JRDataSource;
36
37  /**
38   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39   */
40  public class ComplexExpressionReport {
41
42          public ComplexExpressionReport() {
43                  build();
44          }
45
46          private void build() {
47                  try {
48                          report()
49                                          .setTemplate(Templates.reportTemplate)
50                                          .columns(
51                                                          col.column("Item", "item",␣
    ↪type.stringType()),
52                                                          col.column("Price", new␣
    ↪ComplexExpression()))
53                                          .title(Templates.createTitleComponent(
    ↪"ComplexExpression"))
54                                          .pageFooter(Templates.footerComponent)
55                                          .setDataSource(createDataSource())
56                                          .show();
57                  } catch (DRException e) {
58                          e.printStackTrace();
59                  }
60          }
61
62          private JRDataSource createDataSource() {
63                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
64                  dataSource.add("Book", 20, new BigDecimal(10));
65                  return dataSource;
66          }
67
68          public static void main(String[] args) {
69                  new ComplexExpressionReport();
70          }
71
72          private class ComplexExpression extends AbstractComplexExpression<BigDecimal>
    ↪{
73                  private static final long serialVersionUID = 1L;
74
75                  public ComplexExpression() {
76                          addExpression(field("quantity", Integer.class));
77                          addExpression(field("unitprice", BigDecimal.class));
78                  }
79
```

(continues on next page)

(continued from previous page)

```
                @Override
                public BigDecimal evaluate(List<?> values, ReportParameters
↪reportParameters) {
                        Integer quantity = (Integer) values.get(0);
                        BigDecimal unitPrice = (BigDecimal) values.get(1);
                        return new BigDecimal(quantity).multiply(unitPrice);
                }
        }
}
```

1.25.2 Jasper Expression




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
```

(continues on next page)

(continued from previous page)

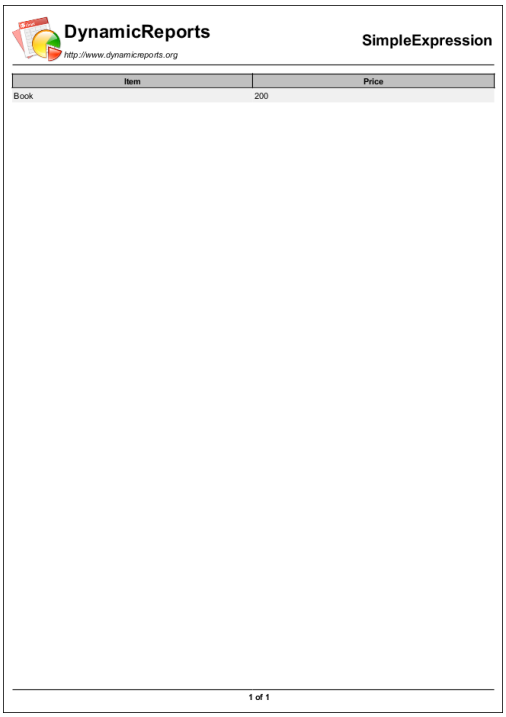
```
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.expression;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.expression.JasperExpression;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class JasperExpressionReport {

        public JasperExpressionReport() {
                build();
        }

        private void build() {
                try {
                        TextColumnBuilder<String> itemColumn = col.column("item", type.stringType())
                                        .setTitle(exp.jasperSyntaxText("Item"));
                        JasperExpression<BigDecimal> priceExpression = exp.jasperSyntax("new BigDecimal($F{quantity}).multiply($F{unitprice})", BigDecimal.class);
                        TextColumnBuilder<BigDecimal> priceColumn = col.column(priceExpression)
                                        .setTitle(exp.jasperSyntaxText("Price"));

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("quantity", Integer.class),
                                                        field("unitprice", BigDecimal.class))
                                        .columns(itemColumn, priceColumn)
                                        .title(Templates.createTitleComponent("JasperExpression"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

```

(continues on next page)

(continued from previous page)

```
68          private JRDataSource createDataSource() {
69                  DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
70                  dataSource.add("Book", 20, new BigDecimal(10));
71                  return dataSource;
72          }
73
74          public static void main(String[] args) {
75                  new JasperExpressionReport();
76          }
77  }
```

1.25.3 Simple Expression




```
1   /**
2    * DynamicReports - Free Java reporting library for creating reports dynamically
3    *
4    * Copyright (C) 2010 - 2018 Ricardo Mariaca
5    * http://www.dynamicreports.org
6    *
7    * This file is part of DynamicReports.
8    *
9    * DynamicReports is free software: you can redistribute it and/or modify
10   * it under the terms of the GNU Lesser General Public License as published by
11   * the Free Software Foundation, either version 3 of the License, or
12   * (at your option) any later version.
13   *
14   * DynamicReports is distributed in the hope that it will be useful,
15   * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

(continues on next page)

(continued from previous page)

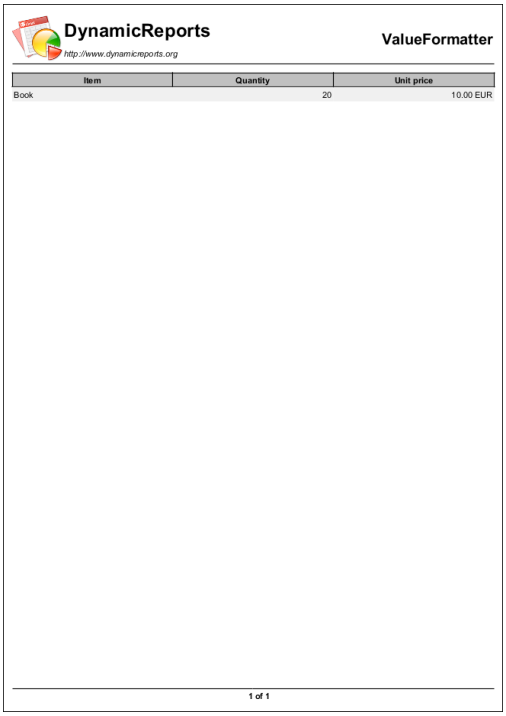
```
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.expression;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SimpleExpressionReport {

        public SimpleExpressionReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("quantity", Integer.
↪class),
                                                        field("unitprice", BigDecimal.
↪class))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Price", new␣
↪SimpleExpression()))
                                        .title(Templates.createTitleComponent(
↪"SimpleExpression"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                dataSource.add("Book", 20, new BigDecimal(10));
```

(continues on next page)

(continued from previous page)

```
                return dataSource;
        }

        public static void main(String[] args) {
                new SimpleExpressionReport();
        }

        private class SimpleExpression extends AbstractSimpleExpression<BigDecimal> {
                private static final long serialVersionUID = 1L;

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        Integer quantity = reportParameters.getValue("quantity");
                        BigDecimal unitPrice = reportParameters.getValue("unitprice");
                        return new BigDecimal(quantity).multiply(unitPrice);
                }
        }
}
```

1.25.4 Value Formatter




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

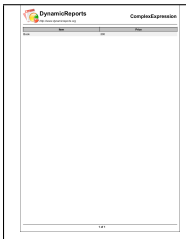
```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.expression;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractValueFormatter;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ValueFormatterReport {

        public ValueFormatterReport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .fields(
                                                        field("unitprice", BigDecimal.
↪class))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()).setValueFormatter(new ValueFormatter()))
                                        .title(Templates.createTitleComponent(
↪"ValueFormatter"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
```

(continues on next page)

(continued from previous page)

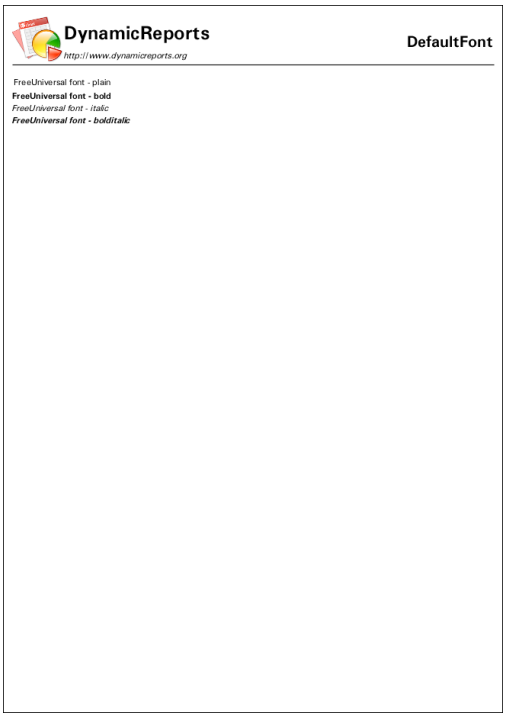
```
61                  }
62          }
63
64          private JRDataSource createDataSource() {
65                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
66                  dataSource.add("Book", 20, new BigDecimal(10));
67                  return dataSource;
68          }
69
70          public static void main(String[] args) {
71                  new ValueFormatterReport();
72          }
73
74          private static class ValueFormatter extends AbstractValueFormatter<String,␣
    ↪Number> {
75                  private static final long serialVersionUID = 1L;
76
77                  @Override
78                  public String format(Number value, ReportParameters reportParameters)
    ↪{
79                          return type.bigDecimalType().valueToString(value,␣
    ↪reportParameters.getLocale()) + " EUR";
80                  }
81          }
82  }
```

Table 10: Expression Examples

 ComplexExpressionReport	 JasperExpressionReport	 SimpleExpressionReport
 ValueFormatterReport		

1.26 Font

1.26.1 Default Font



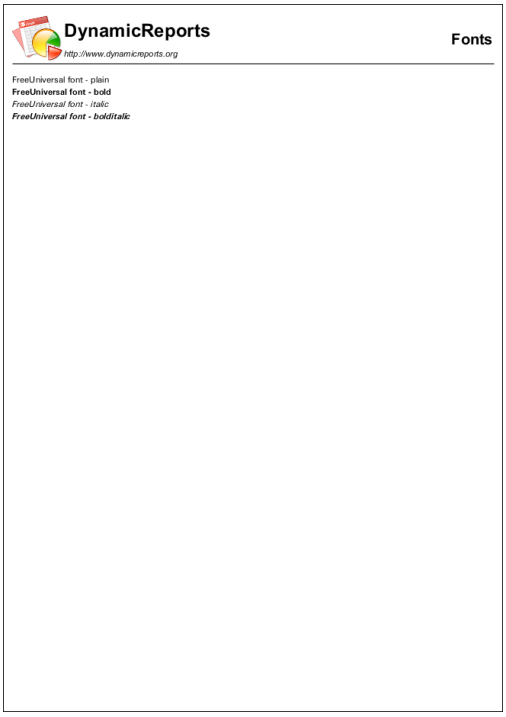

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.fonts;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.style.FontBuilder;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 *
 *         This report is working properly only when the font "FreeUniversal" is registered.
 *         The font is registered in these files: customfonts.xml and jasperreports_extension.properties. The files are located in the root of the classpath.
 */
public class DefaultFontReport {

        public DefaultFontReport() {
                build();
        }

        private void build() {
                FontBuilder defaultFont = stl.font()
                                .setFontName("FreeUniversal");
                StyleBuilder boldStyle = stl.style()
                                .bold();
                StyleBuilder italicStyle = stl.style()
                                .italic();
                StyleBuilder boldItalicStyle = stl.style()
                                .boldItalic();

                try {
                        report()
                                        .setDefaultFont(defaultFont)
                                        .title(
                                                        Templates.createTitleComponent("DefaultFont"),
                                                        cmp.text("FreeUniversal font - plain"),
                                                        cmp.text("FreeUniversal font - bold").setStyle(boldStyle),
                                                        cmp.text("FreeUniversal font - italic").setStyle(italicStyle),
                                                        cmp.text("FreeUniversal font - bolditalic").setStyle(boldItalicStyle))
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new DefaultFontReport();
        }
}
```

1.26.2 Fonts




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.fonts;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.exception.DRException;

/**
```

(continues on next page)

(continued from previous page)

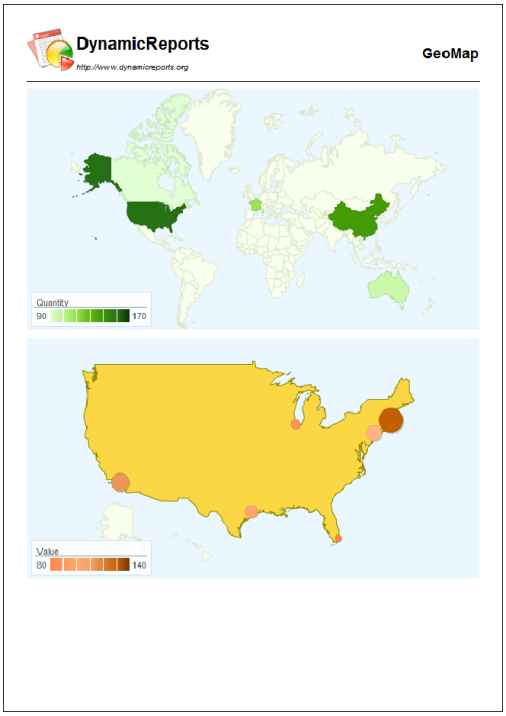
```
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 *
 *         This report is working properly only when the font "FreeUniversal" is registered.
 *         The font is registered in these files: customfonts.xml and jasperreports_extension.properties. The files are located in the root of the classpath.
 */
public class FontsReport {

        public FontsReport() {
                build();
        }

        private void build() {
                StyleBuilder plainStyle = stl.style()
                                .setFontName("FreeUniversal");
                StyleBuilder boldStyle = stl.style(plainStyle)
                                .bold();
                StyleBuilder italicStyle = stl.style(plainStyle)
                                .italic();
                StyleBuilder boldItalicStyle = stl.style(plainStyle)
                                .boldItalic();

                try {
                        report()
                                        .title(
                                                        Templates.createTitleComponent("Fonts"),
                                                        cmp.text("FreeUniversal font - plain").setStyle(plainStyle),
                                                        cmp.text("FreeUniversal font - bold").setStyle(boldStyle),
                                                        cmp.text("FreeUniversal font - italic").setStyle(italicStyle),
                                                        cmp.text("FreeUniversal font - bolditalic").setStyle(boldItalicStyle))
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                new FontsReport();
        }
}
```

Table 11: Font Examples

 DefaultFontReport	 FontsReport

1.27 GeoMap

1.27.1 Geo Map




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
```

(continues on next page)

(continued from previous page)

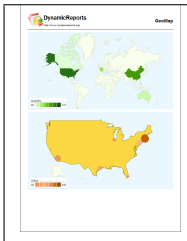
```
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.googlechart;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.googlecharts.report.GoogleCharts;
import net.sf.dynamicreports.googlecharts.report.geomap.GeoMapBuilder;
import net.sf.dynamicreports.googlecharts.report.geomap.GeoMapDataMode;
import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 *
 *         GeoMap component works only for html export.
 *         Visit the following link for more information about geo map parameters
 *         http://code.google.com/apis/chart/interactive/docs/gallery/geomap.html
 */
public class GeoMapReport {

        public GeoMapReport() {
                build();
        }

        private void build() {
                GeoMapBuilder geoMap1 = GoogleCharts.geoMap()
                                .setDataSource(createDataSource1())
                                .setLocation("location", String.class)
                                .setValue("quantity", Integer.class)
                                .setLabel("label", String.class)
                                .setValueLabel("Quantity")
                                .setFixedHeight(300);

                GeoMapBuilder geoMap2 = GoogleCharts.geoMap()
                                .setDataSource(createDataSource2())
                                .setDataMode(GeoMapDataMode.MARKERS)
                                .setRegion("US")
                                .colors(Color.decode("#FF8747"), Color.decode("#FFB581
↪"), Color.decode("#C06000"))
                                .setLocation("location", String.class)
                                .setValue("quantity", Integer.class)
                                .setFixedHeight(300);

                try {
```

(continues on next page)

(continued from previous page)

```
                        JasperHtmlExporterBuilder htmlExporter = export.htmlExporter("c:/report.html")
                                        .setImagesDirName("c:/images")
                                        .setOutputImagesToDir(true);

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(Templates.createTitleComponent("GeoMap"))
                                        .summary(
                                                        geoMap1, cmp.verticalGap(10), geoMap2)
                                        .toHtml(htmlExporter);
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource1() {
                DRDataSource dataSource = new DRDataSource("location", "quantity", "label");
                dataSource.add("US", 170, "United States");
                dataSource.add("CA", 90, "Canada");
                dataSource.add("FR", 120, "France");
                dataSource.add("AU", 100, "Australia");
                dataSource.add("CN", 150, "China");
                return dataSource;
        }

        private JRDataSource createDataSource2() {
                DRDataSource dataSource = new DRDataSource("location", "quantity");
                dataSource.add("New York", 110);
                dataSource.add("Boston", 140);
                dataSource.add("Miami", 80);
                dataSource.add("Chicago", 90);
                dataSource.add("Los Angeles", 120);
                dataSource.add("Houston", 100);
                return dataSource;
        }

        public static void main(String[] args) {
                new GeoMapReport();
        }
}
```

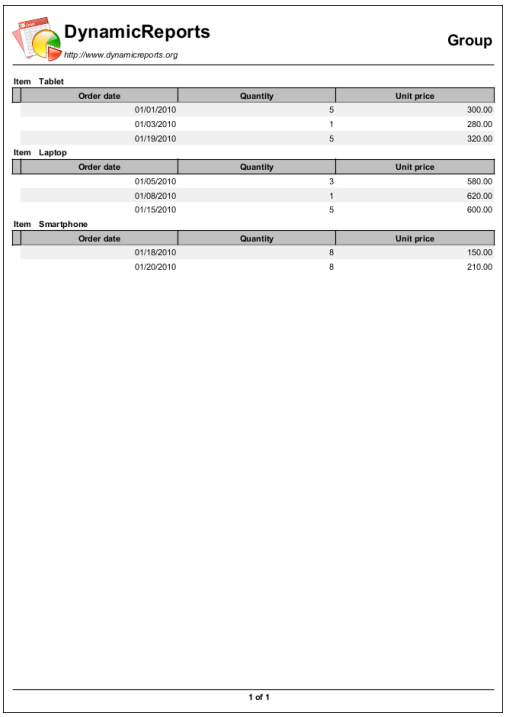
Table 12: GeoMap Examples



GeoMapReport

1.28 Group

1.28.1 Column Group




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
```

(continues on next page)

(continued from previous page)

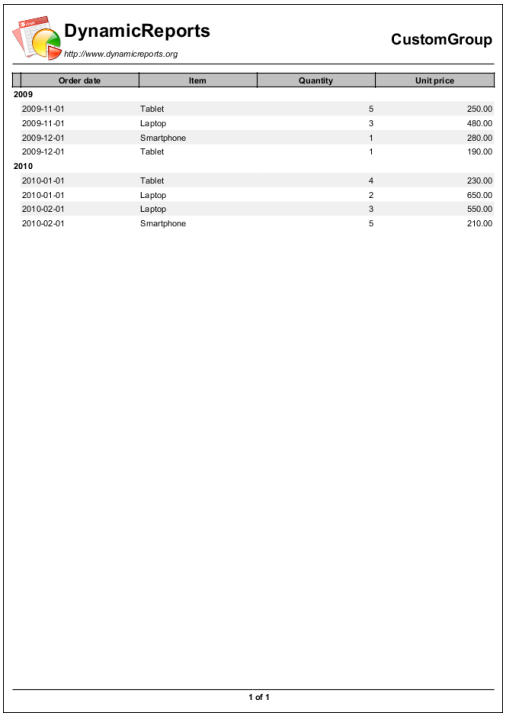
```
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ColumnGroupReport {

        public ColumnGroupReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());

                ColumnGroupBuilder itemGroup = grp.group(itemColumn)
                                .setTitleWidth(30)
                                .setHeaderLayout(GroupHeaderLayout.TITLE_AND_VALUE)
                                .showColumnHeaderAndFooter();

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setShowColumnTitle(false)
                                        .columns(
                                                        itemColumn,
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .groupBy(itemGroup)
                                        .title(Templates.createTitleComponent("Group
↪"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
                dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
                dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
                dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
                dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
                dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new
↪BigDecimal(150));
                dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new
↪BigDecimal(210));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new ColumnGroupReport();
        }
}
```

1.28.2 Custom Group




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
```

