JFreeChart tutorial

last modified July 6, 2020

In this tutorial, we learn how to use JFreeChart. We show how to create various types of charts. The charts are displayed in a Swing application and saved to an image file. We use a Java servlet to create and render a chart in a web browser and retrieve data for a chart from a MySQL database.

JFreeChart library

A chart is a drawing that shows information in a simple way, often using lines and curves to show amounts. *JFreeChart* is a popular Java library for creating charts. JFreeChart allows to create a wide variety of both interactive and non-interactive charts. We can create line charts, bar charts, area charts, scatter charts, pie charts, gantt charts and various specialized charts such as wind chart or bubble chart.

JFreeChart is extensively customizable; it allows to modify colours and paints of chart items, legends, styles of the lines or markers. It automatically draws the axis scales and legends. Charts have a built-in capability to zoom in with mouse. The existing charts can be easily updated through the listeners that the library has on its data collections. It supports multiple output formats including PNG, JPEG, PDF, and SVG.

JFreeChart was started by David Gilbert in 2000. Today, JFreeChart is the most widely used charting library among Java developers.

JFreeChart Maven dependency

```
<dependency>
    <groupId>org.jfree</groupId>
    <artifactId>jfreechart</artifactId>
    <version>1.5.0</version>
</dependency>
```

For our projects we use this Maven dependency.

JFreeChart histogram

A histogram is an accurate representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable.

com/zetcode/HistogramEx.java

```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtils;
import org.jfree.chart.JFreeChart;
import org.jfree.data.statistics.HistogramDataset;

import java.io.File;
import java.io.IOException;

public class HistogramEx {

    public static void main(String[] args) throws IOException {

        double[] vals = {

                0.71477137, 0.55749811, 0.50809619, 0.47027228, 0.25281568,
                0.66633175, 0.50676332, 0.6007552, 0.56892904, 0.49553407,
                0.61093935, 0.65057417, 0.40095626, 0.45969447, 0.51087888,
                0.52894806, 0.49397198, 0.4267163, 0.54091298, 0.34545257,
                0.58548892, 0.3137885, 0.63521146, 0.57541744, 0.59862265,
                0.66261386, 0.56744017, 0.42548488, 0.40841345, 0.47393027,
                0.60882106, 0.45961208, 0.43371424, 0.40876484, 0.64367337,
                0.54092033, 0.34240811, 0.44048106, 0.48874236, 0.68300902,
                0.33563968, 0.58328107, 0.58054283, 0.64710522, 0.37801285,
                0.36748982, 0.44386445, 0.47245989, 0.297599, 0.50295541,
                0.39785732, 0.51370486, 0.46650358, 0.5623638, 0.4446957,
                0.52949791, 0.54611411, 0.41020067, 0.61644868, 0.47493691,
                0.50611458, 0.42518211, 0.45467712, 0.52438467, 0.724529,
                0.59749142, 0.45940223, 0.53099928, 0.65159718, 0.38038268,
                0.51639554, 0.41847437, 0.46022878, 0.57326103, 0.44913632,
                0.61043611, 0.42694949, 0.43997814, 0.58787928, 0.36252603,
                0.50937634, 0.47444256, 0.57992527, 0.29381335, 0.50357977,
                0.42469464, 0.53049697, 0.7163579, 0.39741694, 0.41980533,
                0.68091159, 0.69330702, 0.50518926, 0.55884098, 0.48618324,
                0.48469854, 0.55342267, 0.67159111, 0.62352006, 0.34773486};


        var dataset = new HistogramDataset();
        dataset.addSeries("key", vals, 50);

        JFreeChart histogram = ChartFactory.createHistogram("Normal distribution",
                "y values", "x values", dataset);

        ChartUtils.saveChartAsPNG(new File("histogram.png"), histogram, 450, 400);
    }
}
```

In the example, we generate a histogram for a random distribution of values. The chart is generated into a PNG file and saved on the disk.

```
var dataset = new HistogramDataset();
dataset.addSeries("key", vals, 50);
```

Histogram chart displays values in a `HistogramDataset`. The second argument of the `addSeries()` method is the array of double values. The third is the number of bins, i.e. the number of rectangles drawn on the chart.

```
JFreeChart histogram = ChartFactory.createHistogram("Normal distribution",
    "y values", "x values", dataset);
```

A histogram is created with the `ChartFactory.createHistogram()` method.

```
ChartUtils.saveChartAsPNG(new File("histogram.png"), histogram, 450, 400);
```

We write the chart to a PNG image with `ChartUtils.saveChartAsPNG()`.

