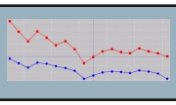
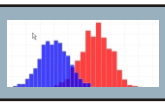
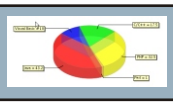
JFreeChart Tutorial

Matthew D'Andrea | E-mail: matthew.dandrea@utoronto.ca

Uses

JFreeChart is an open source library available for Java that allows users to easily generate graphs and charts. It is particularly effective for when a user needs to regenerate graphs that change on a frequent basis (as required for the CSC408 course project).

This tutorial will examine how to create images from these graphs which can then be displayed on web pages.

Download Location

JFreeChart is available for download from:
http://sourceforge.net/project/showfiles.php?group_id=15494&package_id=12428
I recommend downloading either **1.0.0-pre1** or **Version 0.9.21**.

Unpack JFreeChart into a temporary directory.

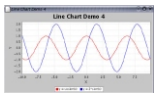
How To Install / Get Started

I will briefly outline the steps needed to setup a Java project in Eclipse so that you can create your own JFreeChart graphs.
You will then be able to test out the code provided in the "**Example Graphs**" section.
Steps:
1. Start Eclipse 3.0. (If you do not have Eclipse installed, visit **www.eclipse.org**.)
2. Go to File -> New -> Project....
3. Select "Java Project" and click on Next. Enter in a project name and click Finish.
4. Select the project in the Package Explorer view. Go to Project -> Properties.
5. Click on "Java Build Path" on the left-hand side of the dialog box and on the right-hand side click on the "Libraries" tab.
6. Click on "Add External JARs..." and locate **jfreechart-1.0.0-pre1.jar** and **jcommon-1.0.0-common.jar**. Click on OK.
7. Select the project in the Package Explorer view. Go to File -> New -> Package. Give the package a name and click on Finish.
8. Select the newly-created package in the Package Explorer view. Go to File -> New -> Class. Give the class a name and click on Finish.
9. You are now ready to start using the JFreeChart library!

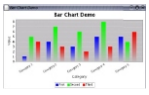
Types of Graphs Constructed with JFreeChart

We will consider the following graphs that can be created using JFreeChart:
- Pie Chart

- XY Chart



- Bar Chart



- Time Series Chart

Example Graphs

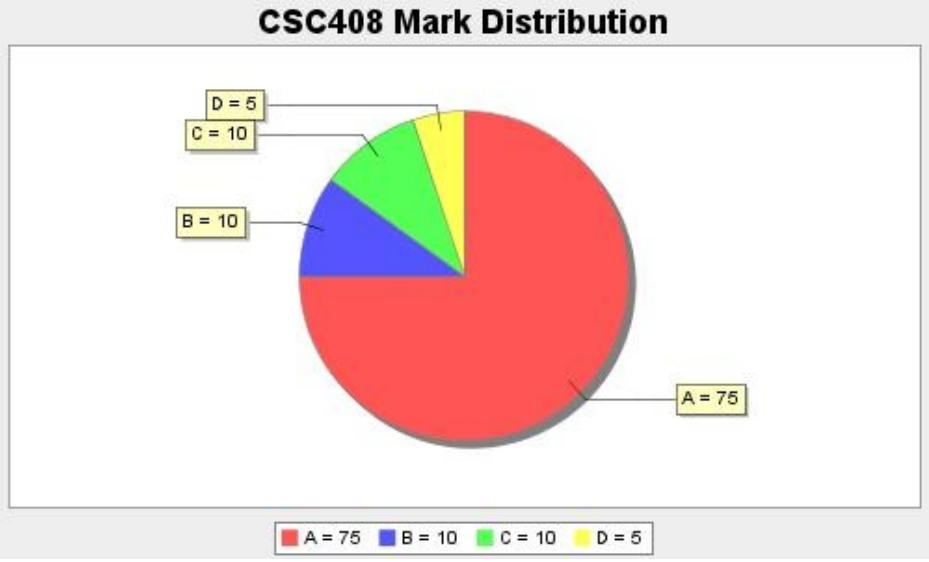
1. Pie Chart Example

Suppose that we want to construct a Pie Chart that reflects the percentage of marks that are in the A, B, C, and D range for CSC408. (You'll notice that the distribution is rather hopeful. :))

Code:

```
package main;

import org.jfree.chart.JFreeChart;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.ChartFactory;
import org.jfree.data.general.DefaultPieDataset;
import java.io.File;

public class PieChartExample {
    public static void main(String[] args) {
        // Create a simple pie chart
        DefaultPieDataset pieDataset = new DefaultPieDataset();
        pieDataset.setValue("A", new Integer(75));
        pieDataset.setValue("B", new Integer(10));
        pieDataset.setValue("C", new Integer(10));
        pieDataset.setValue("D", new Integer(5));

        JFreeChart chart = ChartFactory.createPieChart
            ("CSC408 Mark Distribution",   // Title
             pieDataset,                   // Dataset
             true,                         // Show legend
             true,                         // Use tooltips
             false                         // Configure chart to generate URLs?
             );

        try {
            ChartUtilities.saveChartAsJPEG(new File("C:\\chart.jpg"), chart, 500, 300);
        } catch (Exception e) {
            System.out.println("Problem occurred creating chart.");
        }
    }
}
```

Graph:



Explanation:

To define a data set for a pie chart we create an instance of type `DefaultPieDataSet`.

```
DefaultPieDataset pieDataset = new DefaultPieDataset();
```