(continues on next page)

(continued from previous page)

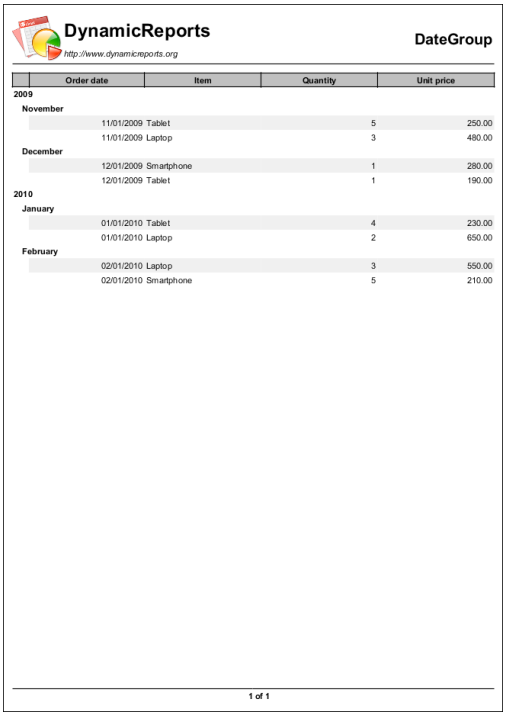
```
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CustomGroupReport {

        public CustomGroupReport() {
                build();
        }

        private void build() {
                CustomGroupBuilder yearGroup = grp.group(new YearExpression());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Order date",
↪"orderdate", type.stringType()),
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
```

(continues on next page)

(continued from previous page)

```
                                        .groupBy(yearGroup)
                                        .title(Templates.createTitleComponent("CustomGroup"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private class YearExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        String orderDate = reportParameters.getValue("orderdate");
                        return orderDate.split("-")[0];
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("orderdate", "item", "quantity", "unitprice");
                dataSource.add("2009-11-01", "Tablet", 5, new BigDecimal(250));
                dataSource.add("2009-11-01", "Laptop", 3, new BigDecimal(480));
                dataSource.add("2009-12-01", "Smartphone", 1, new BigDecimal(280));
                dataSource.add("2009-12-01", "Tablet", 1, new BigDecimal(190));
                dataSource.add("2010-01-01", "Tablet", 4, new BigDecimal(230));
                dataSource.add("2010-01-01", "Laptop", 2, new BigDecimal(650));
                dataSource.add("2010-02-01", "Laptop", 3, new BigDecimal(550));
                dataSource.add("2010-02-01", "Smartphone", 5, new BigDecimal(210));
                return dataSource;
        }

        public static void main(String[] args) {
                new CustomGroupReport();
        }
}
```

1.28.3 Date Group




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

```

(continues on next page)

(continued from previous page)

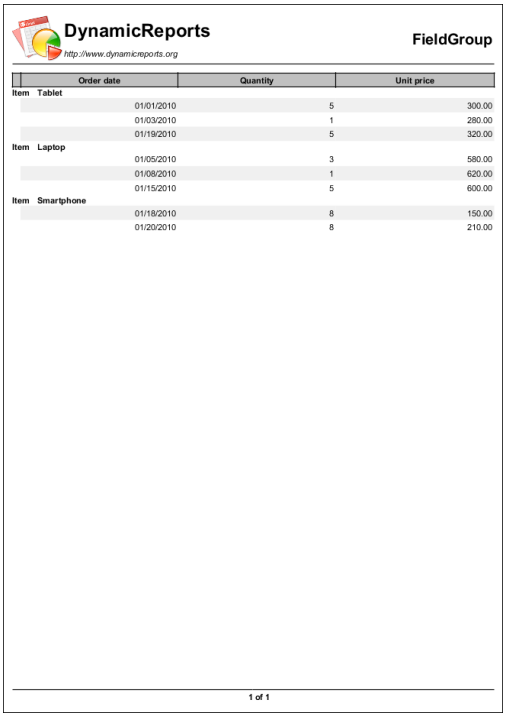
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DateGroupReport {

        public DateGroupReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<Date> yearColumn = col.column("Order year",
↪"orderdate", type.dateYearType());
                TextColumnBuilder<Date> monthColumn = col.column("Order month",
↪"orderdate", type.dateMonthType());

                ColumnGroupBuilder yearGroup = grp.group(yearColumn)
                                .groupByDataType();
                ColumnGroupBuilder monthGroup = grp.group(monthColumn)
                                .groupByDataType();

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        yearColumn,
                                                        monthColumn,
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .groupBy(
                                                        yearGroup, monthGroup)
                                        .title(Templates.createTitleComponent(
↪"DateGroup"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪"quantity", "unitprice");
                dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
                dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new
↪BigDecimal(280));
                dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
                dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
                dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
                dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
                dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new
↪BigDecimal(210));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new DateGroupReport();
        }
}
```

1.28.4 Field Group




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
```

(continues on next page)

(continued from previous page)

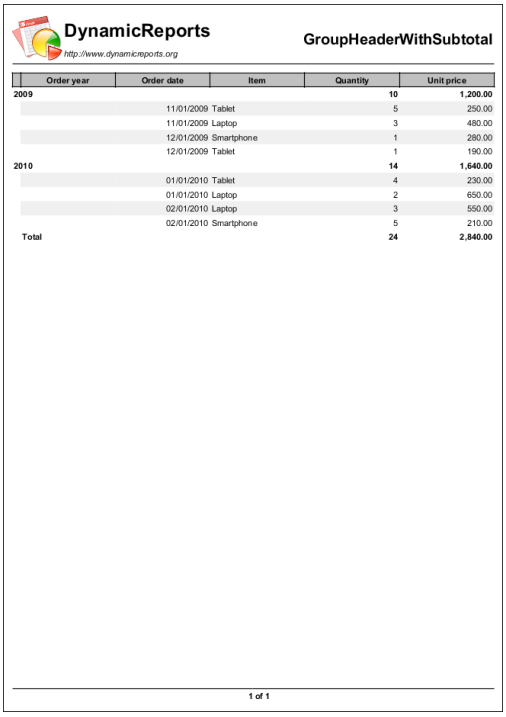
```
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class FieldGroupReport {

        public FieldGroupReport() {
                build();
        }

        private void build() {
                StyleBuilder groupStyle = stl.style().bold();

                CustomGroupBuilder itemGroup = grp.group("item", String.class)
                                .setStyle(groupStyle)
                                .setTitle("Item")
                                .setTitleStyle(groupStyle)
                                .setTitleWidth(30)
                                .setHeaderLayout(GroupHeaderLayout.TITLE_AND_VALUE);

                try {
                        report()
```

(continues on next page)

(continued from previous page)

```
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .groupBy(itemGroup)
                                        .title(Templates.createTitleComponent(
↪"FieldGroup"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
                dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));
                dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
                dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
                dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
                dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
                dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new␣
↪BigDecimal(150));
                dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new␣
↪BigDecimal(210));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new FieldGroupReport();
        }
}
```

1.28.5 Group Header With Subtotal




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

```

(continues on next page)

(continued from previous page)

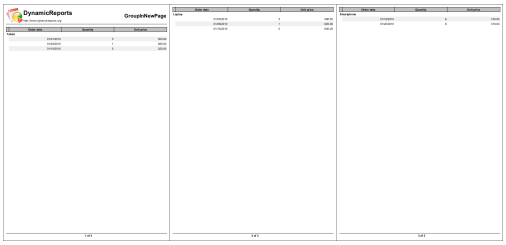
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupHeaderWithSubtotalReport {

        public GroupHeaderWithSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> yearColumn = col.column("Order year", exp.text(""));
                TextColumnBuilder<Date> orderDateColumn = col.column("Order date", "orderdate", type.dateType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                CustomGroupBuilder yearGroup = grp.group(new YearExpression())
                                .groupByDataType()
                                .headerWithSubtotal();

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setSubtotalStyle(stl.style(Templates.boldStyle))
                                        .fields(
                                                        field("orderdate", type.dateYearType()))
                                        .columns(
                                                        yearColumn, orderDateColumn, itemColumn, quantityColumn, unitPriceColumn)
                                        .groupBy(yearGroup)
                                        .subtotalsAtGroupHeader(yearGroup,
                                                        sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))
                                        .subtotalsAtSummary(
                                                        sbt.aggregate(exp.text("Total"), yearColumn, Calculation.NOTHING), sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))
                                        .title(Templates.createTitleComponent("GroupHeaderWithSubtotal"))
                                        .pageFooter(Templates.footerComponent)
```

(continues on next page)

(continued from previous page)

```
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private class YearExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return type.dateYearType().valueToString("orderdate",␣
↪reportParameters);
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪"quantity", "unitprice");
                dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
                dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
                dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new␣
↪BigDecimal(280));
                dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
                dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
                dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
                dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
                dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new␣
↪BigDecimal(210));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new GroupHeaderWithSubtotalReport();
        }
}
```

1.28.6 Group In New Page



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupInNewPageReport {

        public GroupInNewPageReport() {
                build();
        }

```

(continues on next page)

(continued from previous page)

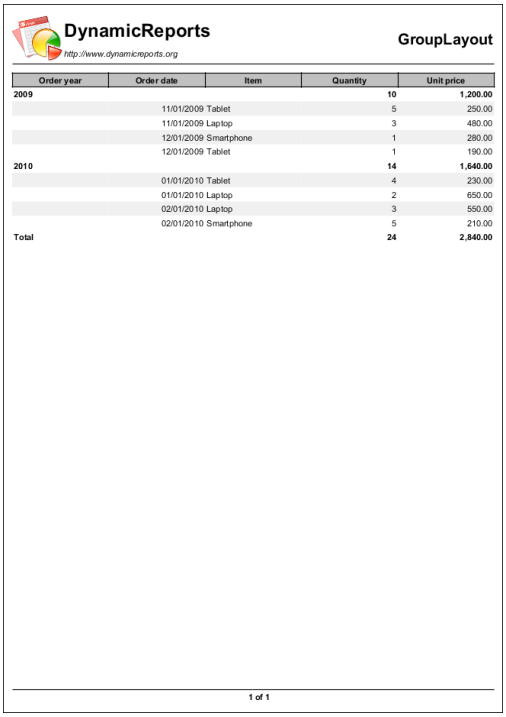
```
47          private void build() {
48                  TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
    ↪type.stringType())
49                                  .setWidth(30);
50
51                  ColumnGroupBuilder itemGroup = grp.group(itemColumn)
52                                  .startInNewPage();
53
54                  try {
55                          report()
56                                          .setTemplate(Templates.reportTemplate)
57                                          .columns(
58                                                          itemColumn,
59                                                          col.column("Order date",
    ↪"orderdate", type.dateType()),
60                                                          col.column("Quantity",
    ↪"quantity", type.integerType()),
61                                                          col.column("Unit price",
    ↪"unitprice", type.bigDecimalType()))
62                                          .groupBy(itemGroup)
63                                          .title(Templates.createTitleComponent(
    ↪"GroupInNewPage"))
64                                          .pageFooter(Templates.footerComponent)
65                                          .setDataSource(createDataSource())
66                                          .show();
67                  } catch (DRException e) {
68                          e.printStackTrace();
69                  }
70          }
71
72          private JRDataSource createDataSource() {
73                  DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪"quantity", "unitprice");
74                  dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
75                  dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));
76                  dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
77                  dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
78                  dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
79                  dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
80                  dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new␣
    ↪BigDecimal(150));
81                  dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new␣
    ↪BigDecimal(310));
82                  return dataSource;
83          }
84
85          private Date toDate(int year, int month, int day) {
86                  Calendar c = Calendar.getInstance();
87                  c.set(Calendar.YEAR, year);
88                  c.set(Calendar.MONTH, month - 1);
89                  c.set(Calendar.DAY_OF_MONTH, day);
90                  return c.getTime();
91          }
92
93          public static void main(String[] args) {
94                  new GroupInNewPageReport();
95          }
```

(continues on next page)

(continued from previous page)

```
}
```

1.28.7 Group Layout



DynamicReports
http://www.dynamicreports.org

GroupLayout

Order year	Order date	Item	Quantity	Unit price
2009			10	1,200.00
	11/01/2009	Tablet	5	250.00
	11/01/2009	Laptop	3	480.00
	12/01/2009	Smartphone	1	280.00
	12/01/2009	Tablet	1	190.00
2010			14	1,640.00
	01/01/2010	Tablet	4	230.00
	01/01/2010	Laptop	2	650.00
	02/01/2010	Laptop	3	550.00
	02/01/2010	Smartphone	5	210.00
Total			24	2,840.00

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
```

(continues on next page)

(continued from previous page)

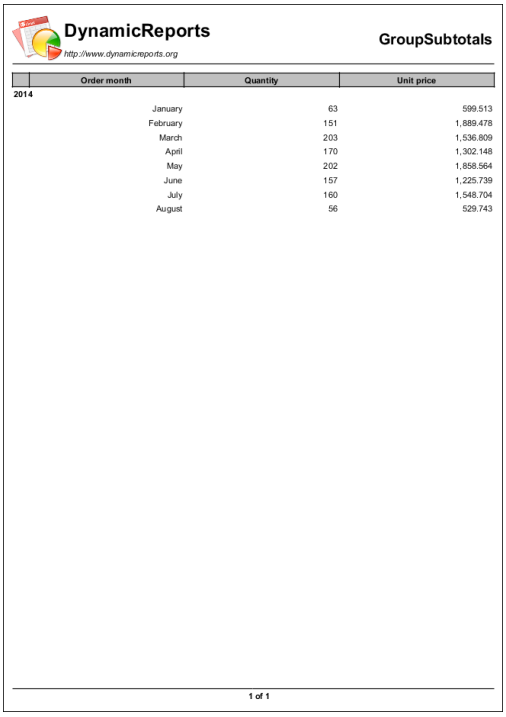
```
26
27  import java.math.BigDecimal;
28  import java.util.Calendar;
29  import java.util.Date;
30
31  import net.sf.dynamicreports.examples.Templates;
32  import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
33  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34  import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
35  import net.sf.dynamicreports.report.constant.Calculation;
36  import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
37  import net.sf.dynamicreports.report.datasource.DRDataSource;
38  import net.sf.dynamicreports.report.definition.ReportParameters;
39  import net.sf.dynamicreports.report.exception.DRException;
40  import net.sf.jasperreports.engine.JRDataSource;
41
42  /**
43   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44   */
45  public class GroupLayoutReport {
46
47          public GroupLayoutReport() {
48                  build();
49          }
50
51          private void build() {
52                  TextColumnBuilder<String> yearColumn = col.column("Order year", exp.
    ↪text(""));
53                  TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
    ↪"orderdate", type.dateType());
54                  TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
    ↪type.stringType());
55                  TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
    ↪"quantity", type.integerType());
56                  TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
    ↪", "unitprice", type.bigDecimalType());
57
58                  CustomGroupBuilder yearGroup = grp.group(new YearExpression())
59                                  .groupByDataType()
60                                  .setHeaderLayout(GroupHeaderLayout.EMPTY)
61                                  .setPadding(0);
62
63                  try {
64                          report()
65                                          .setTemplate(Templates.reportTemplate)
66                                          .setSubtotalStyle(stl.style(Templates.
    ↪boldStyle))
67                                          .fields(
68                                                          field("orderdate", type.
    ↪dateYearType()))
69                                          .columns(
70                                                          yearColumn, orderDateColumn,␣
    ↪itemColumn, quantityColumn, unitPriceColumn)
71                                          .groupBy(yearGroup)
72                                          .subtotalsAtGroupHeader(yearGroup,
73                                                          sbt.first(new␣
    ↪YearExpression(), yearColumn), sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))
```

(continues on next page)

(continued from previous page)

```
74                                          .subtotalsAtSummary(
75                                                  sbt.aggregate(exp.text("Total
↪"), yearColumn, Calculation.NOTHING), sbt.sum(quantityColumn), sbt.
↪sum(unitPriceColumn))
76                                          .title(Templates.createTitleComponent(
↪"GroupLayout"))
77                                          .pageFooter(Templates.footerComponent)
78                                          .setDataSource(createDataSource())
79                                          .show();
80                  } catch (DREception e) {
81                          e.printStackTrace();
82                  }
83          }
84
85          private class YearExpression extends AbstractSimpleExpression<String> {
86                  private static final long serialVersionUID = 1L;
87
88                  @Override
89                  public String evaluate(ReportParameters reportParameters) {
90                          return type.dateYearType().valueToString("orderdate",␣
↪reportParameters);
91                  }
92          }
93
94          private JRDataSource createDataSource() {
95                  DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪"quantity", "unitprice");
96                  dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
97                  dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
98                  dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new␣
↪BigDecimal(280));
99                  dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
100                 dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
101                 dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
102                 dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
103                 dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new␣
↪BigDecimal(210));
104                 return dataSource;
105         }
106
107         private Date toDate(int year, int month, int day) {
108                 Calendar c = Calendar.getInstance();
109                 c.set(Calendar.YEAR, year);
110                 c.set(Calendar.MONTH, month - 1);
111                 c.set(Calendar.DAY_OF_MONTH, day);
112                 return c.getTime();
113         }
114
115         public static void main(String[] args) {
116                 new GroupLayoutReport();
117         }
118 }
```

1.28.8 Group Subtotals



DynamicReports
http://www.dynamicreports.org

GroupSubtotals

	Order month	Quantity	Unit price
2014			
	January	63	599.513
	February	151	1,889.478
	March	203	1,536.809
	April	170	1,302.148
	May	202	1,858.564
	June	157	1,225.739
	July	160	1,548.704
	August	56	529.743

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.group;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

```

(continues on next page)

(continued from previous page)

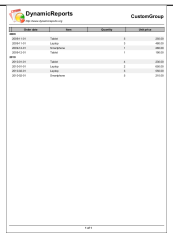
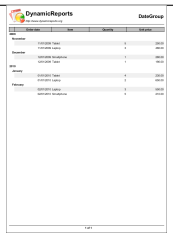
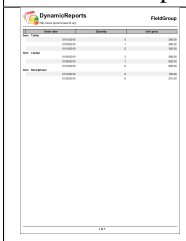
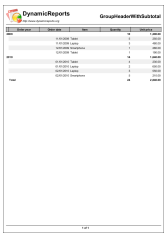
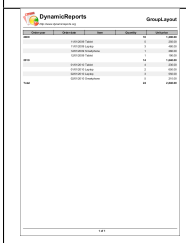
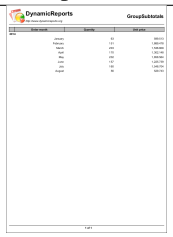
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupSubtotalsReport {

        public GroupSubtotalsReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<Date> yearColumn = col.column("Order year",
↪"orderdate", type.dateYearType());
                TextColumnBuilder<Date> monthColumn = col.column("Order month",
↪"orderdate", type.dateMonthType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                ColumnGroupBuilder yearGroup = grp.group(yearColumn)
                                .groupByDataType();
                ColumnGroupBuilder monthGroup = grp.group(monthColumn)
                                .groupByDataType()
                                .setHeaderLayout(GroupHeaderLayout.EMPTY)
                                .setHideColumn(false);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setSubtotalStyle(Templates.columnStyle)
                                        .setShowColumnValues(false)
                                        .columns(yearColumn, monthColumn,␣
↪quantityColumn, unitPriceColumn)
                                        .groupBy(yearGroup, monthGroup)
                                        .subtotalsAtGroupFooter(monthGroup, sbt.
↪first(monthColumn), sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))
                                        .title(Templates.createTitleComponent(
↪"GroupSubtotals"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("orderdate", "quantity",
↪"unitprice");
```

(continues on next page)

(continued from previous page)

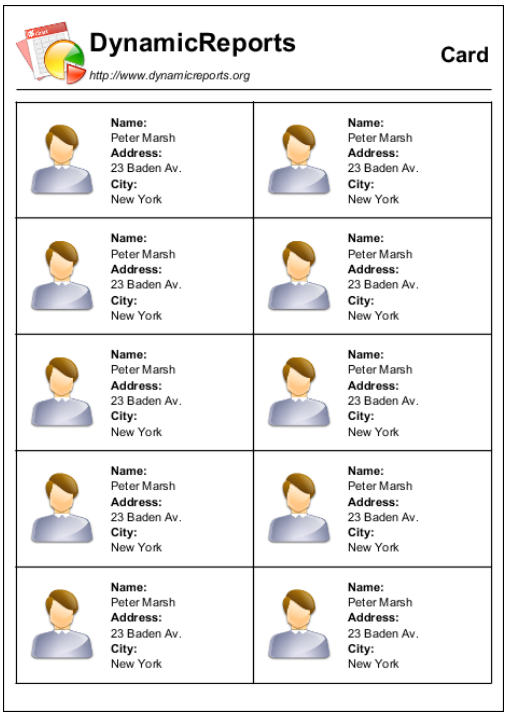
```
                Calendar c = Calendar.getInstance();
                c.add(Calendar.YEAR, -1);
                for (int i = 0; i < 200; i++) {
                        Date date = c.getTime();
                        dataSource.add(date, (int) (Math.random() * 10) + 1, new
↪BigDecimal(Math.random() * 100 + 1));
                        c.add(Calendar.DAY_OF_MONTH, 1);
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new GroupSubtotalsReport();
        }
}
```

Table 13: Group Examples

 ColumnGroupReport	 CustomGroupReport	 DateGroupReport
 FieldGroupReport	 GroupHeaderWithSubtotalReport	 GroupInNewPageReport
 GroupLayoutReport	 GroupSubtotalsReport	

1.29 Miscellaneous

1.29.1 Card




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
```