JFreeChart line chart

A line chart is a basic type of chart which displays information as a series of data points connected by straight line segments. A line chart in JavaFX is created with the `ChartFactory.createXYLineChart()`.

com/zetcode/LineChartEx.java
```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.block.BlockBorder;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;

public class LineChartEx extends JFrame {

    public LineChartEx() {

        initUI();
    }

    private void initUI() {

        XYDataset dataset = createDataset();
        JFreeChart chart = createChart(dataset);

        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
        chartPanel.setBackground(Color.white);
        add(chartPanel);

        pack();
        setTitle("Line chart");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private XYDataset createDataset() {

        var series = new XYSeries("2016");
        series.add(18, 567);
        series.add(20, 612);
        series.add(25, 800);
        series.add(30, 980);
        series.add(40, 1410);
        series.add(50, 2350);

        var dataset = new XYSeriesCollection();
        dataset.addSeries(series);

        return dataset;
    }

    private JFreeChart createChart(XYDataset dataset) {

        JFreeChart chart = ChartFactory.createXYLineChart(
                "Average salary per age",
                "Age",
                "Salary (€)",
                dataset,
                PlotOrientation.VERTICAL,
```

```
                true,
                true,
                false
        );

        XYPlot plot = chart.getXYPlot();

        var renderer = new XYLineAndShapeRenderer();
        renderer.setSeriesPaint(0, Color.RED);
        renderer.setSeriesStroke(0, new BasicStroke(2.0f));

        plot.setRenderer(renderer);
        plot.setBackgroundPaint(Color.white);

        plot.setRangeGridlinesVisible(true);
        plot.setRangeGridlinePaint(Color.BLACK);

        plot.setDomainGridlinesVisible(true);
        plot.setDomainGridlinePaint(Color.BLACK);

        chart.getLegend().setFrame(BlockBorder.NONE);

        chart.setTitle(new TextTitle("Average Salary per Age",
                        new Font("Serif", java.awt.Font.BOLD, 18)
                )
        );

        return chart;
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new LineChartEx();
            ex.setVisible(true);
        });
    }
}
```

In the example, we create a line chart showing average salary per age.

```
var series = new XYSeries("2016");
series.add(18, 567);
series.add(20, 612);
series.add(25, 800);
...
```

`XYSeries` represents a sequence of zero or more data items in the form (x, y).

```
var dataset = new XYSeriesCollection();
dataset.addSeries(series);
```

The series is added to the `XYSeriesCollection`, which is a collection of `XYSeries` objects that can be used as a dataset.

```
JFreeChart chart = ChartFactory.createXYLineChart(
        "Average salary per age",
        "Age",
        "Salary (€)",
        dataset,
        PlotOrientation.VERTICAL,
        true,
        true,
        false
);
```

The `ChartFactory.createXYLineChart()` creates a new line chart. The parameters of the method are: chart title, X axis label, Y axis label, data, plot orientation, and three flags indicating whether to show legend, tooltips, and URLs.

```
XYPlot plot = chart.getXYPlot();
```

We get a reference to the plot in order to customize it.

```
var renderer = new XYLineAndShapeRenderer();
renderer.setSeriesPaint(0, Color.RED);
renderer.setSeriesStroke(0, new BasicStroke(2.0f));
plot.setRenderer(renderer);
```

Here, we set a stroke and a colour for the line of the chart. `XYLineAndShapeRenderer` is an object that connects data points with lines and/or draws shapes at each data point. The renderer is set with the `setRenderer()` method.

```
plot.setBackgroundPaint(Color.white);
```

The `setBackgroundPaint()` sets the background colour of the plot area.

```
plot.setRangeGridlinesVisible(true);
plot.setRangeGridlinePaint(Color.BLACK);

plot.setDomainGridlinesVisible(true);
plot.setDomainGridlinePaint(Color.BLACK);
```

We show the grid lines and paint them in black colour.

```
chart.getLegend().setFrame(BlockBorder.NONE);
```

We remove the border around the legend.

```
chart.setTitle(new TextTitle("Average Salary per Age",
                new Font("Serif", java.awt.Font.BOLD, 18)
        )
);
```

We create a chart title with a new font.