(continues on next page)

(continued from previous page)

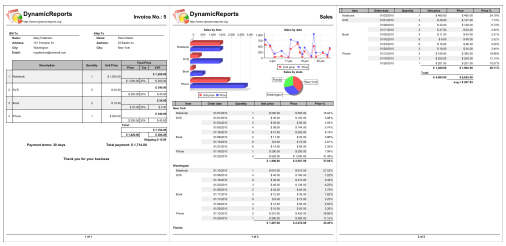
```
import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
import net.sf.dynamicreports.report.builder.component.ImageBuilder;
import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CardReport {

        public CardReport() {
                build();
        }

        private void build() {
                ComponentBuilder<?, ?> cardComponent = createCardComponent();
                HorizontalListBuilder cards = cmp.horizontalFlowList();
                for (int i = 0; i < 10; i++) {
                        cards.add(cardComponent);
                }

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setTextStyle(stl.style())
                                        .setPageFormat(PageType.A5)
                                        .title(
                                                        Templates.
↪createTitleComponent("Card"),
                                                        cards)
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private ComponentBuilder<?, ?> createCardComponent() {
                HorizontalListBuilder cardComponent = cmp.horizontalList();
                StyleBuilder cardStyle = stl.style(stl.pen1Point())
                                .setPadding(10);
                cardComponent.setStyle(cardStyle);

                ImageBuilder image = cmp.image(Templates.class.getResource("images/
↪user_male.png")).setFixedDimension(60, 60);
                cardComponent.add(cmp.hListCell(image).heightFixedOnMiddle());
                cardComponent.add(cmp.horizontalGap(10));

                StyleBuilder boldStyle = stl.style().bold();
                VerticalListBuilder content = cmp.verticalList(
                                cmp.text("Name:").setStyle(boldStyle),
                                cmp.text("Peter Marsh"),
                                cmp.text("Address:").setStyle(boldStyle),
                                cmp.text("23 Baden Av."),
                                cmp.text("City:").setStyle(boldStyle),
                                cmp.text("New York"));
```

(continues on next page)

(continued from previous page)

```
                cardComponent.add(content);
                return cardComponent;
        }

        public static void main(String[] args) {
                new CardReport();
        }
}
```

1.29.2 Concatenated 1



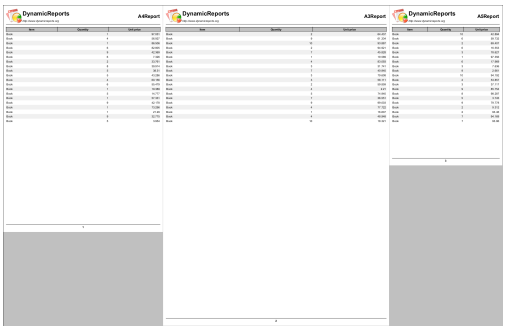
```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.complex.invoice.InvoiceDesign;
import net.sf.dynamicreports.examples.complex.sales.SalesDesign;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.jasper.builder.export.Exporters;
import net.sf.dynamicreports.report.exception.DRException;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
```

(continues on next page)

(continued from previous page)

```
 */
public class ConcatenatedReport1 {

	public ConcatenatedReport1() {
		build();
	}

	private void build() {
		try {
			concatenatedReport()
					.concatenate(
							invoiceReport(), salesReport())
					.toPdf(Exporters.pdfExporter("c:/report.pdf"));
		} catch (DRException e) {
			e.printStackTrace();
		}
	}

	private JasperReportBuilder invoiceReport() throws DRException {
		return new InvoiceDesign().build();
	}

	private JasperReportBuilder salesReport() throws DRException {
		return new SalesDesign().build();
	}

	public static void main(String[] args) {
		new ConcatenatedReport1();
	}
}
```

1.29.3 Concatenated 2



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

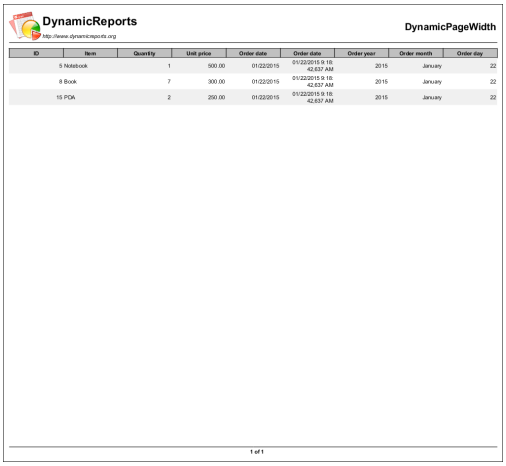
```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.jasper.builder.export.Exporters;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ConcatenatedReport2 {

        public ConcatenatedReport2() {
                build();
        }

        private void build() {
                try {
                        concatenatedReport()
                                        .continuousPageNumbering()
                                        .concatenate(
                                                        createReport(PageType.A4),
                                                        createReport(PageType.A3),
                                                        createReport(PageType.A5))
                                        .toPdf(Exporters.pdfExporter("c:/report.pdf
↪"));
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JasperReportBuilder createReport(PageType pageType) {
                JasperReportBuilder report = report();
                report
                                .setTemplate(Templates.reportTemplate)
                                .setPageFormat(pageType)
```

(continues on next page)

(continued from previous page)

```
                                .columns(
                                                col.column("Item", "item", type.
↪stringType()),
                                                col.column("Quantity", "quantity",␣
↪type.integerType()),
                                                col.column("Unit price", "unitprice",␣
↪type.bigDecimalType()))
                                .title(Templates.createTitleComponent(pageType.name()␣
↪+ "Report"))
                                .pageFooter(
                                                cmp.line(),
                                                cmp.pageNumber().setStyle(Templates.
↪boldCenteredStyle))
                                .setDataSource(createDataSource());

                return report;
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 20; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new ConcatenatedReport2();
        }
}
```

1.29.4 Dynamic Page Width




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
```

(continues on next page)

(continued from previous page)

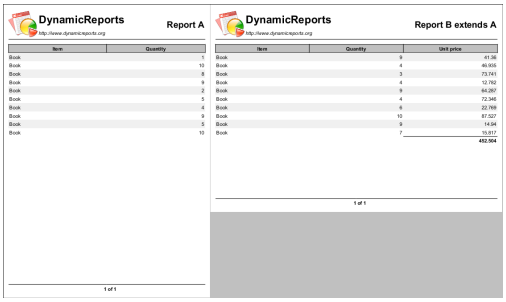
```
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DynamicPageWidthReport {

        public DynamicPageWidthReport() {
                build();
        }

        private void build() {

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .ignorePageWidth()
                                        .columns(
                                                        col.column("ID", "id", type.
↪integerType()),
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()),
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
```

(continues on next page)

(continued from previous page)

```
                                                col.column("Order date",
↪"orderdate", type.dateYearToFractionType()),
                                                col.column("Order year",
↪"orderdate", type.dateYearType()),
                                                col.column("Order month",
↪"orderdate", type.dateMonthType()),
                                                col.column("Order day",
↪"orderdate", type.dateDayType()))
                                        .title(Templates.createTitleComponent(
↪"DynamicPageWidth"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("id", "item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));
                dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
                dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new DynamicPageWidthReport();
        }
}
```

1.29.5 Inheritance




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

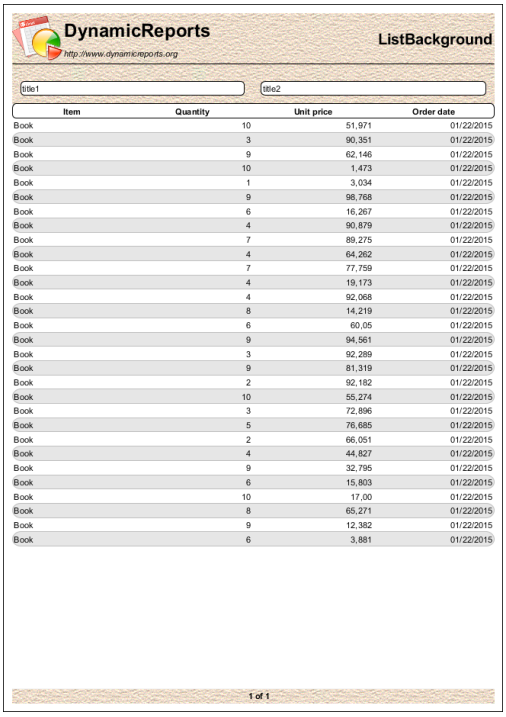
```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.jasper.builder.export.Exporters;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class InheritanceReport {

        public InheritanceReport() {
                build();
        }

        private void build() {
                try {
                        concatenatedReport()
                                        .concatenate(
                                                        new ReportA().report,
                                                        new ReportB().report)
                                        .toPdf(Exporters.pdfExporter("c:/report.pdf
↪"));
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class ReportA {
                protected JasperReportBuilder report = report();

                private ReportA() {
                        configure();
                }

```

(continues on next page)

(continued from previous page)

```
67                 protected void configure() {
68                         report
69                                         .setTemplate(Templates.reportTemplate)
70                                         .setPageFormat(PageType.A5)
71                                         .columns(
72                                                         col.column("Item", "item",␣
↪type.stringType()),
73                                                         col.column("Quantity",
↪"quantity", type.integerType()))
74                                         .title(Templates.
↪createTitleComponent(getTitle()))
75                                         .pageFooter(Templates.footerComponent)
76                                         .setDataSource(createDataSource());
77                 }
78
79                 protected String getTitle() {
80                         return "Report A";
81                 }
82         }
83
84         private class ReportB extends ReportA {
85
86                 @Override
87                 protected void configure() {
88                         super.configure();
89                         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column(
↪"Unit price", "unitprice", type.bigDecimalType());
90                         report
91                                         .setPageFormat(PageType.A5, PageOrientation.
↪LANDSCAPE)
92                                         .addColumn(unitPriceColumn)
93                                         .subtotalsAtSummary(sbt.sum(unitPriceColumn));
94                 }
95
96                 @Override
97                 protected String getTitle() {
98                         return "Report B extends A";
99                 }
100        }
101
102        private JRDataSource createDataSource() {
103                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
104                for (int i = 0; i < 10; i++) {
105                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
106                }
107                return dataSource;
108        }
109
110        public static void main(String[] args) {
111                new InheritanceReport();
112        }
113 }
```

1.29.6 List Background




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.math.BigDecimal;
import java.util.Date;

```

(continues on next page)

(continued from previous page)

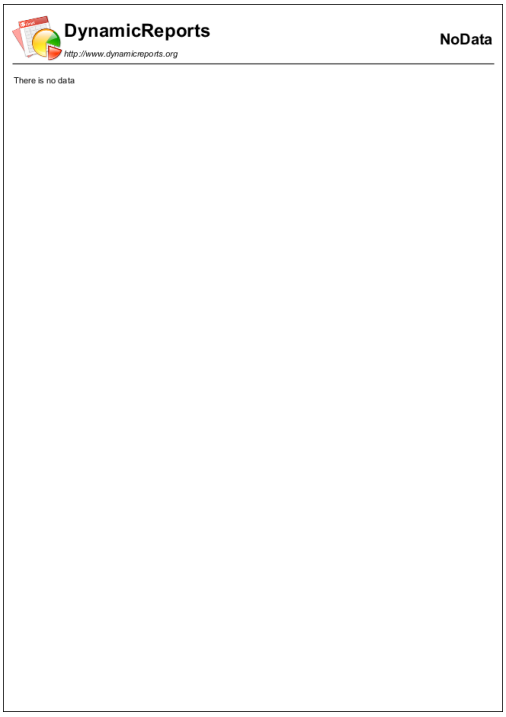
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
import net.sf.dynamicreports.report.builder.component.ImageBuilder;
import net.sf.dynamicreports.report.builder.component.RectangleBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.ImageScale;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ListBackgroundReport {

        public ListBackgroundReport() {
                build();
        }

        private void build() {
                StyleBuilder style1 = stl.style()
                                .setRadius(10)
                                .setBackgroundColor(new Color(230, 230, 230))
                                .setLinePen(stl.pen().setLineColor(Color.LIGHT_GRAY));
                StyleBuilder style2 = stl.style()
                                .setRadius(5);

                ImageBuilder background1 = cmp.image(Templates.class.getResource(
↪"images/background.gif"))
                                .setImageScale(ImageScale.CLIP)
                                .setStyle(style1);
                RectangleBuilder background2 = cmp.rectangle()
                                .setStyle(style2);
                RectangleBuilder background3 = cmp.rectangle()
                                .setStyle(style1)
                                .setPrintWhenExpression(exp.printInOddRow());

                HorizontalListBuilder title1 = cmp.horizontalList()
                                .add(cmp.text("title1"))
                                .setBackgroundComponent(background2);
                HorizontalListBuilder title2 = cmp.horizontalList()
                                .add(cmp.text("title2"))
                                .setBackgroundComponent(background2);
                HorizontalListBuilder title = cmp.horizontalList()
                                .add(title1, cmp.horizontalGap(20), title2)
                                .setStyle(stl.style(10));

                try {
                        report()
                                        .setColumnStyle(Templates.columnStyle)
                                        .setColumnTitleStyle(Templates.
↪boldCenteredStyle)
                                        .setTitleBackgroundComponent(background1)
                                        .
↪setColumnHeaderBackgroundComponent(background2)
                                        .setPageFooterBackgroundComponent(background1)
                                        .setDetailBackgroundComponent(background3)
```

(continues on next page)

(continued from previous page)

```
85                                          .columns(
86                                                  col.column("Item", "item",␣
↪type.stringType()),
87                                                  col.column("Quantity",
↪"quantity", type.integerType()),
88                                                  col.column("Unit price",
↪"unitprice", type.bigDecimalType()),
89                                                  col.column("Order date",
↪"orderdate", type.dateType()))
90                                          .title(
91                                                  Templates.
↪createTitleComponent("ListBackground"),
92                                                  title)
93                                          .pageFooter(cmp.pageXofY().setStyle(Templates.
↪boldCenteredStyle))
94                                          .setDataSource(createDataSource())
95                                          .show();
96                  } catch (DREException e) {
97                          e.printStackTrace();
98                  }
99          }
100
101         private JRDataSource createDataSource() {
102                 DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
103                 for (int i = 0; i < 30; i++) {
104                         dataSource.add("Book", new Date(), (int) (Math.random() * 10)␣
↪+ 1, new BigDecimal(Math.random() * 100 + 1));
105                 }
106                 return dataSource;
107         }
108
109         public static void main(String[] args) {
110                 new ListBackgroundReport();
111         }
112 }
```

1.29.7 No Data



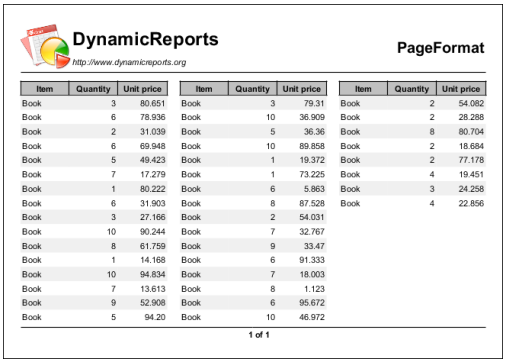

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class NoDataReport {

        public NoDataReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, quantityColumn, priceColumn)
                                        .noData(Templates.createTitleComponent("NoData"), cmp.text("There is no data"))
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                // dataSource.add("Book", 10, new BigDecimal(100));
                return dataSource;
        }

        public static void main(String[] args) {
                new NoDataReport();
        }
}
```

1.29.8 Page Format



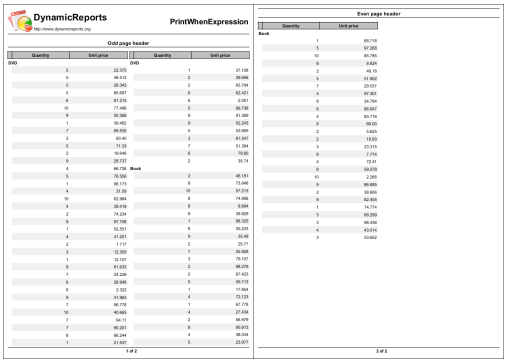

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PageFormatReport {

        public PageFormatReport() {
```

(continues on next page)

(continued from previous page)

```
43                  build();
44          }
45
46          private void build() {
47                  TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
48                  TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
49                  TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
↪"unitprice", type.bigDecimalType());
50
51                  try {
52                          report()
53                                          .setTemplate(Templates.reportTemplate)
54                                          .setPageFormat(PageType.A5, PageOrientation.
↪LANDSCAPE)
55                                          .setPageColumnsPerPage(3)
56                                          .setPageColumnSpace(15)
57                                          .setPageMargin(margin(20))
58                                          .columns(
59                                                          itemColumn, quantityColumn,␣
↪priceColumn)
60                                          .title(Templates.createTitleComponent(
↪"PageFormat"))
61                                          .pageFooter(Templates.footerComponent)
62                                          .setDataSource(createDataSource())
63                                          .show();
64                  } catch (DRException e) {
65                          e.printStackTrace();
66                  }
67          }
68
69          private JRDataSource createDataSource() {
70                  DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
71                  for (int i = 0; i < 40; i++) {
72                          dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
73                  }
74                  return dataSource;
75          }
76
77          public static void main(String[] args) {
78                  new PageFormatReport();
79          }
80  }
```

1.29.9 Print When Expression




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.constant.Constants;
import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
```

(continues on next page)

(continued from previous page)

```
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PrintWhenExpressionReport {

        public PrintWhenExpressionReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());

                ColumnGroupBuilder itemGroup = grp.group("itemGroup", itemColumn)
                                .setHeaderLayout(GroupHeaderLayout.EMPTY);

                TextFieldBuilder<String> groupHeader = cmp.text(new GroupHeaderExpression())
                                .setStyle(Templates.groupStyle)
                                .setPrintWhenExpression(new PrintGroupHeaderExpression())
                                .removeLineWhenBlank();

                VerticalListBuilder oddPageHeader = cmp.verticalList()
                                .add(
                                                cmp.text("Odd page header").setStyle(Templates.bold12CenteredStyle),
                                                cmp.line())
                                .setPrintWhenExpression(new PrintInOddPageExpression())
                                .removeLineWhenBlank();

                VerticalListBuilder evenPageHeader = cmp.verticalList()
                                .add(
                                                cmp.line(),
                                                cmp.text("Even page header").setStyle(Templates.bold12CenteredStyle),
                                                cmp.line())
                                .setPrintWhenExpression(new PrintInEvenPageExpression())
                                .removeLineWhenBlank();

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setPageColumnsPerPage(2)
                                        .setPageColumnSpace(5)
                                        .columns(
                                                        itemColumn,
                                                        col.column("Quantity", "quantity", type.integerType()),
                                                        col.column("Unit price", "unitprice", type.bigDecimalType()))
                                        .groupBy(itemGroup)
                                        .title(Templates.createTitleComponent("PrintWhenExpression"))
                                        .detailHeader(
                                                        cmp.columnBreak().setPrintWhenExpression(new PrintGroupHeaderColumnBreakExpression()),
```

(continues on next page)

(continued from previous page)

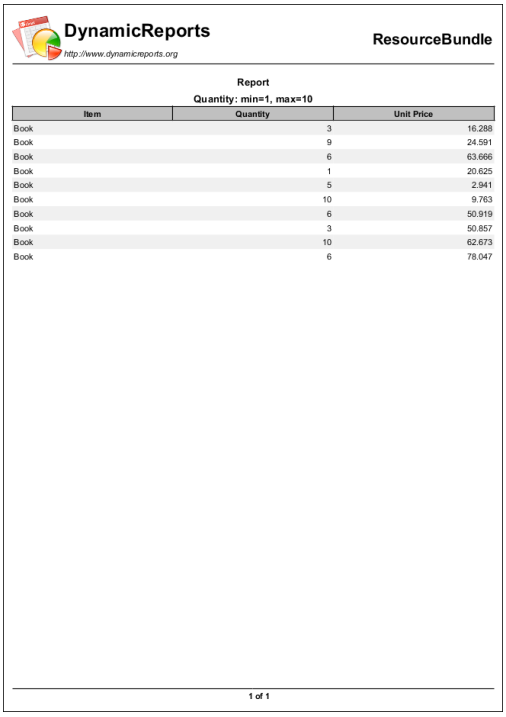
```
                                                        groupHeader)
                                        .pageHeader(
                                                        oddPageHeader, evenPageHeader,
↪ cmp.verticalGap(10))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 50; i++) {
                        dataSource.add("DVD", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                for (int i = 0; i < 50; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public class GroupHeaderExpression extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪UID;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        return reportParameters.getValue("item");
                }
        }

        public class PrintGroupHeaderExpression extends AbstractSimpleExpression
↪<Boolean> {
                private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪UID;

                @Override
                public Boolean evaluate(ReportParameters reportParameters) {
                        return reportParameters.getColumnRowNumber() == 1 ||␣
↪reportParameters.getGroupCount("itemGroup") == 1;
                }
        }

        public class PrintGroupHeaderColumnBreakExpression extends␣
↪AbstractSimpleExpression<Boolean> {
                private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪UID;

                @Override
                public Boolean evaluate(ReportParameters reportParameters) {
                        return reportParameters.getColumnRowNumber() == 1 &&␣
↪reportParameters.getGroupCount("itemGroup") != 1;
```

(continues on next page)

(continued from previous page)

```
136                 }
137         }
138
139         public class PrintInOddPageExpression extends AbstractSimpleExpression
    ↪<Boolean> {
140                 private static final long serialVersionUID = Constants.SERIAL_VERSION_
    ↪UID;
141
142                 @Override
143                 public Boolean evaluate(ReportParameters reportParameters) {
144                         return reportParameters.getPageNumber().doubleValue() % 2 !=␣
    ↪0;
145                 }
146         }
147
148         public class PrintInEvenPageExpression extends AbstractSimpleExpression
    ↪<Boolean> {
149                 private static final long serialVersionUID = Constants.SERIAL_VERSION_
    ↪UID;
150
151                 @Override
152                 public Boolean evaluate(ReportParameters reportParameters) {
153                         return reportParameters.getPageNumber().doubleValue() % 2 ==␣
    ↪0;
154                 }
155         }
156
157         public static void main(String[] args) {
158                 new PrintWhenExpressionReport();
159         }
160 }
```

1.29.10 Resource Bundle


DynamicReports
http://www.dynamicreports.org

ResourceBundle

Report

Quantity: min=1, max=10

Item	Quantity	Unit Price
Book	3	16.288
Book	9	24.591
Book	6	63.666
Book	1	20.625
Book	5	2.941
Book	10	9.763
Book	6	50.919
Book	3	50.857
Book	10	62.673
Book	6	78.047

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.List;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

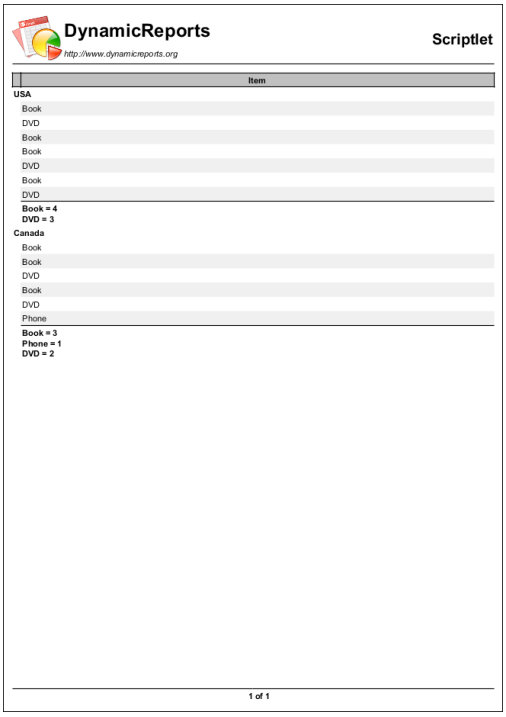
```
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.Constants;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ResourceBundleReport {

        public ResourceBundleReport() {
                build();
        }

        private void build() {
                try {
                        TextColumnBuilder<String> itemColumn = col.column("item", type.stringType())
                                        .setTitle(exp.message("item_title"));
                        TextColumnBuilder<Integer> quantityColumn = col.column("quantity", type.integerType())
                                        .setTitle(exp.message("quantity_title"));
                        TextColumnBuilder<BigDecimal> priceColumn = col.column("unitprice", type.bigDecimalType())
                                        .setTitle(exp.message("unitprice_title"));

                        TextFieldBuilder<String> title = cmp.text(exp.message("report_title"))
                                        .setStyle(Templates.bold12CenteredStyle);
                        TextFieldBuilder<String> subtitle = cmp.text(new SubtitleExpression())
                                        .setStyle(Templates.bold12CenteredStyle);

                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setResourceBundle(ResourceBundleReport.class.getName())
                                        .columns(
                                                        itemColumn, quantityColumn, priceColumn)
                                        .title(
                                                        Templates.createTitleComponent("ResourceBundle"),
                                                        title, subtitle)
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

```

(continues on next page)

(continued from previous page)

```
80          private JRDataSource createDataSource() {
81                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
82                  for (int i = 0; i < 10; i++) {
83                          dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
    ↪BigDecimal(Math.random() * 100 + 1));
84                  }
85                  return dataSource;
86          }
87
88          public static void main(String[] args) {
89                  new ResourceBundleReport();
90          }
91
92          public class SubtitleExpression extends AbstractComplexExpression<String> {
93                  private static final long serialVersionUID = Constants.SERIAL_VERSION_
    ↪UID;
94
95                  public SubtitleExpression() {
96                          addExpression(variable("quantity", Integer.class, Calculation.
    ↪LOWEST));
97                          addExpression(variable("quantity", Integer.class, Calculation.
    ↪HIGHEST));
98                  }
99
100                 @Override
101                 public String evaluate(List<?> values, ReportParameters␣
    ↪reportParameters) {
102                         return reportParameters.getMessage("report_subtitle", new␣
    ↪Object[] { values.get(0), values.get(1) });
103                 }
104         }
105 }
```

1.29.11 Scriptlet




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.util.HashMap;
import java.util.Map;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

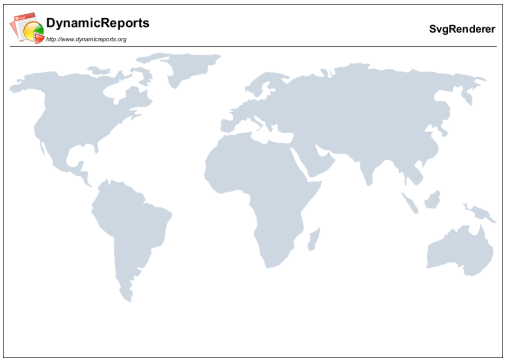
```
import net.sf.dynamicreports.report.base.AbstractScriptlet;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

import org.apache.commons.lang3.StringUtils;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ScriptletReport {
        private Map<String, Integer> itemsCount;

        public ScriptletReport() {
                build();
        }

        private void build() {
                itemsCount = new HashMap<String, Integer>();

                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType());
                CustomSubtotalBuilder<String> itemSbt = sbt.customValue(new ItemSubtotal(), itemColumn);
                CustomGroupBuilder group = grp.group("country", String.class);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .scriptlets(new ReportScriptlet())
                                        .columns(itemColumn)
                                        .groupBy(group)
                                        .subtotalsAtGroupFooter(group, itemSbt)
                                        .title(Templates.createTitleComponent("Scriptlet"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class ItemSubtotal extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        String result = "";
                        for (String item : itemsCount.keySet()) {
                                result += item + " = " + itemsCount.get(item) + "\n";
                        }
```

(continues on next page)

(continued from previous page)

```
85                          return StringUtils.removeEnd(result, "\n");
86                  }
87          }
88
89          private class ReportScriptlet extends AbstractScriptlet {
90
91                  @Override
92                  public void afterDetailEval(ReportParameters reportParameters) {
93                          super.afterDetailEval(reportParameters);
94                          String item = reportParameters.getValue("item");
95                          Integer count;
96                          if (itemsCount.containsKey(item)) {
97                                  count = itemsCount.get(item);
98                          } else {
99                                  count = 0;
100                         }
101                         itemsCount.put(item, ++count);
102                 }
103
104                 @Override
105                 public void afterGroupInit(String groupName, ReportParameters␣
    ↪reportParameters) {
106                         super.afterGroupInit(groupName, reportParameters);
107                         itemsCount.clear();
108                 }
109         }
110
111         private JRDataSource createDataSource() {
112                 DRDataSource dataSource = new DRDataSource("country", "item");
113                 dataSource.add("USA", "Book");
114                 dataSource.add("USA", "DVD");
115                 dataSource.add("USA", "Book");
116                 dataSource.add("USA", "Book");
117                 dataSource.add("USA", "DVD");
118                 dataSource.add("USA", "Book");
119                 dataSource.add("USA", "DVD");
120
121                 dataSource.add("Canada", "Book");
122                 dataSource.add("Canada", "Book");
123                 dataSource.add("Canada", "DVD");
124                 dataSource.add("Canada", "Book");
125                 dataSource.add("Canada", "DVD");
126                 dataSource.add("Canada", "Phone");
127                 return dataSource;
128         }
129
130         public static void main(String[] args) {
131                 new ScriptletReport();
132         }
133 }
```

1.29.12 Svg Renderer



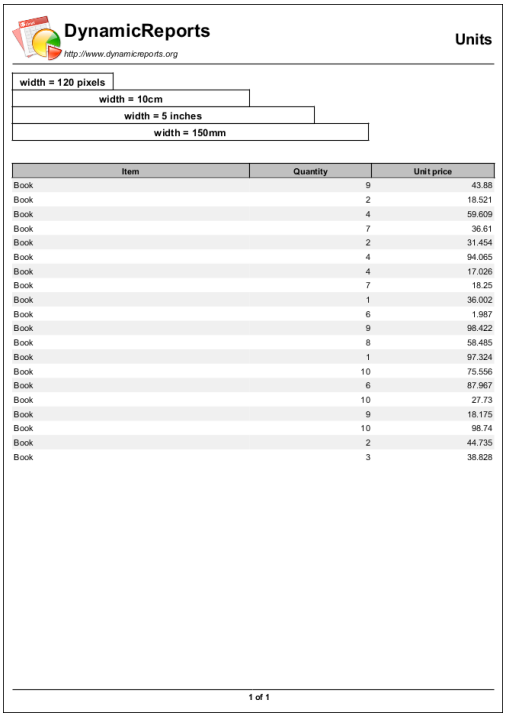

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.constant.PageOrientation;
import net.sf.dynamicreports.report.constant.PageType;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.util.JRLoader;
import net.sf.jasperreports.renderers.Renderable;
import net.sf.jasperreports.renderers.SimpleDataRenderer;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class SvgRendererReport {

        public SvgRendererReport() {
                build();
```

(continues on next page)

(continued from previous page)

```
43          }
44
45          private void build() {
46                  try {
47                          Renderable image = new SimpleDataRenderer(JRLoader.
    ↪loadBytes(Templates.class.getResource("images/map.svg")), null);
48
49                          report()
50                                          .setTemplate(Templates.reportTemplate)
51                                          .setPageFormat(PageType.A4, PageOrientation.
    ↪LANDSCAPE)
52                                          .title(
53                                                          Templates.
    ↪createTitleComponent("SvgRenderer"),
54                                                          cmp.image(image).
    ↪setHeight(500))
55                                          .show();
56                  } catch (DRException e) {
57                          e.printStackTrace();
58                  } catch (JRException e) {
59                          e.printStackTrace();
60                  }
61          }
62
63          public static void main(String[] args) {
64                  new SvgRendererReport();
65          }
66  }
```

1.29.13 Units



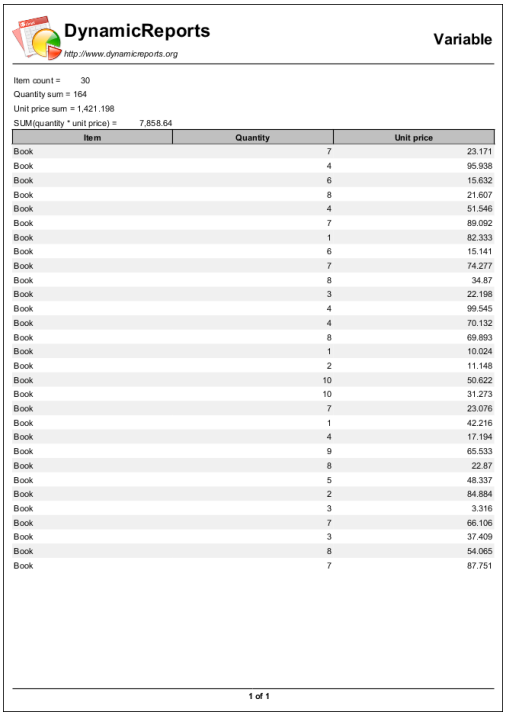
```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class UnitsReport {

        public UnitsReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType())
                                .setFixedWidth(cm(10));
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
↪"unitprice", type.bigDecimalType());

                StyleBuilder style = stl.style(Templates.bold12CenteredStyle)
                                .setBorder(stl.pen1Point());
                TextFieldBuilder<String> text1 = cmp.text("width = 120 pixels")
```

(continues on next page)

(continued from previous page)

```
                                .setFixedWidth(120)
                                .setStyle(style);
                TextFieldBuilder<String> text2 = cmp.text("width = 10cm")
                                .setFixedWidth(cm(10))
                                .setStyle(style);
                TextFieldBuilder<String> text3 = cmp.text("width = 5 inches")
                                .setFixedWidth(inch(5))
                                .setStyle(style);
                TextFieldBuilder<String> text4 = cmp.text("width = 150mm")
                                .setFixedWidth(mm(150))
                                .setStyle(style);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, quantityColumn,␣
↪priceColumn)
                                        .title(
                                                        Templates.
↪createTitleComponent("Units"),
                                                        text1, text2, text3, text4,␣
↪cmp.verticalGap(cm(1)))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 20; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
                new UnitsReport();
        }
}
```

1.29.14 Variable



DynamicReports
http://www.dynamicreports.org

Variable

Item count = 30
Quantity sum = 164
Unit price sum = 1,421.198
SUM(quantity * unit price) = 7,858.64

Item	Quantity	Unit price
Book	7	23.171
Book	4	95.938
Book	6	15.632
Book	8	21.607
Book	4	51.546
Book	7	89.092
Book	1	82.333
Book	6	15.141
Book	7	74.277
Book	8	34.87
Book	3	22.198
Book	4	99.545
Book	4	70.132
Book	8	69.893
Book	1	10.024
Book	2	11.148
Book	10	50.622
Book	10	31.273
Book	7	23.076
Book	1	42.216
Book	4	17.194
Book	9	65.533
Book	8	22.87
Book	5	48.337
Book	2	84.884
Book	3	3.316
Book	7	66.106
Book	3	37.409
Book	8	54.065
Book	7	87.751

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.List;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.VariableBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.Evaluation;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class VariableReport {

        public VariableReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                VariableBuilder<Integer> itemCount = variable(itemColumn, Calculation.
↪COUNT);
                VariableBuilder<Integer> quantitySum = variable("quantitySum",␣
↪quantityColumn, Calculation.SUM);
                VariableBuilder<Integer> priceSum = variable(new␣
↪PriceExpression(quantityColumn, unitPriceColumn), Calculation.SUM);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .variables(quantitySum)
                                        .columns(
                                                        itemColumn, quantityColumn,␣
↪unitPriceColumn)
                                        .title(
                                                        Templates.
↪createTitleComponent("Variable"),
                                                        cmp.horizontalList(cmp.text(
↪"Item count =").setFixedWidth(80), cmp.text(itemCount)),
                                                        cmp.text(new␣
↪QuantitySumTextExpression()).setEvaluationTime(Evaluation.REPORT),
                                                        cmp.text(new␣
↪UnitPriceSumTextExpression(unitPriceColumn)),
                                                        cmp.horizontalList(cmp.text(
↪"SUM(quantity * unit price) =").setFixedWidth(150), cmp.text(priceSum).setPattern("
↪#,###.00")))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
```

(continues on next page)

(continued from previous page)

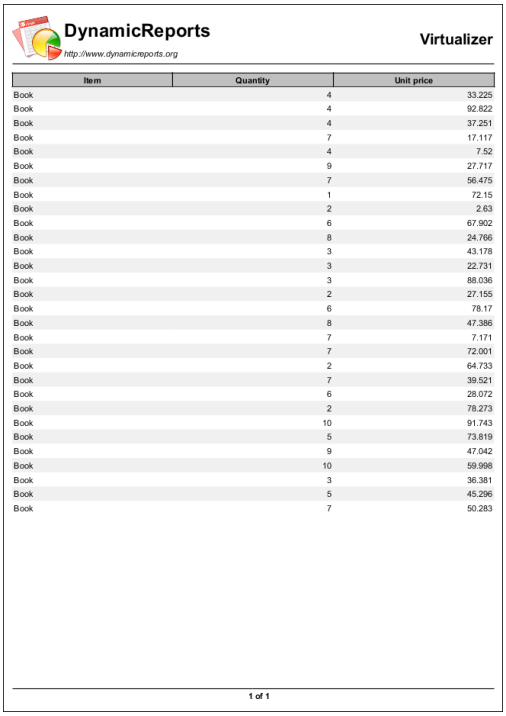
```
75                  } catch (DRException e) {
76                          e.printStackTrace();
77                  }
78          }
79
80          private JRDataSource createDataSource() {
81                  DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
82                  for (int i = 0; i < 30; i++) {
83                          dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
84                  }
85                  return dataSource;
86          }
87
88          public static void main(String[] args) {
89                  new VariableReport();
90          }
91
92          private class QuantitySumTextExpression extends AbstractSimpleExpression
↪<String> {
93                  private static final long serialVersionUID = 1L;
94
95                  @Override
96                  public String evaluate(ReportParameters reportParameters) {
97                          Integer quantitySum = reportParameters.getValue("quantitySum
↪");
98                          return "Quantity sum = " + quantitySum;
99                  }
100         }
101
102         private class UnitPriceSumTextExpression extends AbstractComplexExpression
↪<String> {
103                 private static final long serialVersionUID = 1L;
104
105                 public UnitPriceSumTextExpression(TextColumnBuilder<BigDecimal>␣
↪unitPriceColumn) {
106                         addExpression(variable(unitPriceColumn, Calculation.SUM));
107                 }
108
109                 @Override
110                 public String evaluate(List<?> values, ReportParameters␣
↪reportParameters) {
111                         BigDecimal unitPriceSum = (BigDecimal) values.get(0);
112                         return "Unit price sum = " + type.bigDecimalType().
↪valueToString(unitPriceSum, reportParameters.getLocale());
113                 }
114         }
115
116         private class PriceExpression extends AbstractSimpleExpression<BigDecimal> {
117                 private static final long serialVersionUID = 1L;
118
119                 private TextColumnBuilder<Integer> quantityColumn;
120                 private TextColumnBuilder<BigDecimal> unitPriceColumn;
121
122                 public PriceExpression(TextColumnBuilder<Integer> quantityColumn,␣
↪TextColumnBuilder<BigDecimal> unitPriceColumn) {
```