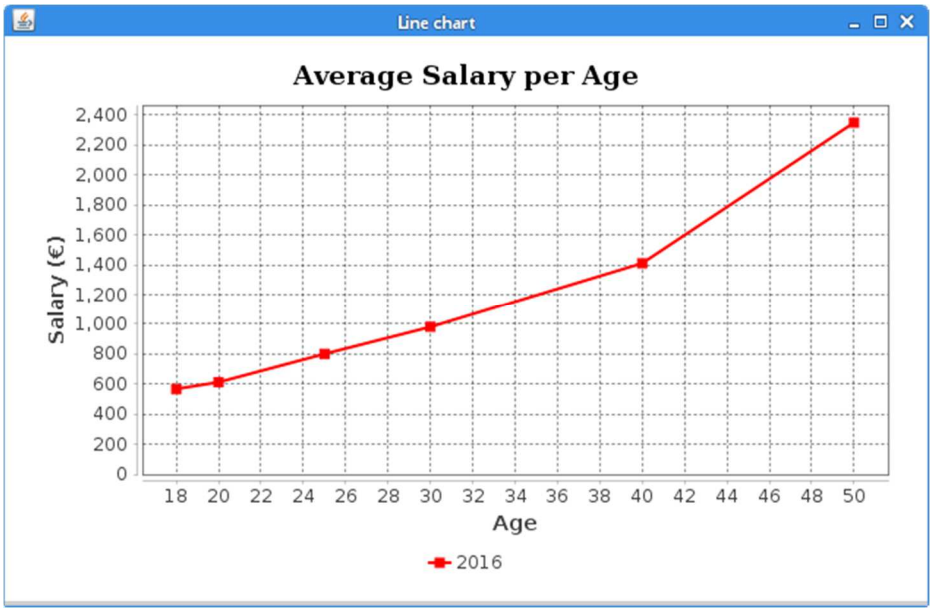

Figure: Line chart

A line chart with two series

In the second example, we create a line chart having two data series.

com/zetcode/LineChartEx2.java

```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.block.BlockBorder;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;

public class LineChartEx2 extends JFrame {

    public LineChartEx2() {

        initUI();
    }

    private void initUI() {

        XYDataset dataset = createDataset();
        JFreeChart chart = createChart(dataset);
        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
        chartPanel.setBackground(Color.white);

        add(chartPanel);

        pack();
        setTitle("Line chart");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private XYDataset createDataset() {

        var series1 = new XYSeries("2014");
        series1.add(18, 530);
        series1.add(20, 580);
        series1.add(25, 740);
        series1.add(30, 901);
        series1.add(40, 1300);
        series1.add(50, 2219);

        var series2 = new XYSeries("2016");
        series2.add(18, 567);
        series2.add(20, 612);
        series2.add(25, 800);
        series2.add(30, 980);
        series2.add(40, 1210);
        series2.add(50, 2350);

        var dataset = new XYSeriesCollection();
        dataset.addSeries(series1);
        dataset.addSeries(series2);

        return dataset;
    }
```

```
    private JFreeChart createChart(final XYDataset dataset) {

        JFreeChart chart = ChartFactory.createXYLineChart(
                "Average salary per age",
                "Age",
                "Salary (€)",
                dataset,
                PlotOrientation.VERTICAL,
                true,
                true,
                false
        );

        XYPlot plot = chart.getXYPlot();

        var renderer = new XYLineAndShapeRenderer();

        renderer.setSeriesPaint(0, Color.RED);
        renderer.setSeriesStroke(0, new BasicStroke(2.0f));
        renderer.setSeriesPaint(1, Color.BLUE);
        renderer.setSeriesStroke(1, new BasicStroke(2.0f));

        plot.setRenderer(renderer);
        plot.setBackgroundPaint(Color.white);
        plot.setRangeGridlinesVisible(false);
        plot.setDomainGridlinesVisible(false);

        chart.getLegend().setFrame(BlockBorder.NONE);

        chart.setTitle(new TextTitle("Average Salary per Age",
                        new Font("Serif", Font.BOLD, 18)
                )
        );

        return chart;
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new LineChartEx2();
            ex.setVisible(true);
        });
    }
}
```