(continues on next page)

(continued from previous page)

```
                        this.quantityColumn = quantityColumn;
                        this.unitPriceColumn = unitPriceColumn;
                }

                @Override
                public BigDecimal evaluate(ReportParameters reportParameters) {
                        Integer quantity = reportParameters.getValue(quantityColumn);
                        BigDecimal unitPrice = reportParameters.
↪getValue(unitPriceColumn);
                        return unitPrice.multiply(new BigDecimal(quantity));
                }
        }
}
```

1.29.15 Virtualizer



DynamicReports
http://www.dynamicreports.org

Virtualizer

Item	Quantity	Unit price
Book	4	33.225
Book	4	92.822
Book	4	37.251
Book	7	17.117
Book	4	7.52
Book	9	27.717
Book	7	56.475
Book	1	72.15
Book	2	2.63
Book	6	67.902
Book	8	24.766
Book	3	43.178
Book	3	22.731
Book	3	88.036
Book	2	27.155
Book	6	78.17
Book	8	47.386
Book	7	7.171
Book	7	72.001
Book	2	64.733
Book	7	39.521
Book	6	28.072
Book	2	78.273
Book	10	91.743
Book	5	73.819
Book	9	47.042
Book	10	59.998
Book	3	36.381
Book	5	45.296
Book	7	50.283

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
```

(continues on next page)

(continued from previous page)

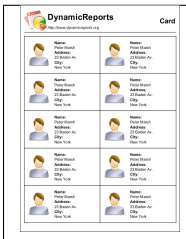
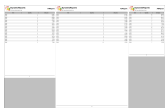
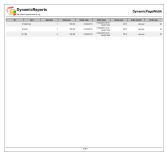
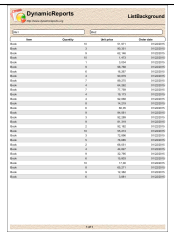
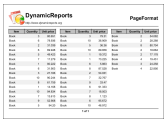
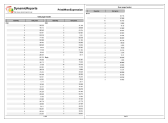
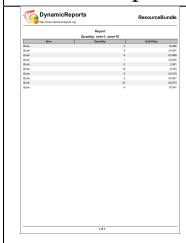
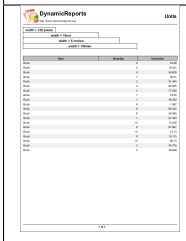
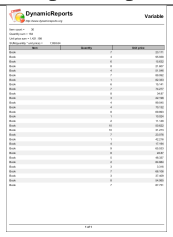
```
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.miscellaneous;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.fill.JRFileVirtualizer;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class VirtualizerReport {

        public VirtualizerReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
↪"unitprice", type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, quantityColumn,␣
↪priceColumn)
                                        .title(Templates.createTitleComponent(
↪"Virtualizer"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .setVirtualizer(new JRFileVirtualizer(2))
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
```

(continues on next page)

(continued from previous page)

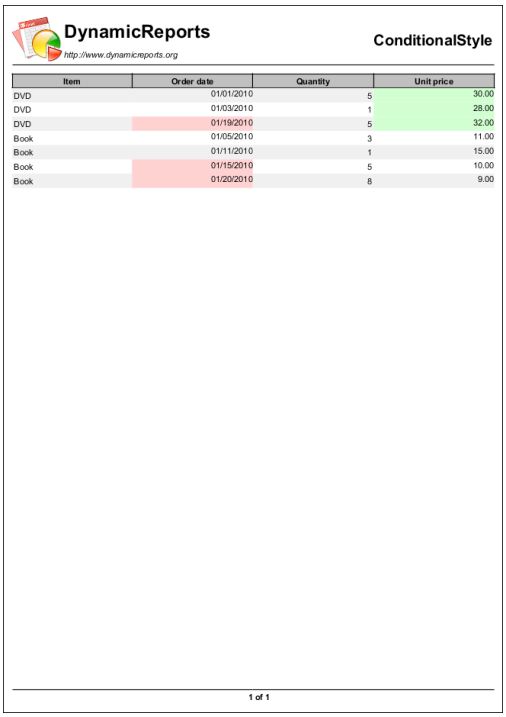
```
66                  DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
67                  for (int i = 0; i < 30; i++) {
68                          // for (int i = 0; i < 100000; i++) {
69                          dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
70                  }
71                  return dataSource;
72          }
73
74          public static void main(String[] args) {
75                  new VirtualizerReport();
76          }
77  }
```

Table 14: Miscellaneous Examples

 CardReport	 ConcatenatedReport1	 ConcatenatedReport2
 DynamicPageWidthReport	 InheritanceReport	 ListBackgroundReport
 NoDataReport	 PageFormatReport	 PrintWhenExpressionReport
 ResourceBundleReport	 ScriptletReport	 SvgRendererReport
 UnitsReport	 VariableReport	 VirtualizerReport

1.30 Style

1.30.1 Conditional Style




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.style;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
```

(continues on next page)

(continued from previous page)

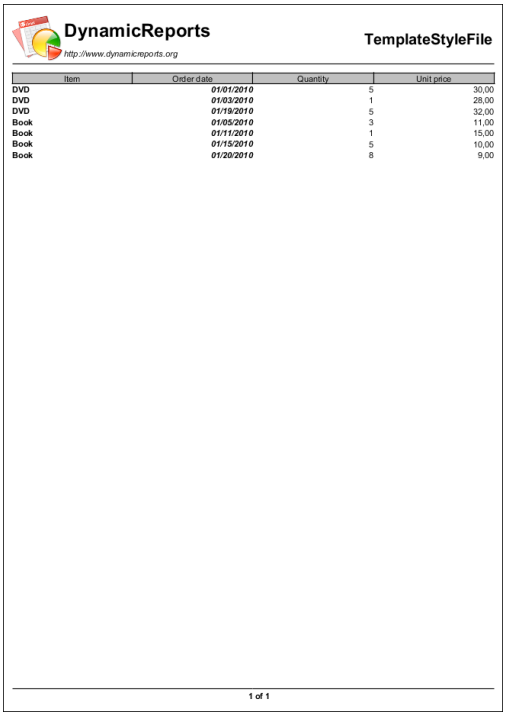
```
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.ConditionalStyleBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class ConditionalStyleReport {

        public ConditionalStyleReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
                TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                ConditionalStyleBuilder condition1 = stl.conditionalStyle(new
↪OrderDateConditionExpression())
                                .setBackgroundColor(new Color(255, 210, 210));
                ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.
↪greater(unitPriceColumn, 20))
                                .setBackgroundColor(new Color(210, 255, 210));

                StyleBuilder orderDateStyle = stl.style()
                                .conditionalStyles(condition1);
                orderDateColumn.setStyle(orderDateStyle);

                StyleBuilder unitPriceStyle = stl.style()
                                .conditionalStyles(condition2);
                unitPriceColumn.setStyle(unitPriceStyle);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, orderDateColumn,
↪quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent(
↪"ConditionalStyle"))
                                        .pageFooter(Templates.footerComponent)
```

(continues on next page)

(continued from previous page)

```
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
                dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
                dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
                dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
                dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
                dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));
                dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new ConditionalStyleReport();
        }

        private class OrderDateConditionExpression extends AbstractSimpleExpression
↪<Boolean> {
                private static final long serialVersionUID = 1L;

                @Override
                public Boolean evaluate(ReportParameters reportParameters) {
                        Date orderDate = reportParameters.getValue("orderdate");
                        Integer quantity = reportParameters.getValue("quantity");
                        return orderDate.after(toDate(2010, 1, 10)) && quantity > 1;
                }
        }
}
```

1.30.2 Template Style File




```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.style;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

```

(continues on next page)

(continued from previous page)

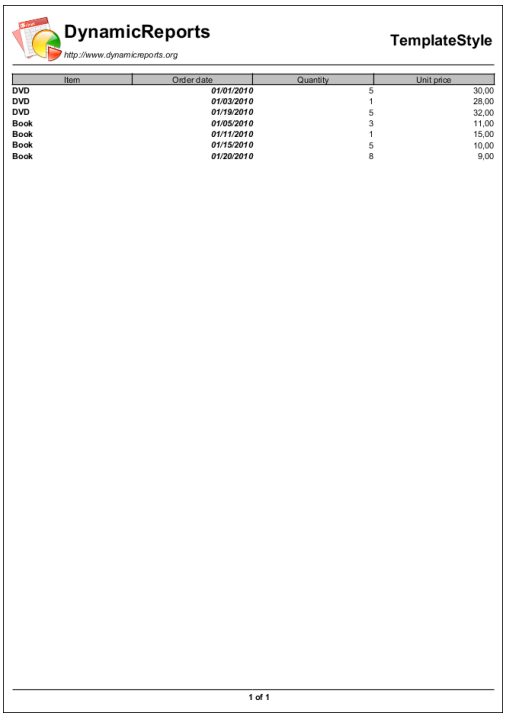
```
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TemplateStyleFileReport {

        public TemplateStyleFileReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item", type.stringType())
                                .setStyle(stl.templateStyle("style1"));
                TextColumnBuilder<Date> orderDateColumn = col.column("Order date", "orderdate", type.dateType())
                                .setStyle(stl.templateStyle("style2"));
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice", type.bigDecimalType());

                try {
                        report()
                                        .templateStyles(stl.loadStyles(TemplateStyleFileReport.class.getResource("TemplateStyles.jrtx")))
                                        .setColumnStyle(stl.templateStyle("columnStyle"))
                                        .setColumnTitleStyle(stl.templateStyle("columnTitleStyle"))
                                        .columns(
                                                        itemColumn, orderDateColumn, quantityColumn, unitPriceColumn)
                                        .title(Templates.createTitleComponent("TemplateStyleFile"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate", "quantity", "unitprice");
                dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
                dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
                dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
                dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
                dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
                dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new TemplateStyleFileReport();
        }
}
```

1.30.3 Template Style




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.style;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.ReportTemplateBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TemplateStyleReport {

        public TemplateStyleReport() {
                build();
        }

        private void build() {
                StyleBuilder style1 = stl.style()
                                .setName("style1")
                                .bold();
                StyleBuilder style2 = stl.style(style1)
                                .setName("style2")
                                .italic();
                StyleBuilder columnStyle = stl.style()
                                .setName("columnStyle")
                                .setVerticalTextAlignment(VerticalTextAlignment.
↪MIDDLE);
                StyleBuilder columnTitleStyle = stl.style(columnStyle)
                                .setName("columnTitleStyle")
                                .setBorder(stl.pen1Point())
                                .setHorizontalTextAlignment(HorizontalTextAlignment.
↪CENTER)
                                .setBackgroundColor(Color.LIGHT_GRAY);
```

(continues on next page)

(continued from previous page)

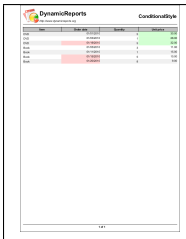
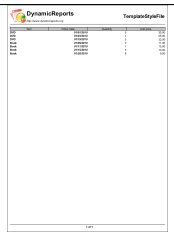
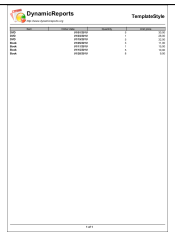
```
66                 ReportTemplateBuilder template = template()
67                                 .templateStyles(style1, style2, columnStyle,␣
↪columnTitleStyle);
68
69                 TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType())
70                                 .setStyle(stl.templateStyle("style1"));
71                 TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateType())
72                                 .setStyle(stl.templateStyle("style2"));
73                 TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
74                 TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
75
76                 try {
77                         report()
78                                         .setTemplate(template)
79                                         .setColumnStyle(stl.templateStyle("columnStyle
↪"))
80                                         .setColumnTitleStyle(stl.templateStyle(
↪"columnTitleStyle"))
81                                         .columns(
82                                                         itemColumn, orderDateColumn,␣
↪quantityColumn, unitPriceColumn)
83                                         .title(Templates.createTitleComponent(
↪"TemplateStyle"))
84                                         .pageFooter(Templates.footerComponent)
85                                         .setDataSource(createDataSource())
86                                         .show();
87                 } catch (DRException e) {
88                         e.printStackTrace();
89                 }
90         }
91
92         private JRDataSource createDataSource() {
93                 DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
94                 dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
95                 dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
96                 dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
97                 dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
98                 dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
99                 dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));
100                dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
101                return dataSource;
102        }
103
104        private Date toDate(int year, int month, int day) {
105                Calendar c = Calendar.getInstance();
106                c.set(Calendar.YEAR, year);
107                c.set(Calendar.MONTH, month - 1);
108                c.set(Calendar.DAY_OF_MONTH, day);
109                return c.getTime();
110        }
111
112        public static void main(String[] args) {
```

(continues on next page)

(continued from previous page)

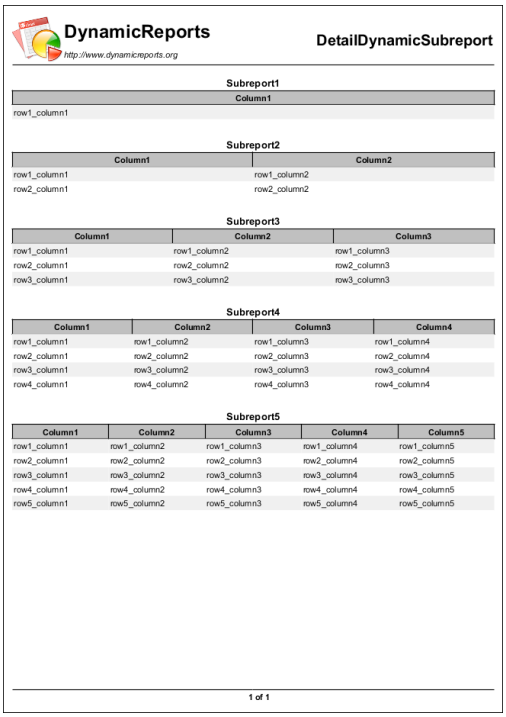
```
                new TemplateStyleReport();
        }
}
```

Table 15: Style Examples

		
ConditionalStyleReport	TemplateStyleFileReport	TemplateStyleReport

1.31 Subreport

1.31.1 Detail Dynamic Subreport



DynamicReports
http://www.dynamicreports.org

DetailDynamicSubreport

Subreport1

Column1
row1_column1

Subreport2

Column1	Column2
row1_column1	row1_column2
row2_column1	row2_column2

Subreport3

Column1	Column2	Column3
row1_column1	row1_column2	row1_column3
row2_column1	row2_column2	row2_column3
row3_column1	row3_column2	row3_column3

Subreport4

Column1	Column2	Column3	Column4
row1_column1	row1_column2	row1_column3	row1_column4
row2_column1	row2_column2	row2_column3	row2_column4
row3_column1	row3_column2	row3_column3	row3_column4
row4_column1	row4_column2	row4_column3	row4_column4

Subreport5

Column1	Column2	Column3	Column4	Column5
row1_column1	row1_column2	row1_column3	row1_column4	row1_column5
row2_column1	row2_column2	row2_column3	row2_column4	row2_column5
row3_column1	row3_column2	row3_column3	row3_column4	row3_column5
row4_column1	row4_column2	row4_column3	row4_column4	row4_column5
row5_column1	row5_column2	row5_column3	row5_column4	row5_column5

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
```

(continues on next page)

(continued from previous page)

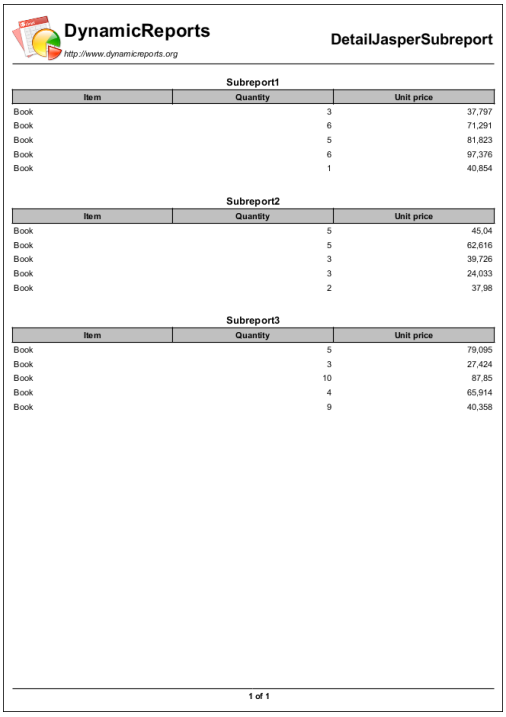
```
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subreport;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;
import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JREmptyDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DetailDynamicSubreport {

        public DetailDynamicSubreport() {
                build();
        }

        private void build() {
                SubreportBuilder subreport = cmp.subreport(new SubreportExpression())
                                .setDataSource(new SubreportDataSourceExpression());

                try {
                        report()
                                        .title(Templates.createTitleComponent(
↪"DetailDynamicSubreport"))
                                        .detail(
                                                        subreport,
                                                        cmp.verticalGap(20))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                return new JREmptyDataSource(5);
```

(continues on next page)

(continued from previous page)

```
65          }
66
67          private class SubreportExpression extends AbstractSimpleExpression
    ↪<JasperReportBuilder> {
68                  private static final long serialVersionUID = 1L;
69
70                  @Override
71                  public JasperReportBuilder evaluate(ReportParameters␣
    ↪reportParameters) {
72                          int masterRowNumber = reportParameters.getReportRowNumber();
73                          JasperReportBuilder report = report();
74                          report
75                                          .setTemplate(Templates.reportTemplate)
76                                          .title(cmp.text("Subreport" +␣
    ↪masterRowNumber).setStyle(Templates.bold12CenteredStyle));
77
78                          for (int i = 1; i <= masterRowNumber; i++) {
79                                  report.addColumn(col.column("Column" + i, "column" +␣
    ↪i, type.stringType()));
80                          }
81
82                          return report;
83                  }
84          }
85
86          private class SubreportDataSourceExpression extends AbstractSimpleExpression
    ↪<JRDataSource> {
87                  private static final long serialVersionUID = 1L;
88
89                  @Override
90                  public JRDataSource evaluate(ReportParameters reportParameters) {
91                          int masterRowNumber = reportParameters.getReportRowNumber();
92                          String[] columns = new String[masterRowNumber];
93                          for (int i = 1; i <= masterRowNumber; i++) {
94                                  columns[i - 1] = "column" + i;
95                          }
96                          DRDataSource dataSource = new DRDataSource(columns);
97
98                          for (int i = 1; i <= masterRowNumber; i++) {
99                                  Object[] values = new Object[masterRowNumber];
100                                 for (int j = 1; j <= masterRowNumber; j++) {
101                                         values[j - 1] = "row" + i + "_column" + j;
102                                 }
103                                 dataSource.add(values);
104                         }
105
106                         return dataSource;
107                 }
108         }
109
110         public static void main(String[] args) {
111                 new DetailDynamicSubreport();
112         }
113 }
```

1.31.2 Detail Jasper Subreport



DynamicReports
http://www.dynamicreports.org

DetailJasperSubreport

Subreport1

Item	Quantity	Unit price
Book	3	37,797
Book	6	71,291
Book	5	81,823
Book	6	97,376
Book	1	40,854

Subreport2

Item	Quantity	Unit price
Book	5	45,04
Book	5	62,616
Book	3	39,726
Book	3	24,033
Book	2	37,98

Subreport3

Item	Quantity	Unit price
Book	5	79,095
Book	3	27,424
Book	10	87,85
Book	4	65,914
Book	9	40,358

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subreport;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