The example draws a line chart with two data series.

```
var series1 = new XYSeries("2014");
series1.add(18, 530);
series1.add(20, 580);
series1.add(25, 740);
...
```

We create the first series; it contains data for 2014.

```
var series2 = new XYSeries("2016");
series2.add(18, 567);
series2.add(20, 612);
series2.add(25, 800);
...
```

A second data series is created; it contains data for 2016.

```
var dataset = new XYSeriesCollection();
dataset.addSeries(series1);
dataset.addSeries(series2);
```

The series are added to the `XYSeriesCollection` with the `addSeries()` method.

```
renderer.setSeriesPaint(0, Color.RED);
renderer.setSeriesStroke(0, new BasicStroke(2.0f));

renderer.setSeriesPaint(1, Color.BLUE);
renderer.setSeriesStroke(1, new BasicStroke(2.0f));
```

One line is painted in red colour and one in blue.

```
plot.setRangeGridlinesVisible(false);
plot.setDomainGridlinesVisible(false);
```

The grid lines are turned off.

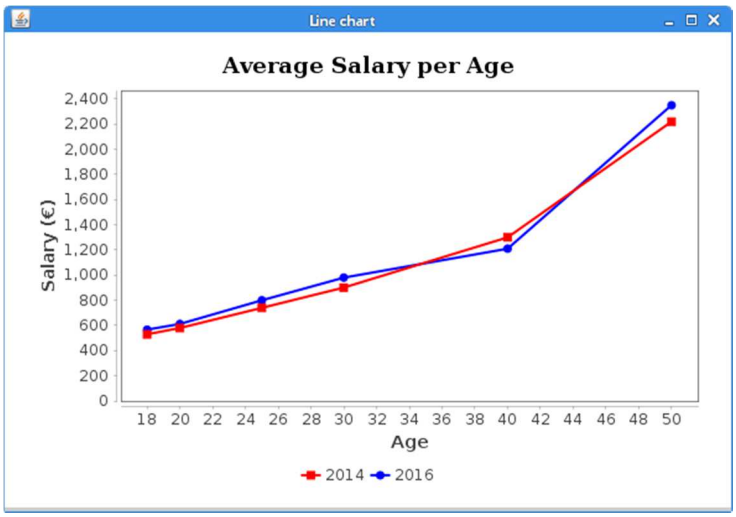

Figure: Line chart with two series

Saving line chart to image

`ChartUtils` is a collection of utility methods for JFreeChart. It includes methods for converting charts to image formats and creating simple HTML image maps.

com/zetcode/LineChartToPNGEx.java

```
package com.zetcode;

import java.io.File;
import java.io.IOException;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtils;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

public class LineChartToPNGEx {

    public static void main(String[] args) throws IOException {

        var series1 = new XYSeries("2014");
        series1.add(18, 530);
        series1.add(20, 580);
        series1.add(25, 740);
        series1.add(30, 901);
        series1.add(40, 1300);
        series1.add(50, 2219);

        var dataset = new XYSeriesCollection();
        dataset.addSeries(series1);

        JFreeChart chart = ChartFactory.createXYLineChart(
                "Average salary per age",
                "Age",
                "Salary (€)",
                dataset,
                PlotOrientation.VERTICAL,
                true,
                true,
                false
        );

        ChartUtils.saveChartAsPNG(new File("line_chart.png"), chart, 450, 400);
    }
}
```

The example creates a line chart and saves it into a PNG file.

```
ChartUtils.saveChartAsPNG(new File("line_chart.png"), chart, 450, 400);
```

The `ChartUtils.saveChartAsPNG()` saves a chart to the specified file in PNG format.