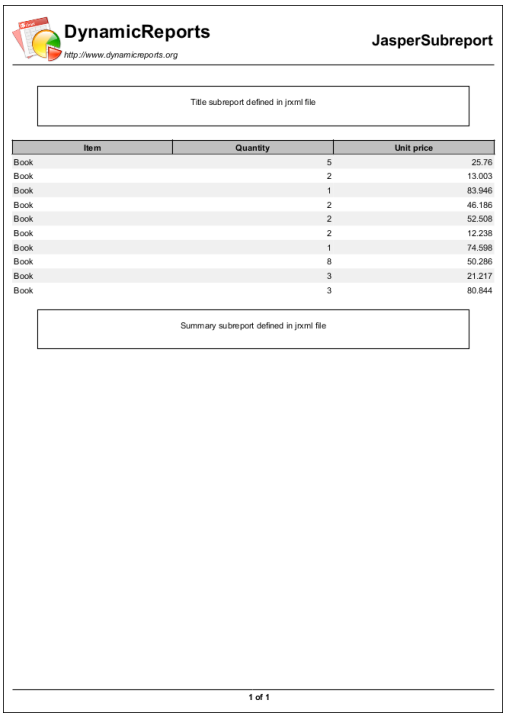
```
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JREmptyDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperReport;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class DetailJasperSubreport {

        public DetailJasperSubreport() {
                build();
        }

        private void build() {
                try {
                        SubreportBuilder subreport = cmp.
↪subreport(getJasperSubreport())
                                        .setDataSource(new␣
↪SubreportDataSourceExpression());

                        report()
                                        .title(Templates.createTitleComponent(
↪"DetailJasperSubreport"))
                                        .detail(
                                                        subreport,
                                                        cmp.verticalGap(20))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                } catch (JRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                return new JREmptyDataSource(3);
        }

        private JasperReport getJasperSubreport() throws JRException {
                InputStream is = DetailJasperSubreport.class.getResourceAsStream(
↪"subreport.jrxml");
                return JasperCompileManager.compileReport(is);
        }

        private class SubreportDataSourceExpression extends AbstractSimpleExpression
↪<JRDataSource> {
                private static final long serialVersionUID = 1L;

```

(continues on next page)

(continued from previous page)

```
                @Override
                public JRDataSource evaluate(ReportParameters reportParameters) {
                        DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
                        for (int i = 0; i < 5; i++) {
                                dataSource.add("Book", (int) (Math.random() * 10) + 1,
↪ new BigDecimal(Math.random() * 100 + 1));
                        }
                        return dataSource;
                }
        }

        public static void main(String[] args) {
                new DetailJasperSubreport();
        }
}
```

1.31.3 Jasper Subreport




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

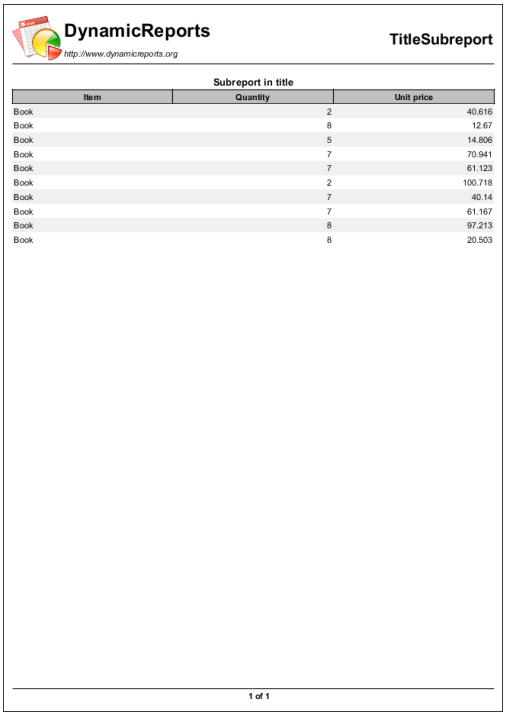
```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subreport;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.engine.util.JRLoader;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class JasperSubreport {

        public JasperSubreport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .title(
                                                        Templates.
↪createTitleComponent("JasperSubreport"),
                                                        cmp.
↪subreport(getJasperTitleSubreport()))
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .pageFooter(Templates.footerComponent)
                                        .summary(cmp.
↪subreport(getJasperSummarySubreport()))
                                        .setDataSource(createDataSource())
```

(continues on next page)

(continued from previous page)

```
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                } catch (JRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 10; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        private JasperReport getJasperTitleSubreport() throws JRException {
                InputStream is = JasperSubreport.class.getResourceAsStream(
↪"titlesubreport.jrxml");
                return JasperCompileManager.compileReport(is);
        }

        private JasperReport getJasperSummarySubreport() throws JRException {
                InputStream is = JasperSubreport.class.getResourceAsStream(
↪"summarysubreport.jasper");
                return (JasperReport) JRLoader.loadObject(is);
        }

        public static void main(String[] args) {
                new JasperSubreport();
        }
}
```

1.31.4 Title Subreport




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subreport;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
```

(continues on next page)

(continued from previous page)

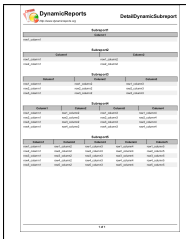
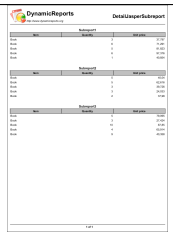
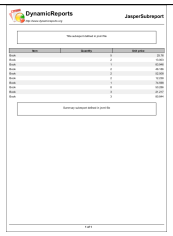
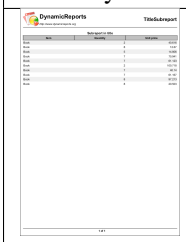
```
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TitleSubreport {

        public TitleSubreport() {
                build();
        }

        private void build() {
                try {
                        report()
                                        .title(
                                                        Templates.
↪createTitleComponent("TitleSubreport"),
                                                        cmp.
↪subreport(createSubreport()))
                                        .pageFooter(Templates.footerComponent)
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JasperReportBuilder createSubreport() {
                JasperReportBuilder report = report();
                report
                                .setTemplate(Templates.reportTemplate)
                                .title(cmp.text("Subreport in title").
↪setStyle(Templates.bold12CenteredStyle))
                                .columns(
                                                col.column("Item", "item", type.
↪stringType()),
                                                col.column("Quantity", "quantity",␣
↪type.integerType()),
                                                col.column("Unit price", "unitprice",␣
↪type.bigDecimalType()))
                                .setDataSource(createSubreportDataSource());

                return report;
        }

        private JRDataSource createSubreportDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
                for (int i = 0; i < 10; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new␣
↪BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        public static void main(String[] args) {
```

(continues on next page)

(continued from previous page)

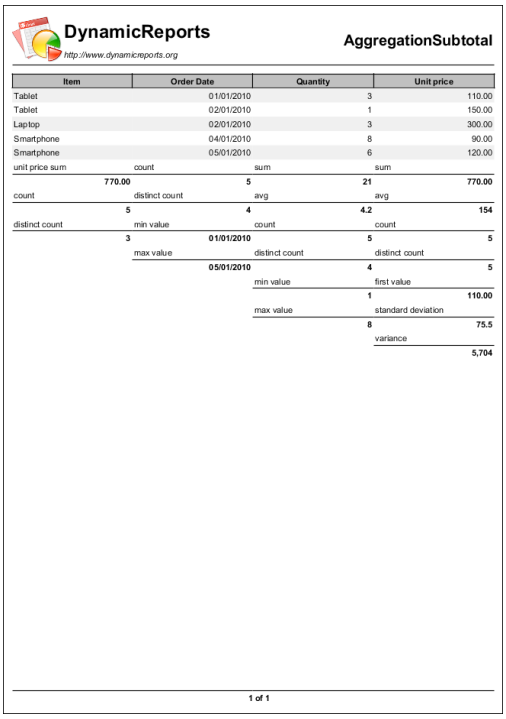
```
80                      new TitleSubreport();
81              }
82  }
```

Table 16: Subreport Examples

 DetailDynamicSubreport	 DetailJasperSubreport	 JasperSubreport
 TitleSubreport		

1.32 Subtotal

1.32.1 Aggregation Subtotal



DynamicReports
http://www.dynamicreports.org

AggregationSubtotal

Item	Order Date	Quantity	Unit price
Tablet	01/01/2010	3	110.00
Tablet	02/01/2010	1	150.00
Laptop	02/01/2010	3	300.00
Smartphone	04/01/2010	8	90.00
Smartphone	05/01/2010	6	120.00

unit price sum	count	sum	sum
770.00	5	21	770.00
count	distinct count	avg	avg
5	4	4.2	154
distinct count	min value	count	count
3	01/01/2010	5	5
	max value	distinct count	distinct count
	05/01/2010	4	5
		min value	first value
		1	110.00
		max value	standard deviation
		8	75.5
			variance
			5,704

1 of 1

```java
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class AggregationSubtotalReport {

        public AggregationSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Date> orderDateColumn = col.column("Order Date",
↪"orderdate", type.dateType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());

                AggregationSubtotalBuilder<Long> itemCount = sbt.count(itemColumn)
```

(continues on next page)

(continued from previous page)

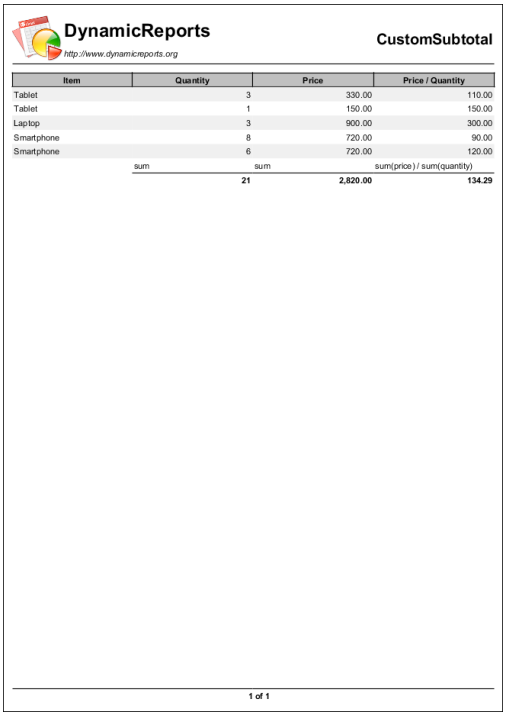
```
                                    .setLabel("count");
                AggregationSubtotalBuilder<Long> itemDistCount = sbt.
↪distinctCount(itemColumn)
                                    .setLabel("distinct count");
                AggregationSubtotalBuilder<BigDecimal> itemSum = sbt.
↪sum(unitPriceColumn)
                                    .setLabel("unit price sum")
                                    .setShowInColumn(itemColumn);

                AggregationSubtotalBuilder<Long> orderDateCount = sbt.
↪count(orderDateColumn)
                                    .setLabel("count");
                AggregationSubtotalBuilder<Long> orderDateDistCount = sbt.
↪distinctCount(orderDateColumn)
                                    .setLabel("distinct count");
                AggregationSubtotalBuilder<Date> orderDateMin = sbt.
↪min(orderDateColumn)
                                    .setLabel("min value");
                AggregationSubtotalBuilder<Date> orderDateMax = sbt.
↪max(orderDateColumn)
                                    .setLabel("max value");

                AggregationSubtotalBuilder<Integer> quantitySum = sbt.
↪sum(quantityColumn)
                                    .setLabel("sum");
                AggregationSubtotalBuilder<Number> quantityAvg = sbt.
↪avg(quantityColumn)
                                    .setLabel("avg");
                AggregationSubtotalBuilder<Long> quantityCount = sbt.
↪count(quantityColumn)
                                    .setLabel("count");
                AggregationSubtotalBuilder<Long> quantityDistCount = sbt.
↪distinctCount(quantityColumn)
                                    .setLabel("distinct count");
                AggregationSubtotalBuilder<Integer> quantityMin = sbt.
↪min(quantityColumn)
                                    .setLabel("min value");
                AggregationSubtotalBuilder<Integer> quantityMax = sbt.
↪max(quantityColumn)
                                    .setLabel("max value");

                AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.
↪sum(unitPriceColumn)
                                    .setLabel("sum");
                AggregationSubtotalBuilder<Number> unitPriceAvg = sbt.
↪avg(unitPriceColumn)
                                    .setLabel("avg");
                AggregationSubtotalBuilder<Long> unitPriceCount = sbt.
↪count(unitPriceColumn)
                                    .setLabel("count");
                AggregationSubtotalBuilder<Long> unitPriceDistCount = sbt.
↪distinctCount(unitPriceColumn)
                                    .setLabel("distinct count");
                AggregationSubtotalBuilder<BigDecimal> unitPriceFirst = sbt.
↪first(unitPriceColumn)
                                    .setLabel("first value");
                AggregationSubtotalBuilder<Number> unitPriceStdDev = sbt.
↪stdDev(unitPriceColumn)
```

(continues on next page)

(continued from previous page)

```
                                .setLabel("standard deviation");
                AggregationSubtotalBuilder<Number> unitPriceVar = sbt.var(unitPriceColumn)
                                .setLabel("variance");

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, orderDateColumn, quantityColumn, unitPriceColumn)
                                        .subtotalsAtSummary(
                                                        itemSum, itemCount, itemDistCount,
                                                        orderDateCount, orderDateDistCount, orderDateMin, orderDateMax,
                                                        quantitySum, quantityAvg, quantityCount, quantityDistCount, quantityMin, quantityMax,
                                                        unitPriceSum, unitPriceAvg, unitPriceCount, unitPriceDistCount, unitPriceFirst, unitPriceStdDev, unitPriceVar)
                                        .title(Templates.createTitleComponent("AggregationSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate", "quantity", "unitprice");
                dataSource.add("Tablet", toDate(2010, 1, 1), 3, new BigDecimal(110));
                dataSource.add("Tablet", toDate(2010, 2, 1), 1, new BigDecimal(150));
                dataSource.add("Laptop", toDate(2010, 2, 1), 3, new BigDecimal(300));
                dataSource.add("Smartphone", toDate(2010, 4, 1), 8, new BigDecimal(90));
                dataSource.add("Smartphone", toDate(2010, 5, 1), 6, new BigDecimal(120));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new AggregationSubtotalReport();
        }
}
```

1.32.2 Custom Subtotal




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
```

(continues on next page)

(continued from previous page)

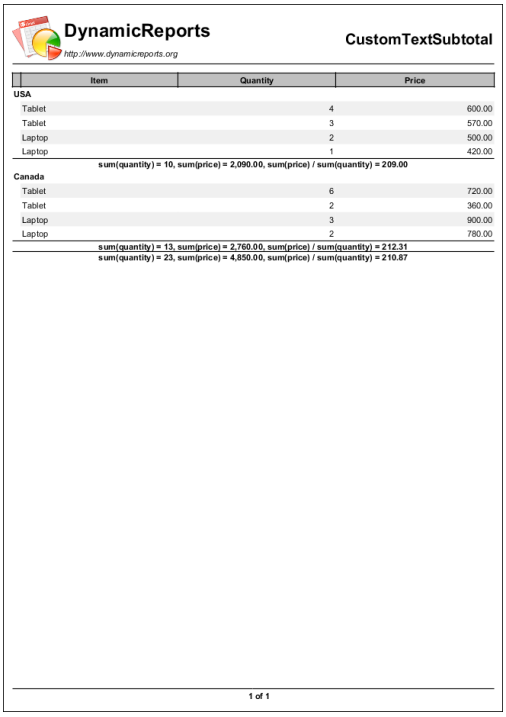
```
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CustomSubtotalReport {
        private AggregationSubtotalBuilder<Integer> quantitySum;
        private AggregationSubtotalBuilder<BigDecimal> priceSum;

        public CustomSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪", type.bigDecimalType());
                TextColumnBuilder<BigDecimal> unitPriceColumn = priceColumn.divide(2,␣
↪quantityColumn)
                                .setTitle("Price / Quantity");

                quantitySum = sbt.sum(quantityColumn)
                                .setLabel("sum");
                priceSum = sbt.sum(priceColumn)
                                .setLabel("sum");
                CustomSubtotalBuilder<BigDecimal> unitPriceSbt = sbt.customValue(new␣
↪UnitPriceSubtotal(), unitPriceColumn)
                                .setLabel("sum(price) / sum(quantity)")
                                .setDataType(type.bigDecimalType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, quantityColumn,␣
↪priceColumn, unitPriceColumn)
                                        .subtotalsAtSummary(
                                                        quantitySum, priceSum,␣
↪unitPriceSbt)
                                        .title(Templates.createTitleComponent(
↪"CustomSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }
```

(continues on next page)

(continued from previous page)

```
80
81          private class UnitPriceSubtotal extends AbstractSimpleExpression<BigDecimal> {
82                  private static final long serialVersionUID = 1L;
83
84                  @Override
85                  public BigDecimal evaluate(ReportParameters reportParameters) {
86                          Integer quantitySumValue = reportParameters.
    ↪getValue(quantitySum);
87                          BigDecimal priceSumValue = reportParameters.
    ↪getValue(priceSum);
88                          return priceSumValue.divide(new BigDecimal(quantitySumValue),␣
    ↪2, BigDecimal.ROUND_HALF_UP);
89                  }
90          }
91
92          private JRDataSource createDataSource() {
93                  DRDataSource dataSource = new DRDataSource("item", "quantity", "price
    ↪");
94                  dataSource.add("Tablet", 3, new BigDecimal(330));
95                  dataSource.add("Tablet", 1, new BigDecimal(150));
96                  dataSource.add("Laptop", 3, new BigDecimal(900));
97                  dataSource.add("Smartphone", 8, new BigDecimal(720));
98                  dataSource.add("Smartphone", 6, new BigDecimal(720));
99                  return dataSource;
100         }
101
102         public static void main(String[] args) {
103                 new CustomSubtotalReport();
104         }
105 }
```

1.32.3 Custom Text Subtotal




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
```

(continues on next page)

(continued from previous page)

```
import net.sf.dynamicreports.report.builder.VariableBuilder;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.Calculation;
import net.sf.dynamicreports.report.constant.Evaluation;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.definition.ReportParameters;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CustomTextSubtotalReport {

        public CustomTextSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪", type.bigDecimalType());

                ColumnGroupBuilder countryGroup = grp.group(countryColumn);

                VariableBuilder<Integer> quantitySum = variable(quantityColumn,␣
↪Calculation.SUM);
                VariableBuilder<BigDecimal> priceSum = variable(priceColumn,␣
↪Calculation.SUM);

                VariableBuilder<Integer> quantityGrpSum = variable(quantityColumn,␣
↪Calculation.SUM);
                quantityGrpSum.setResetGroup(countryGroup);
                VariableBuilder<BigDecimal> priceGrpSum = variable(priceColumn,␣
↪Calculation.SUM);
                priceGrpSum.setResetType(Evaluation.FIRST_GROUP);

                StyleBuilder subtotalStyle = stl.style()
                                .bold()
                                .setTopBorder(stl.pen1Point())
                                .setHorizontalTextAlignment(HorizontalTextAlignment.
↪CENTER);

                TextFieldBuilder<String> summarySbt = cmp.text(new␣
↪CustomTextSubtotal(quantitySum, priceSum))
                                .setStyle(subtotalStyle);

                TextFieldBuilder<String> groupSbt = cmp.text(new␣
↪CustomTextSubtotal(quantityGrpSum, priceGrpSum))
```

(continues on next page)

(continued from previous page)

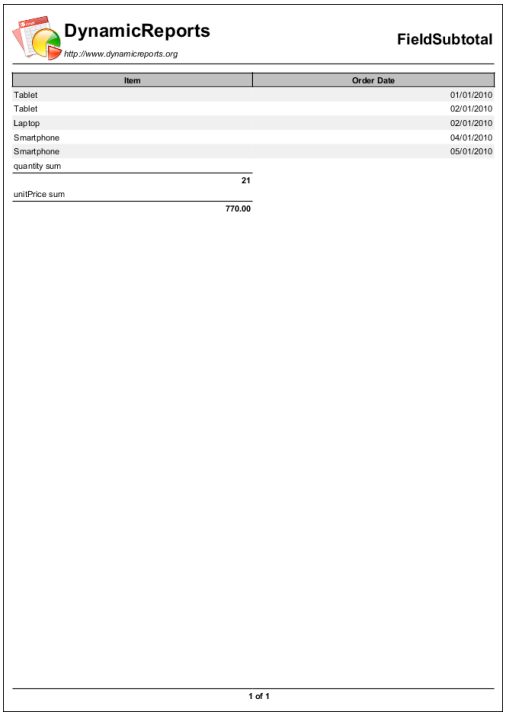
```
                                .setStyle(subtotalStyle);

                countryGroup.footer(groupSbt);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .variables(
                                                        quantitySum, priceSum,␣
↪quantityGrpSum, priceGrpSum)
                                        .columns(
                                                        countryColumn, itemColumn,␣
↪quantityColumn, priceColumn)
                                        .groupBy(
                                                        countryGroup)
                                        .summary(
                                                        summarySbt)
                                        .title(Templates.createTitleComponent(
↪"CustomTextSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private class CustomTextSubtotal extends AbstractSimpleExpression<String> {
                private static final long serialVersionUID = 1L;

                private VariableBuilder<Integer> quantitySum;
                private VariableBuilder<BigDecimal> priceSum;

                public CustomTextSubtotal(VariableBuilder<Integer> quantitySum,␣
↪VariableBuilder<BigDecimal> priceSum) {
                        this.quantitySum = quantitySum;
                        this.priceSum = priceSum;
                }