JFreeChart area chart

An area chart displays graphically quantitative data that change over time. An area chart is created with `ChartFactory.createAreaChart()` method in JFreeChart.

com/zetcode/AreaChartEx.java

```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.chart.renderer.AreaRendererEndType;
import org.jfree.chart.renderer.category.AreaRenderer;
import org.jfree.chart.title.TextTitle;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.general.DatasetUtils;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Font;
```

```
public class AreaChartEx extends JFrame {

    public AreaChartEx() {

        initUI();
    }

    private void initUI() {

        CategoryDataset dataset = createDataset();

        JFreeChart chart = createChart(dataset);
        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
        chartPanel.setBackground(Color.white);
        add(chartPanel);

        pack();
        setTitle("Area chart");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private CategoryDataset createDataset() {

        double[][] data = new double[][]{
                {82502, 84026, 85007, 86216, 85559, 84491, 87672,
                        88575, 89837, 90701}
        };

        CategoryDataset dataset = DatasetUtils.createCategoryDataset(
                new String[]{"Oil"}, new String[]{"2004", "2005", "2006",
                        "2007", "2008", "2009", "2010", "2011", "2012", "2013"},
                data
        );

        return dataset;
    }

    private JFreeChart createChart(CategoryDataset dataset) {

        JFreeChart chart = ChartFactory.createAreaChart(
                "Oil consumption",
                "Time",
                "Thousands bbl/day",
                dataset,
                PlotOrientation.VERTICAL,
                false,
                true,
                true
        );

        CategoryPlot plot = (CategoryPlot) chart.getPlot();
        plot.setForegroundAlpha(0.3f);

        AreaRenderer renderer = (AreaRenderer) plot.getRenderer();
        renderer.setEndType(AreaRendererEndType.LEVEL);

        chart.setTitle(new TextTitle("Oil consumption",
                new Font("Serif", java.awt.Font.BOLD, 18))
        );

        return chart;
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new AreaChartEx();
            ex.setVisible(true);
        });
    }
}
```

The example shows an area chart showing world crude oil consumption by year.

```
double[][] data = new double[][]{
    {82502, 84026, 85007, 86216, 85559, 84491, 87672,
        88575, 89837, 90701}
};

CategoryDataset dataset = DatasetUtils.createCategoryDataset(
    new String[]{"Oil"}, new String[]{"2004", "2005", "2006",
            "2007", "2008", "2009", "2010", "2011", "2012", "2013"},
    data
);
```

A dataset is created with the `DatasetUtils.createCategoryDataset()` method. A category dataset values associated with categories. In our examples, we have years associated with oil consumption.

```
CategoryPlot plot = (CategoryPlot) chart.getPlot();
plot.setForegroundAlpha(0.3f);
```

We mave the chart transparent with the `setForegroundAlpha()` method.

```
AreaRenderer renderer = (AreaRenderer) plot.getRenderer();
renderer.setEndType(AreaRendererEndType.LEVEL);
```

We adjust the endings of the chart.

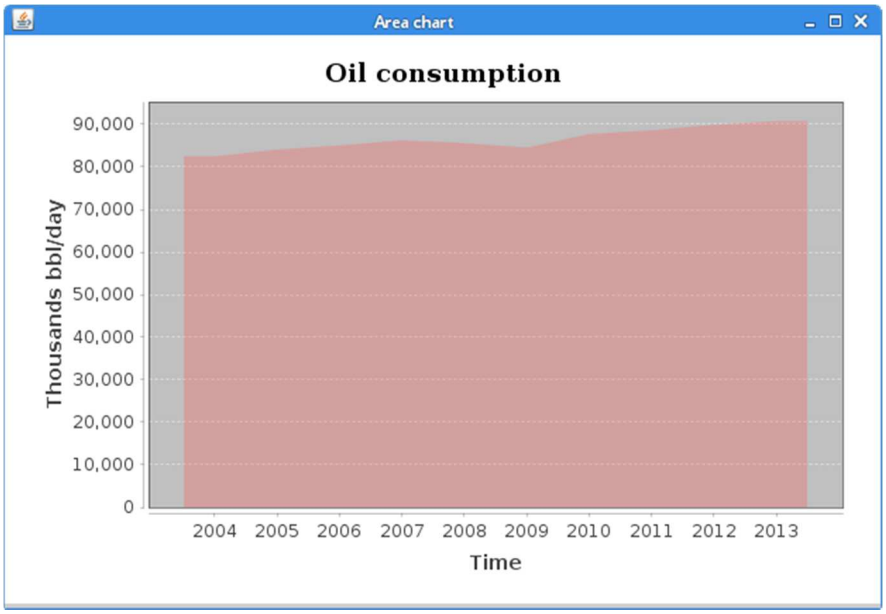

Figure: Area chart

JFreeChart bar chart

A bar chart presents grouped data with rectangular bars with lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