                @Override
                public String evaluate(ReportParameters reportParameters) {
                        Integer quantitySumValue = reportParameters.
↪getValue(quantitySum);
                        BigDecimal priceSumValue = reportParameters.
↪getValue(priceSum);
                        BigDecimal unitPriceSbt = priceSumValue.divide(new␣
↪BigDecimal(quantitySumValue), 2, BigDecimal.ROUND_HALF_UP);
                        return "sum(quantity) = " + type.integerType().
↪valueToString(quantitySum, reportParameters) + ", " +
                                        "sum(price) = " + type.bigDecimalType().
↪valueToString(priceSum, reportParameters) + ", " +
                                        "sum(price) / sum(quantity) = " + type.
↪bigDecimalType().valueToString(unitPriceSbt, reportParameters.getLocale());
                }
        }

        private JRDataSource createDataSource() {
```

(continues on next page)

(continued from previous page)

```
                DRDataSource dataSource = new DRDataSource("country", "item",
↪"quantity", "price");
                dataSource.add("USA", "Tablet", 4, new BigDecimal(600));
                dataSource.add("USA", "Tablet", 3, new BigDecimal(570));
                dataSource.add("USA", "Laptop", 2, new BigDecimal(500));
                dataSource.add("USA", "Laptop", 1, new BigDecimal(420));
                dataSource.add("Canada", "Tablet", 6, new BigDecimal(720));
                dataSource.add("Canada", "Tablet", 2, new BigDecimal(360));
                dataSource.add("Canada", "Laptop", 3, new BigDecimal(900));
                dataSource.add("Canada", "Laptop", 2, new BigDecimal(780));
                return dataSource;
        }

        public static void main(String[] args) {
                new CustomTextSubtotalReport();
        }
}
```

1.32.4 Field Subtotal




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
```

(continues on next page)

(continued from previous page)

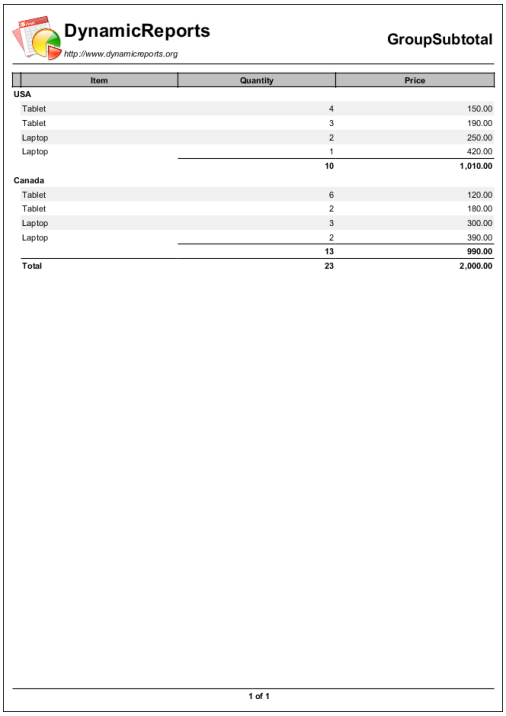
```
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class FieldSubtotalReport {

        public FieldSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Date> orderDateColumn = col.column("Order Date",
↪"orderdate", type.dateType());

                AggregationSubtotalBuilder<Integer> quantitySum = sbt.sum("quantity",␣
↪Integer.class, itemColumn)
                                .setLabel("quantity sum");
                AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.sum(
↪"unitprice", BigDecimal.class, itemColumn)
                                .setLabel("unitPrice sum");

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        itemColumn, orderDateColumn)
                                        .subtotalsAtSummary(
                                                        quantitySum, unitPriceSum)
```

(continues on next page)

(continued from previous page)

```
                                        .title(Templates.createTitleComponent(
↪"FieldSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DREException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
                dataSource.add("Tablet", toDate(2010, 1, 1), 3, new BigDecimal(110));
                dataSource.add("Tablet", toDate(2010, 2, 1), 1, new BigDecimal(150));
                dataSource.add("Laptop", toDate(2010, 2, 1), 3, new BigDecimal(300));
                dataSource.add("Smartphone", toDate(2010, 4, 1), 8, new␣
↪BigDecimal(90));
                dataSource.add("Smartphone", toDate(2010, 5, 1), 6, new␣
↪BigDecimal(120));
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new FieldSubtotalReport();
        }
}
```

1.32.5 Group Subtotal



DynamicReports
http://www.dynamicreports.org

GroupSubtotal

Item	Quantity	Price
USA		
Tablet	4	150.00
Tablet	3	190.00
Laptop	2	250.00
Laptop	1	420.00
	10	1,010.00
Canada		
Tablet	6	120.00
Tablet	2	180.00
Laptop	3	300.00
Laptop	2	390.00
	13	990.00
Total	23	2,000.00

1 of 1

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
```

(continues on next page)

(continued from previous page)

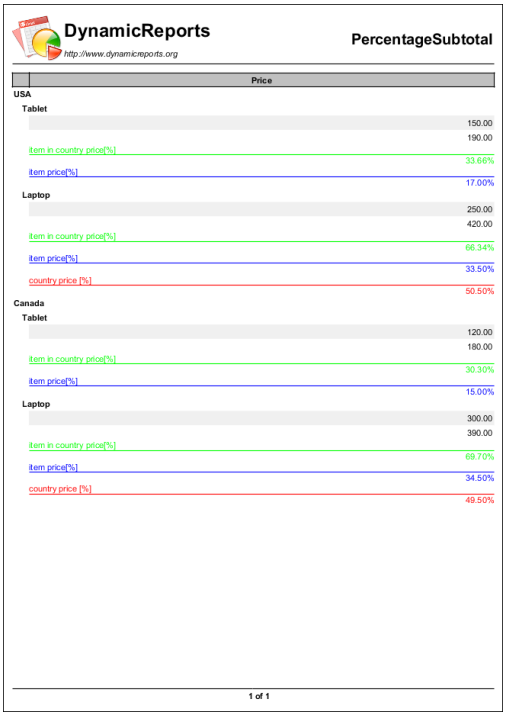
```
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class GroupSubtotalReport {

        public GroupSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪", type.bigDecimalType());

                ColumnGroupBuilder countryGroup = grp.group(countryColumn);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        countryColumn, itemColumn,␣
↪quantityColumn, priceColumn)
                                        .groupBy(
                                                        countryGroup)
                                        .subtotalsAtFirstGroupFooter(
                                                        sbt.sum(quantityColumn))
                                        .subtotalsAtGroupFooter(
                                                        countryGroup, sbt.
↪sum(priceColumn))
                                        .subtotalsAtSummary(
                                                        sbt.text("Total", itemColumn),
↪ sbt.sum(quantityColumn), sbt.sum(priceColumn))
                                        .title(Templates.createTitleComponent(
↪"GroupSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("country", "item",
↪"quantity", "price");
                dataSource.add("USA", "Tablet", 4, new BigDecimal(150));
                dataSource.add("USA", "Tablet", 3, new BigDecimal(190));
```

(continues on next page)

(continued from previous page)

```
                dataSource.add("USA", "Laptop", 2, new BigDecimal(250));
                dataSource.add("USA", "Laptop", 1, new BigDecimal(420));
                dataSource.add("Canada", "Tablet", 6, new BigDecimal(120));
                dataSource.add("Canada", "Tablet", 2, new BigDecimal(180));
                dataSource.add("Canada", "Laptop", 3, new BigDecimal(300));
                dataSource.add("Canada", "Laptop", 2, new BigDecimal(390));
                return dataSource;
        }

        public static void main(String[] args) {
                new GroupSubtotalReport();
        }
}
```

1.32.6 Percentage Subtotal




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
```

(continues on next page)

(continued from previous page)

```
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.subtotal;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.builder.subtotal.PercentageSubtotalBuilder;
import net.sf.dynamicreports.report.constant.PercentageTotalType;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class PercentageSubtotalReport {

        public PercentageSubtotalReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());
                TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪", type.bigDecimalType());

                ColumnGroupBuilder countryGroup = grp.group(countryColumn);
                ColumnGroupBuilder itemGroup = grp.group(itemColumn);

                StyleBuilder countryLabelStyle = stl.style()
                                .setForegroundColor(Color.RED);
                StyleBuilder countryStyle = stl.style(countryLabelStyle)
                                .setTopBorder(stl.pen1Point());
                StyleBuilder itemInCountryLabelStyle = stl.style()
                                .setForegroundColor(Color.GREEN);
                StyleBuilder itemInCountryStyle = stl.style(itemInCountryLabelStyle)
                                .setTopBorder(stl.pen1Point());
                StyleBuilder itemLabelStyle = stl.style()
                                .setForegroundColor(Color.BLUE);
                StyleBuilder itemStyle = stl.style(itemLabelStyle)
```

(continues on next page)

(continued from previous page)

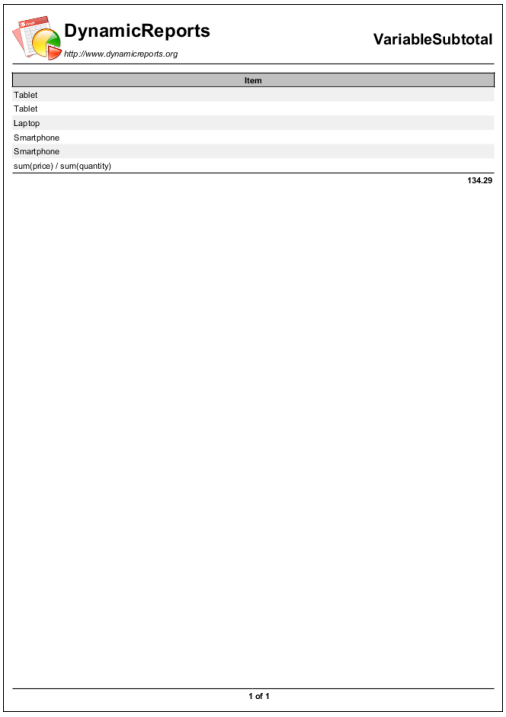
```
                                .setTopBorder(stl.pen1Point());

                PercentageSubtotalBuilder countryPercentage = sbt.percentage(priceColumn)
                                .setLabel("country price [%]")
                                .setLabelStyle(countryLabelStyle)
                                .setStyle(countryStyle);

                PercentageSubtotalBuilder itemInCountryPercentage = sbt.percentage(priceColumn)
                                .setLabel("item in country price[%]")
                                .setLabelStyle(itemInCountryLabelStyle)
                                .setStyle(itemInCountryStyle);

                PercentageSubtotalBuilder itemPercentage = sbt.percentage(priceColumn)
                                .setLabel("item price[%]")
                                .setLabelStyle(itemLabelStyle)
                                .setStyle(itemStyle)
                                .setTotalType(PercentageTotalType.REPORT);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .columns(
                                                        countryColumn, itemColumn, priceColumn)
                                        .groupBy(
                                                        countryGroup, itemGroup)
                                        .subtotalsOfPercentageAtGroupFooter(
                                                        countryGroup, countryPercentage)
                                        .subtotalsOfPercentageAtGroupFooter(
                                                        itemGroup, itemInCountryPercentage, itemPercentage)
                                        .title(Templates.createTitleComponent("PercentageSubtotal"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                DRDataSource dataSource = new DRDataSource("country", "item", "price");
                dataSource.add("USA", "Tablet", new BigDecimal(150));
                dataSource.add("USA", "Tablet", new BigDecimal(190));
                dataSource.add("USA", "Laptop", new BigDecimal(250));
                dataSource.add("USA", "Laptop", new BigDecimal(420));
                dataSource.add("Canada", "Tablet", new BigDecimal(120));
                dataSource.add("Canada", "Tablet", new BigDecimal(180));
                dataSource.add("Canada", "Laptop", new BigDecimal(300));
                dataSource.add("Canada", "Laptop", new BigDecimal(390));
                return dataSource;
        }
```

(continues on next page)

(continued from previous page)

```
118
119         public static void main(String[] args) {
120                 new PercentageSubtotalReport();
121         }
122 }
```

1.32.7 Variable Subtotal




```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
```

(continues on next page)

(continued from previous page)

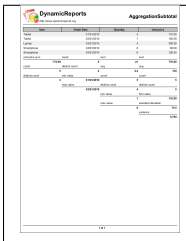
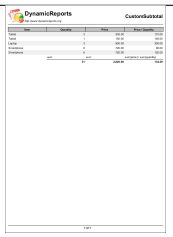
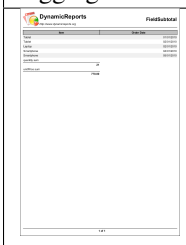
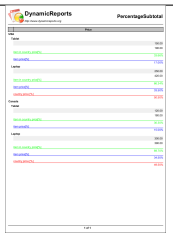
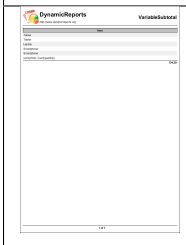
```
22
23  package net.sf.dynamicreports.examples.subtotal;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31  import net.sf.dynamicreports.report.builder.VariableBuilder;
32  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33  import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
34  import net.sf.dynamicreports.report.constant.Calculation;
35  import net.sf.dynamicreports.report.datasource.DRDataSource;
36  import net.sf.dynamicreports.report.definition.ReportParameters;
37  import net.sf.dynamicreports.report.exception.DRException;
38  import net.sf.jasperreports.engine.JRDataSource;
39
40  /**
41   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42   */
43  public class VariableSubtotalReport {
44          private VariableBuilder<Integer> quantitySum;
45          private VariableBuilder<BigDecimal> priceSum;
46
47          public VariableSubtotalReport() {
48                  build();
49          }
50
51          private void build() {
52                  quantitySum = variable("quantity", Integer.class, Calculation.SUM);
53                  priceSum = variable("price", BigDecimal.class, Calculation.SUM);
54
55                  TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
    ↪type.stringType());
56
57                  CustomSubtotalBuilder<BigDecimal> unitPriceSbt = sbt.customValue(new␣
    ↪UnitPriceSubtotal(), itemColumn)
58                                  .setLabel("sum(price) / sum(quantity)")
59                                  .setDataType(type.bigDecimalType());
60
61                  try {
62                          report()
63                                          .setTemplate(Templates.reportTemplate)
64                                          .variables(
65                                                          quantitySum, priceSum)
66                                          .columns(
67                                                          itemColumn)
68                                          .subtotalsAtSummary(
69                                                          unitPriceSbt)
70                                          .title(Templates.createTitleComponent(
    ↪"VariableSubtotal"))
71                                          .pageFooter(Templates.footerComponent)
72                                          .setDataSource(createDataSource())
73                                          .show();
74                  } catch (DRException e) {
75                          e.printStackTrace();
```

(continues on next page)

(continued from previous page)

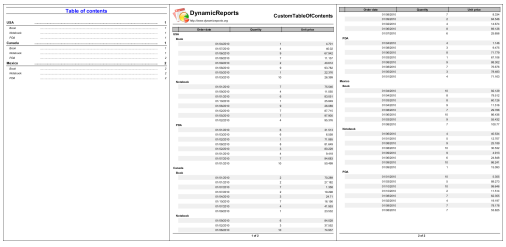
```
76                  }
77          }
78
79          private class UnitPriceSubtotal extends AbstractSimpleExpression<BigDecimal> {
80                  private static final long serialVersionUID = 1L;
81
82                  @Override
83                  public BigDecimal evaluate(ReportParameters reportParameters) {
84                          Integer quantitySumValue = reportParameters.
    ↪getValue(quantitySum);
85                          BigDecimal priceSumValue = reportParameters.
    ↪getValue(priceSum);
86                          return priceSumValue.divide(new BigDecimal(quantitySumValue),␣
    ↪2, BigDecimal.ROUND_HALF_UP);
87                  }
88          }
89
90          private JRDataSource createDataSource() {
91                  DRDataSource dataSource = new DRDataSource("item", "quantity", "price
    ↪");
92                  dataSource.add("Tablet", 3, new BigDecimal(330));
93                  dataSource.add("Tablet", 1, new BigDecimal(150));
94                  dataSource.add("Laptop", 3, new BigDecimal(900));
95                  dataSource.add("Smartphone", 8, new BigDecimal(720));
96                  dataSource.add("Smartphone", 6, new BigDecimal(720));
97                  return dataSource;
98          }
99
100         public static void main(String[] args) {
101                 new VariableSubtotalReport();
102         }
103 }
```

Table 17: Subtotal Examples

 AggregationSubtotalReport	 CustomSubtotalReport	 CustomTextSubtotalReport
 FieldSubtotalReport	 GroupSubtotalReport	 PercentageSubtotalReport
 VariableSubtotalReport		

1.33 Table Of Contents

1.33.1 Custom Table Of Contents



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
```

(continues on next page)

(continued from previous page)

```
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.tableofcontents;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.awt.Color;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.builder.tableofcontents.TableOfContentsCustomizer;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class CustomTableOfContentsReport {

        public CustomTableOfContentsReport() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
 ↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
 ↪type.stringType());

                StyleBuilder titleTocStyle = stl.style()
                                .setForegroundColor(Color.BLUE)
                                .setFontSize(18)
                                .bold()
                                .setHorizontalTextAlignment(HorizontalTextAlignment.
 ↪CENTER);
                StyleBuilder headingToc0Style = stl.style(Templates.rootStyle)
                                .setFontSize(12)
                                .bold();
                StyleBuilder headingToc1Style = stl.style(Templates.rootStyle)
                                .italic();
```

(continues on next page)

(continued from previous page)

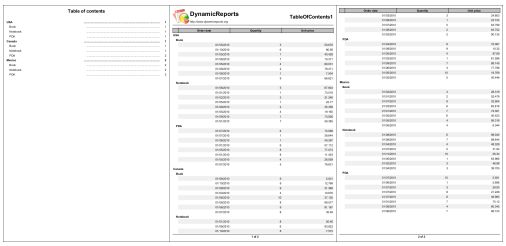
```
66
67                CustomTableOfContentsCustomizer tableOfContentsCustomizer = new␣
↪CustomTableOfContentsCustomizer();
68                tableOfContentsCustomizer.setTitleStyle(titleTocStyle);
69                tableOfContentsCustomizer.setHeadingStyle(0, headingToc0Style);
70                tableOfContentsCustomizer.setHeadingStyle(1, headingToc1Style);
71                tableOfContentsCustomizer.setTextFixedWidth(100);
72                tableOfContentsCustomizer.setPageIndexFixedWidth(30);
73
74                try {
75                        report()
76                                        .setTemplate(Templates.reportTemplate)
77                                        .tableOfContents(tableOfContentsCustomizer)
78                                        .columns(
79                                                        countryColumn,
80                                                        itemColumn,
81                                                        col.column("Order date",
↪"orderdate", type.dateType()),
82                                                        col.column("Quantity",
↪"quantity", type.integerType()),
83                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
84                                        .groupBy(countryColumn, itemColumn)
85                                        .title(Templates.createTitleComponent(
↪"CustomTableOfContents"))
86                                        .pageFooter(Templates.footerComponent)
87                                        .setDataSource(createDataSource())
88                                        .show();
89                } catch (DRException e) {
90                        e.printStackTrace();
91                }
92        }
93
94        private JRDataSource createDataSource() {
95                String[] countries = new String[] { "USA", "Canada", "Mexico" };
96                String[] items = new String[] { "Book", "Notebook", "PDA" };
97                DRDataSource dataSource = new DRDataSource("country", "item",
↪"orderdate", "quantity", "unitprice");
98                for (String country : countries) {
99                        for (String item : items) {
100                                for (int i = 0; i < 8; i++) {
101                                        dataSource.add(country, item, toDate(2010, 1,␣
↪(int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
102                                                        new BigDecimal(Math.random()␣
↪* 100 + 1));
103                                }
104                        }
105                }
106                return dataSource;
107        }
108
109        private Date toDate(int year, int month, int day) {
110                Calendar c = Calendar.getInstance();
111                c.set(Calendar.YEAR, year);
112                c.set(Calendar.MONTH, month - 1);
113                c.set(Calendar.DAY_OF_MONTH, day);
114                return c.getTime();
```

(continues on next page)

(continued from previous page)