com/zetcode/BarChartEx.java

```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
```

```
import javax.swing.BorderFactory;
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.EventQueue;

public class BarChartEx extends JFrame {

    public BarChartEx() {

        initUI();
    }

    private void initUI() {

        CategoryDataset dataset = createDataset();

        JFreeChart chart = createChart(dataset);
        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
        chartPanel.setBackground(Color.white);
        add(chartPanel);

        pack();
        setTitle("Bar chart");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private CategoryDataset createDataset() {

        var dataset = new DefaultCategoryDataset();
        dataset.setValue(46, "Gold medals", "USA");
        dataset.setValue(38, "Gold medals", "China");
        dataset.setValue(29, "Gold medals", "UK");
        dataset.setValue(22, "Gold medals", "Russia");
        dataset.setValue(13, "Gold medals", "South Korea");
        dataset.setValue(11, "Gold medals", "Germany");

        return dataset;
    }

    private JFreeChart createChart(CategoryDataset dataset) {

        JFreeChart barChart = ChartFactory.createBarChart(
                "Olympic gold medals in London",
                "",
                "Gold medals",
                dataset,
                PlotOrientation.VERTICAL,
                false, true, false);

        return barChart;
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new BarChartEx();
            ex.setVisible(true);
        });
    }
}
```

The code example uses a bar chart to show the number of Olympic gold medals per country in London 2012.

```
var dataset = new DefaultCategoryDataset();
dataset.setValue(46, "Gold medals", "USA");
dataset.setValue(38, "Gold medals", "China");
dataset.setValue(29, "Gold medals", "UK");
dataset.setValue(22, "Gold medals", "Russia");
dataset.setValue(13, "Gold medals", "South Korea");
dataset.setValue(11, "Gold medals", "Germany");
```

We use a `DefaultCategoryDataset` to create a dataset.

```
JFreeChart barChart = ChartFactory.createBarChart(
        "Olympic gold medals in London",
        "",
        "Gold medals",
        dataset,
        PlotOrientation.VERTICAL,
        false, true, false);
```

A bar chart is created with the `ChartFactory.createBarChart()` method.

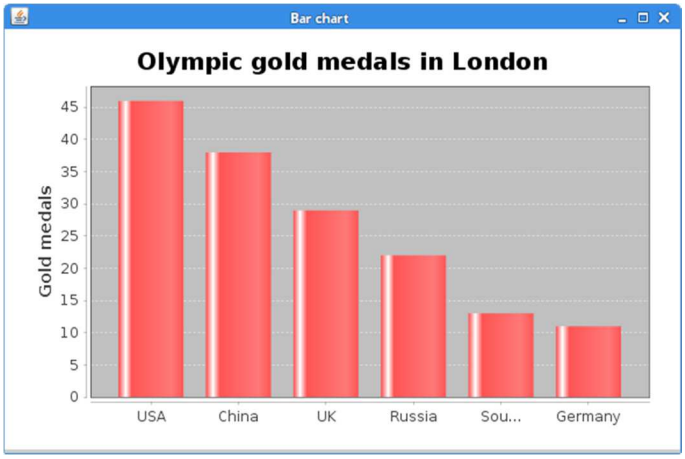

Figure: Bar chart

JFreeChart pie chart

A pie chart is a circular chart which is divided into slices to illustrate numerical proportion. A pie chart is created with the `ChartFactory.createPieChart()` method in JFreeChart.

com/zetcode/PieChartEx.java
```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.data.general.DefaultPieDataset;

import javax.swing.BorderFactory;
import javax.swing.JFrame;
import java.awt.Color;
import java.awt.EventQueue;

public class PieChartEx extends JFrame {

    public PieChartEx() {

        initUI();
    }

    private void initUI() {

        DefaultPieDataset dataset = createDataset();

        JFreeChart chart = createChart(dataset);
        ChartPanel chartPanel = new ChartPanel(chart);
        chartPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
        chartPanel.setBackground(Color.white);
        add(chartPanel);

        pack();
        setTitle("Pie chart");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```

```
    private DefaultPieDataset createDataset() {

        var dataset = new DefaultPieDataset();
        dataset.setValue("Apache", 52);
        dataset.setValue("Nginx", 31);
        dataset.setValue("IIS", 12);
        dataset.setValue("LiteSpeed", 2);
        dataset.setValue("Google server", 1);
        dataset.setValue("Others", 2);

        return dataset;
    }

    private JFreeChart createChart(DefaultPieDataset dataset) {

        JFreeChart pieChart = ChartFactory.createPieChart(
                "Web servers market share",
                dataset,
                false, true, false);

        return pieChart;
    }

    public static void main(String[] args) {

        EventQueue.invokeLater(() -> {

            var ex = new PieChartEx();
            ex.setVisible(true);
        });
    }
}
```

The example uses a pie chart to show the market share of web servers.

```
var dataset = new DefaultPieDataset();
dataset.setValue("Apache", 52);
dataset.setValue("Nginx", 31);
dataset.setValue("IIS", 12);
...
```

A `DefaultPieDataset` is used to create a dataset.

```
JFreeChart barChart = ChartFactory.createPieChart(
        "Web servers market share",
        dataset,
        false, true, false);
```

A new pie chart is created with the `ChartFactory.createPieChart()` method.

JFreeChart in a servlet

In the following example, we use a Java servlet to create a pie chart. The chart is rendered in a web browser.

com/zetcode/web/DoChart.java

```
package com.zetcode.web;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtils;
import org.jfree.chart.JFreeChart;
import org.jfree.data.general.DefaultPieDataset;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.OutputStream;
```

```
@WebServlet(name = "DoChart", urlPatterns = {"/DoChart"})
public class DoChart extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws IOException {

        response.setContentType("image/png");

        OutputStream outputStream = response.getOutputStream();

        JFreeChart chart = getChart();
        int width = 500;
        int height = 350;

        ChartUtils.writeChartAsPNG(outputStream, chart, width, height);
    }

    public JFreeChart getChart() {

        var dataset = new DefaultPieDataset();
        dataset.setValue("Croatia", 22);
        dataset.setValue("Bohemia", 34);
        dataset.setValue("Bulgaria", 18);
        dataset.setValue("Spain", 5);
        dataset.setValue("Others", 21);

        JFreeChart chart = ChartFactory.createPieChart("Popular destinations",
                dataset, true, false, false);

        chart.setBorderVisible(false);

        return chart;
    }
}
```

The `DoChart` servlet creates a pie chart and sends it to the client.

```
response.setContentType("image/png");
```

The `setContentType()` sets the content to a PNG image.

```
OutputStream outputStream = response.getOutputStream();
```

With the `getOutputStream()` method we get an output stream. It is a tunnel to which we send the data.