```
        }

        public static void main(String[] args) {
                new CustomTableOfContentsReport();
        }

        private class CustomTableOfContentsCustomizer extends␣
↪TableOfContentsCustomizer {
                private static final long serialVersionUID = 1L;

                @Override
                protected ComponentBuilder<?, ?> title() {
                        VerticalListBuilder verticalList = cmp.verticalList();
                        verticalList.add(cmp.line());
                        verticalList.add(super.title());
                        verticalList.add(cmp.line());
                        return verticalList;
                }

                @Override
                protected ComponentBuilder<?, ?> headingComponent(int level) {
                        if (level == 0) {
                                VerticalListBuilder verticalList = cmp.verticalList();
                                verticalList.add(super.headingComponent(level));
                                verticalList.add(cmp.line());
                                return verticalList;
                        } else {
                                return super.headingComponent(level);
                        }
                }
        }
}
```

1.33.2 Table Of Contents



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
```

(continues on next page)

(continued from previous page)

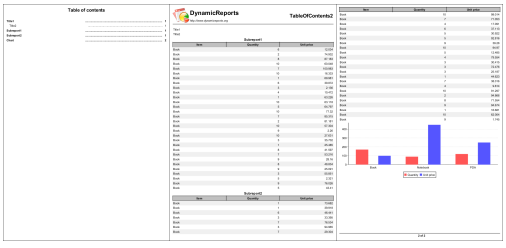
```
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.tableofcontents;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TableOfContentsReport1 {

        public TableOfContentsReport1() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .tableOfContents()
                                        .columns(
                                                        countryColumn,
                                                        itemColumn,
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .groupBy(countryColumn, itemColumn)
                                        .title(Templates.createTitleComponent(
↪"TableOfContents1"))
```

(continues on next page)

(continued from previous page)

```
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }

        private JRDataSource createDataSource() {
                String[] countries = new String[] { "USA", "Canada", "Mexico" };
                String[] items = new String[] { "Book", "Notebook", "PDA" };
                DRDataSource dataSource = new DRDataSource("country", "item",
↪"orderdate", "quantity", "unitprice");
                for (String country : countries) {
                        for (String item : items) {
                                for (int i = 0; i < 8; i++) {
                                        dataSource.add(country, item, toDate(2010, 1,␣
↪(int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
                                                        new BigDecimal(Math.random()␣
↪* 100 + 1));
                                }
                        }
                }
                return dataSource;
        }

        private Date toDate(int year, int month, int day) {
                Calendar c = Calendar.getInstance();
                c.set(Calendar.YEAR, year);
                c.set(Calendar.MONTH, month - 1);
                c.set(Calendar.DAY_OF_MONTH, day);
                return c.getTime();
        }

        public static void main(String[] args) {
                new TableOfContentsReport1();
        }
}
```

1.33.3 Table Of Contents 2



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

```
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.tableofcontents;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
import net.sf.dynamicreports.report.builder.FieldBuilder;
import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
import net.sf.dynamicreports.report.builder.tableofcontents.
↪TableOfContentsHeadingBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TableOfContentsReport2 {

        public TableOfContentsReport2() {
                build();
        }

        private void build() {
                FieldBuilder<String> itemField = field("item", type.stringType());
                FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
                FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());

                TableOfContentsHeadingBuilder tocHeading1 = tableOfContentsHeading();
                TextFieldBuilder<String> title1 = cmp.text("Title1")
                                .setTableOfContentsHeading(tocHeading1);

                TableOfContentsHeadingBuilder tocHeading2 = tableOfContentsHeading()
                                .setParentHeading(tocHeading1);
```

(continues on next page)

(continued from previous page)

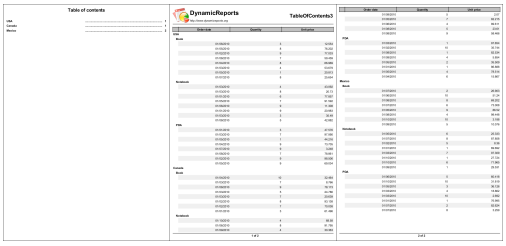
```
59                  TextFieldBuilder<String> title2 = cmp.text("Title2")
60                                  .setTableOfContentsHeading(tocHeading2);
61
62                  BarChartBuilder chart = cht.barChart()
63                                  .setDataSource(createChartDataSource())
64                                  .setCategory(itemField)
65                                  .series(
66                                                  cht.serie(quantityField).setLabel(
    ↪"Quantity"),
67                                                  cht.serie(unitPriceField).setLabel(
    ↪"Unit price"))
68                                  .setTableOfContentsHeading("Chart");
69
70                  try {
71                          report()
72                                          .setTemplate(Templates.reportTemplate)
73                                          .tableOfContents()
74                                          .title(
75                                                          Templates.
    ↪createTitleComponent("TableOfContents2"),
76                                                          title1, title2,
77                                                          cmp.
    ↪subreport(createSubreport(1)),
78                                                          cmp.
    ↪subreport(createSubreport(2)),
79                                                          chart)
80                                          .pageFooter(Templates.footerComponent)
81                                          .show();
82                  } catch (DRException e) {
83                          e.printStackTrace();
84                  }
85          }
86
87          private JasperReportBuilder createSubreport(int index) {
88                  TextFieldBuilder<String> title = cmp.text("Subreport" + index)
89                                  .setStyle(Templates.bold12CenteredStyle)
90                                  .setTableOfContentsHeading(tableOfContentsHeading());
91
92                  JasperReportBuilder report = report();
93                  report
94                                  .setTemplate(Templates.reportTemplate)
95                                  .title(title)
96                                  .columns(
97                                                  col.column("Item", "item", type.
    ↪stringType()),
98                                                  col.column("Quantity", "quantity",␣
    ↪type.integerType()),
99                                                  col.column("Unit price", "unitprice",␣
    ↪type.bigDecimalType()))
100                                 .setDataSource(createSubreportDataSource());
101
102                 return report;
103         }
104
105         private JRDataSource createSubreportDataSource() {
106                 DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
```

(continues on next page)

(continued from previous page)

```
                for (int i = 0; i < 30; i++) {
                        dataSource.add("Book", (int) (Math.random() * 10) + 1, new BigDecimal(Math.random() * 100 + 1));
                }
                return dataSource;
        }

        private JRDataSource createChartDataSource() {
                DRDataSource dataSource = new DRDataSource("item", "quantity", "unitprice");
                dataSource.add("Book", 170, new BigDecimal(100));
                dataSource.add("Notebook", 90, new BigDecimal(450));
                dataSource.add("PDA", 120, new BigDecimal(250));
                return dataSource;
        }

        public static void main(String[] args) {
                new TableOfContentsReport2();
        }
}
```

1.33.4 Table Of Contents 3



```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

```

(continues on next page)

(continued from previous page)

```
package net.sf.dynamicreports.examples.tableofcontents;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class TableOfContentsReport3 {

        public TableOfContentsReport3() {
                build();
        }

        private void build() {
                TextColumnBuilder<String> countryColumn = col.column("Country",
↪"country", type.stringType());
                TextColumnBuilder<String> itemColumn = col.column("Item", "item",␣
↪type.stringType());

                ColumnGroupBuilder countryGroup = grp.group(countryColumn);
                ColumnGroupBuilder itemGroup = grp.group(itemColumn)
                                .setAddToTableOfContents(false);

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .tableOfContents()
                                        .columns(
                                                        countryColumn,
                                                        itemColumn,
                                                        col.column("Order date",
↪"orderdate", type.dateType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
                                        .groupBy(countryGroup, itemGroup)
                                        .title(Templates.createTitleComponent(
↪"TableOfContents3"))
                                        .pageFooter(Templates.footerComponent)
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                }
        }
```

(continues on next page)

(continued from previous page)

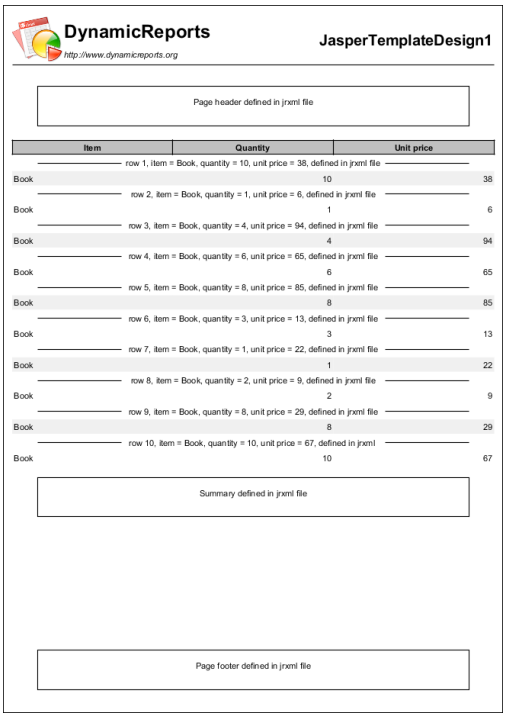
```
74
75          private JRDataSource createDataSource() {
76                  String[] countries = new String[] { "USA", "Canada", "Mexico" };
77                  String[] items = new String[] { "Book", "Notebook", "PDA" };
78                  DRDataSource dataSource = new DRDataSource("country", "item",
    ↪"orderdate", "quantity", "unitprice");
79                  for (String country : countries) {
80                          for (String item : items) {
81                                  for (int i = 0; i < 8; i++) {
82                                          dataSource.add(country, item, toDate(2010, 1,␣
    ↪(int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
83                                                                          new BigDecimal(Math.random()␣
    ↪* 100 + 1));
84                                  }
85                          }
86                  }
87                  return dataSource;
88          }
89
90          private Date toDate(int year, int month, int day) {
91                  Calendar c = Calendar.getInstance();
92                  c.set(Calendar.YEAR, year);
93                  c.set(Calendar.MONTH, month - 1);
94                  c.set(Calendar.DAY_OF_MONTH, day);
95                  return c.getTime();
96          }
97
98          public static void main(String[] args) {
99                  new TableOfContentsReport3();
100         }
101 }
```

Table 18: Table Of Contents Examples

 CustomTableOfContentsReport	 TableOfContentsReport1	 TableOfContentsReport2
 TableOfContentsReport3		

1.34 Template Design

1.34.1 Jasper Template Design 1



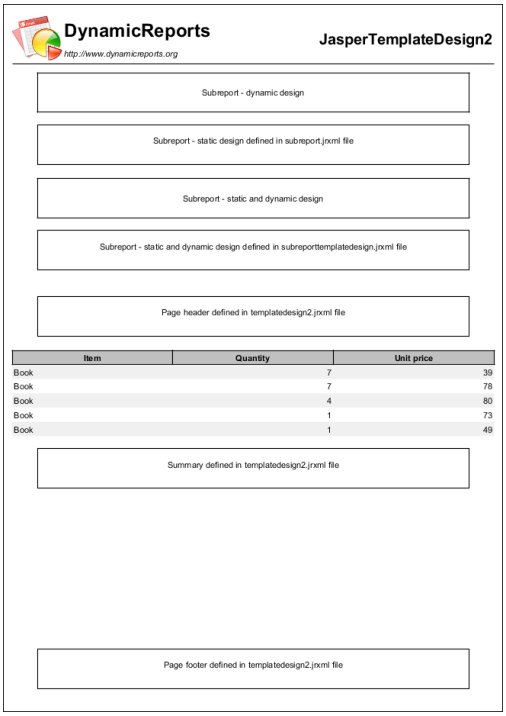

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.templatedesign;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;
```

(continues on next page)

(continued from previous page)

```
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.datasource.DRDataSource;
31  import net.sf.dynamicreports.report.exception.DRException;
32  import net.sf.jasperreports.engine.JRDataSource;
33
34  /**
35   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36   */
37  public class JasperTemplateDesignReport1 {
38
39          public JasperTemplateDesignReport1() {
40                  build();
41          }
42
43          private void build() {
44                  InputStream is = JasperTemplateDesignReport1.class.
    ↪getResourceAsStream("templatedesign1.jrxml");
45
46                  try {
47                          report()
48                                          .setTemplate(Templates.reportTemplate)
49                                          .setTemplateDesign(is)
50                                          .columns(
51                                                          col.column("Item", "item",␣
    ↪type.stringType()),
52                                                          col.column("Quantity",
    ↪"quantity", type.integerType()),
53                                                          col.column("Unit price",
    ↪"unitprice", type.integerType()))
54                                          .title(Templates.createTitleComponent(
    ↪"JasperTemplateDesign1"))
55                                          .setDataSource(createDataSource())
56                                          .show();
57                  } catch (DRException e) {
58                          e.printStackTrace();
59                  }
60          }
61
62          private JRDataSource createDataSource() {
63                  DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪"unitprice");
64                  for (int i = 0; i < 10; i++) {
65                          dataSource.add("Book", (int) (Math.random() * 10) + 1, (int)␣
    ↪(Math.random() * 100) + 1);
66                  }
67                  return dataSource;
68          }
69
70          public static void main(String[] args) {
71                  new JasperTemplateDesignReport1();
72          }
73  }
```

1.34.2 Jasper Template Design 2




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.templatedesign;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
```

(continues on next page)

(continued from previous page)

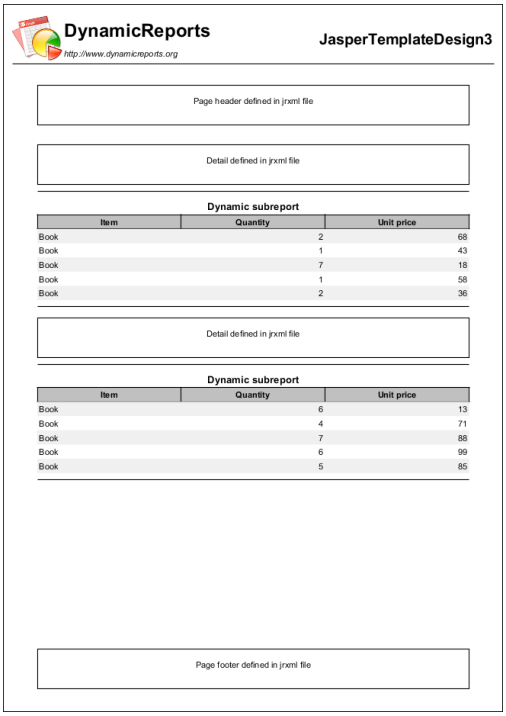
```
import net.sf.dynamicreports.report.builder.style.StyleBuilder;
import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
import net.sf.dynamicreports.report.datasource.DRDataSource;
import net.sf.dynamicreports.report.exception.DRException;
import net.sf.jasperreports.engine.JRDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperReport;

/**
 * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
 */
public class JasperTemplateDesignReport2 {

        public JasperTemplateDesignReport2() {
                build();
        }

        private void build() {
                InputStream is = JasperTemplateDesignReport2.class.
↪getResourceAsStream("templatedesign2.jrxml");

                try {
                        report()
                                        .setTemplate(Templates.reportTemplate)
                                        .setTemplateDesign(is)
                                        .columns(
                                                        col.column("Item", "item",␣
↪type.stringType()),
                                                        col.column("Quantity",
↪"quantity", type.integerType()),
                                                        col.column("Unit price",
↪"unitprice", type.integerType()))
                                        .title(
                                                        Templates.
↪createTitleComponent("JasperTemplateDesign2"),
                                                        cmp.
↪subreport(createDynamicSubreport()),
                                                        cmp.
↪subreport(createStaticSubreport()),
                                                        cmp.
↪subreport(createStaticAndDynamicSubreport()))
                                        .setDataSource(createDataSource())
                                        .show();
                } catch (DRException e) {
                        e.printStackTrace();
                } catch (JRException e) {
                        e.printStackTrace();
                }
        }

        private JasperReportBuilder createDynamicSubreport() {
                return createSubreport("Subreport - dynamic design");
        }

        private JasperReportBuilder createStaticAndDynamicSubreport() throws␣
↪DRException {
```

(continues on next page)

(continued from previous page)

```
80                  InputStream is = JasperTemplateDesignReport2.class.
↪getResourceAsStream("subreporttemplatedesign.jrxml");
81                  JasperReportBuilder report = createSubreport("Subreport - static and␣
↪dynamic design");
82                  report.setTemplateDesign(is);
83                  return report;
84          }
85
86          private JasperReportBuilder createSubreport(String title) {
87                  StyleBuilder style = stl.style()
88                                  .setHorizontalTextAlignment(HorizontalTextAlignment.
↪CENTER)
89                                  .setVerticalTextAlignment(VerticalTextAlignment.
↪MIDDLE)
90                                  .setBorder(stl.pen1Point());
91
92                  JasperReportBuilder report = report();
93                  report
94                                  .setTemplate(Templates.reportTemplate)
95                                  .title(
96                                                  cmp.horizontalList(
97                                                                  cmp.gap(30, 47),
98                                                                  cmp.text(title).
↪setStyle(style),
99                                                                  cmp.gap(30, 47)));
100
101                 return report;
102         }
103
104         private JasperReport createStaticSubreport() throws JRException {
105                 InputStream is = JasperTemplateDesignReport2.class.
↪getResourceAsStream("subreport.jrxml");
106                 return JasperCompileManager.compileReport(is);
107         }
108
109         private JRDataSource createDataSource() {
110                 DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
111                 for (int i = 0; i < 5; i++) {
112                         dataSource.add("Book", (int) (Math.random() * 10) + 1, (int)␣
↪(Math.random() * 100) + 1);
113                 }
114                 return dataSource;
115         }
116
117         public static void main(String[] args) {
118                 new JasperTemplateDesignReport2();
119         }
120 }
```

1.34.3 Jasper Template Design 3




```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
 * http://www.dynamicreports.org
 *
 * This file is part of DynamicReports.
 *
 * DynamicReports is free software: you can redistribute it and/or modify
 * it under the terms of the GNU Lesser General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * DynamicReports is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU Lesser General Public License for more details.
 *
 * You should have received a copy of the GNU Lesser General Public License
 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
 */

package net.sf.dynamicreports.examples.templatedesign;

import static net.sf.dynamicreports.report.builder.DynamicReports.*;

import java.io.InputStream;

import net.sf.dynamicreports.examples.Templates;
import net.sf.dynamicreports.report.exception.DRException;
```

(continues on next page)

(continued from previous page)

```
31 import net.sf.jasperreports.engine.JRDataSource;
32 import net.sf.jasperreports.engine.JREmptyDataSource;
33
34 /**
35  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36  */
37 public class JasperTemplateDesignReport3 {
38
39         public JasperTemplateDesignReport3() {
40                 build();
41         }
42
43         private void build() {
44                 InputStream is = JasperTemplateDesignReport3.class.
   ↪getResourceAsStream("templatedesign3.jrxml");
45
46                 try {
47                         report()
48                                                 .setTemplateDesign(is)
49                                                 .title(Templates.createTitleComponent(
   ↪"JasperTemplateDesign3"))
50                                                 .setDataSource(createDataSource())
51                                                 .show();
52                 } catch (DRException e) {
53                         e.printStackTrace();
54                 }
55         }
56
57         private JRDataSource createDataSource() {
58                 return new JREmptyDataSource(2);
59         }
60
61         public static void main(String[] args) {
62                 new JasperTemplateDesignReport3();
63         }
64 }
```

Table 19: Template Design

 JasperTemplateDesignReport1	 JasperTemplateDesignReport1	 JasperTemplateDesignReport1