```
ChartUtils.writeChartAsPNG(outputStream, chart, width, height);
```

The `ChartUtils.writeChartAsPNG()` writes the binary data to the outputstrem.

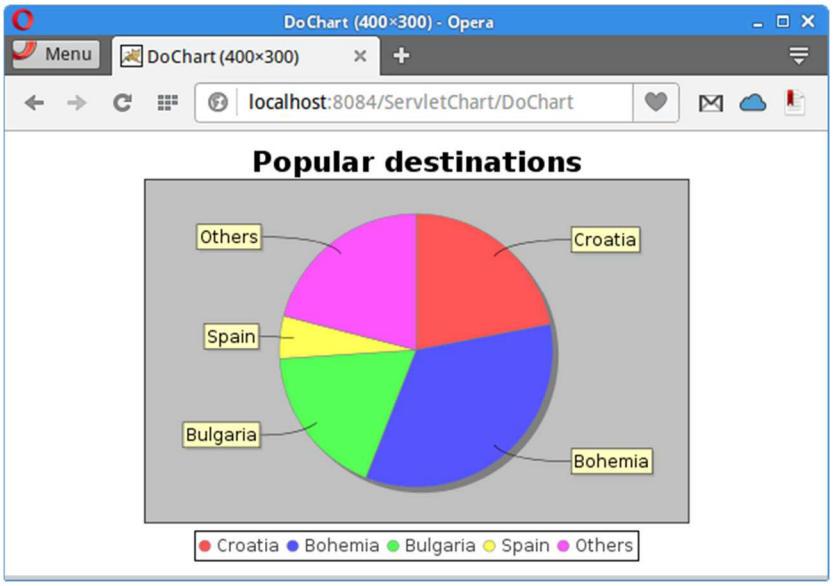

Figure: A pie chart in a browser

Showing data from a MySQL database

`JDBCCategoryDataset` is a `CategoryDataset` implementation over a database JDBC result set. The dataset is populated via a call to `executeQuery()` with the string SQL query.

medals.sql
```
DROP TABLE IF EXISTS GoldMedalsLondon;

CREATE TABLE GoldMedalsLondon (
  Id int(11) NOT NULL AUTO_INCREMENT,
  Country text,
  Medals int(11) DEFAULT NULL,
  PRIMARY KEY (Id)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;

LOCK TABLES GoldMedalsLondon WRITE;
INSERT INTO GoldMedalsLondon VALUES (1,'USA',46),(2,'China',38),(3,'UK',29),
  (4,'Russia',22),(5,'South Korea',13),(6,'Germany',11);
UNLOCK TABLES;
```

We have this data in a MySQL database table.

```
mysql> SELECT * FROM GoldMedalsLondon;
+----+-------------+--------+
| Id | Country     | Medals |
+----+-------------+--------+
|  1 | USA         |     46 |
|  2 | China       |     38 |
|  3 | UK          |     29 |
|  4 | Russia      |     22 |
|  5 | South Korea |     13 |
|  6 | Germany     |     11 |
+----+-------------+--------+
6 rows in set (0.00 sec)
```

We show the data with the `mysql` tool.

com/zetcode/MySQLChartEx.java
```
package com.zetcode;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtils;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
```

```
import org.jfree.data.jdbc.JDBCCategoryDataset;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySQLChartEx {

    private static JDBCCategoryDataset dataset;

    public static void main(String[] args) throws IOException, SQLException {

        String url = "jdbc:mysql://localhost:3306/testdb";
        String user = "root";
        String password = "andrea";

        try (Connection con = DriverManager.getConnection(url, user, password)) {

            dataset = new JDBCCategoryDataset(con);
            dataset.executeQuery("SELECT Country, Medals FROM GoldMedalsLondon");
        }

        JFreeChart barChart = ChartFactory.createBarChart(
                "Olympic Gold medals in London",
                "",
                "Gold medals",
                dataset,
                PlotOrientation.VERTICAL,
                false, true, false);

        ChartUtils.saveChartAsPNG(new File("medals.png"), barChart, 450, 400);
    }
}
```

The example retrieves data from a MySQL table, creates a bar chart, and saves it into a PNG image.

```
try (Connection con = DriverManager.getConnection(url, user, password)) {

    dataset = new JDBCCategoryDataset(con);
    dataset.executeQuery("SELECT Country, Medals FROM GoldMedalsLondon");
}
```

`JDBCCategoryDataset` is created; it takes a database connection as a parameter.

```
dataset.executeQuery("SELECT Country, Medals FROM GoldMedalsLondon");
```

The `executeQuery()` populates the dataset by executing the supplied query against the existing database connection. The SQL query must return at least two columns. The first column is the category name and remaining columns are values.

```
JFreeChart barChart = ChartFactory.createBarChart(
        "Olympic Gold medals in London",
        "",
        "Gold medals",
        dataset,
        PlotOrientation.VERTICAL,
        false, true, false);
```

The bar chart is created.

```
ChartUtils.saveChartAsPNG(new File("medals.png"), barChart, 450, 400);\
```

The bar chart is saved to a PNG file with the `ChartUtils.saveChartAsPNG()` method.