The setValue() method is used to set the name of a piece of the pie (e.g. “A”) and its corresponding percentage (e.g. 75) value.

```
PieDataset.setValue(’A”, new Integer(75));
```

A graph object of type `JFreeChart` is generated using one of the `ChartFactory` methods and passing the data set as one of the arguments. In this case, we use the createPieChart() method to produce a pie chart.
Modification: Use createPieChart3D() to produce a corresponding 3D version of the pie chart.

```
JFreeChart chart = ChartFactory.createPieChart3D(”CSC408 Mark Distribution”, pieDataset,
true, true, false);
```

A JPEG image of the graph is produced using the `ChartUtilities` method saveChartAsJPEG().

```
ChartUtilities.saveChartAsJPEG(new File(”C:\\chart.jpg”), chart, 500, 300);
```

Modification: To generate a PNG image use the method saveChartAsPNG().

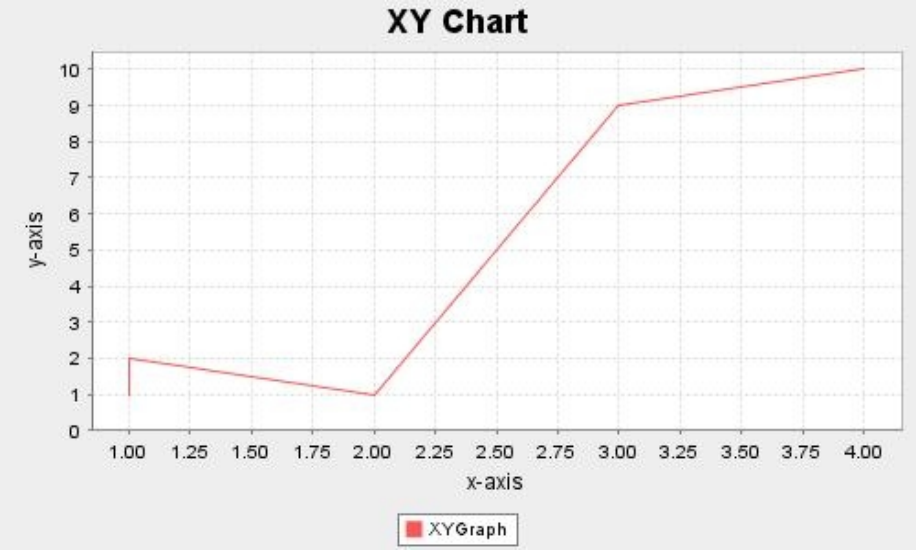
2. XY Chart Example

Suppose that we want to construct a line with the following set of (x, y) coordinates:
{(1, 1), (1, 2), (2, 1), (3, 9), (4, 10)}.

Code:

```
public class XYChartExample {
    public static void main(String[] args) {
        // Create a simple XY chart
        XYSeries series = new XYSeries("XYGraph");
        series.add(1, 1);
        series.add(1, 2);
        series.add(2, 1);
        series.add(3, 9);
        series.add(4, 10);

        // Add the series to your data set
        XYSeriesCollection dataset = new XYSeriesCollection();
        dataset.addSeries(series);

        // Generate the graph
        JFreeChart chart = ChartFactory.createXYLineChart(
            "XY Chart",                 // Title
            "x-axis",                   // x-axis Label
            "y-axis",                   // y-axis Label
            dataset,                    // Dataset
            PlotOrientation.VERTICAL,   // Plot Orientation
            true,                       // Show Legend
            true,                       // Use tooltips
            false                       // Configure chart to generate URLs?
            );

        try {
            ChartUtilities.saveChartAsJPEG(new File("C:\\chart.jpg"), chart, 500, 300);
        } catch (IOException e) {
            System.err.println("Problem occurred creating chart.");
        }
    }
}
```

Graph:



Explanation:

To define a set of (x, y) coordinates, use an object of class `XYSeries`.

```
XYSeries series = new XYSeries("XYGraph");
```

Add the series to your data set of type `XYSeriesCollection`.

```
dataset.addSeries(series);
```

Note: Multiple series (lines) can be added to one data set. Use this feature if you want more than one line to appear on the same graph.

A graph object of type `JFreeChart` is generated using the `ChartFactory` method createXYLineChart(). Note the extra arguments that did not appear for a pie chart. The 2nd and 3rd arguments specify the label for the x and y axes.

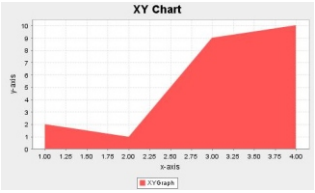
Meaning of Plot Orientation:

`PlotOrientation.VERTICAL` - y-axis is displayed as the vertical axis

`PlotOrientation.HORIZONTAL` - x-axis becomes the vertical axis

Modification:

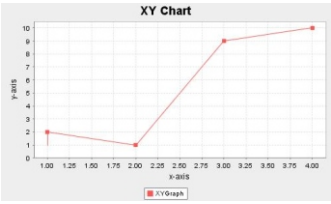
In the current code, we used the createXYLineChart() method to display a line of data in the graph. As an alternative, the method createXYAreaChart() could be used to display the area under the line of data.



Change to Area Chart

It is possible to modify the graph to show each of the 5 data points:

```
XYItemRenderer rend = chart.getXYPlot().getRenderer();
StandardXYItemRenderer rr = (StandardXYItemRenderer)rend;
rr.setPlotShapes(true);
```



Mark data points

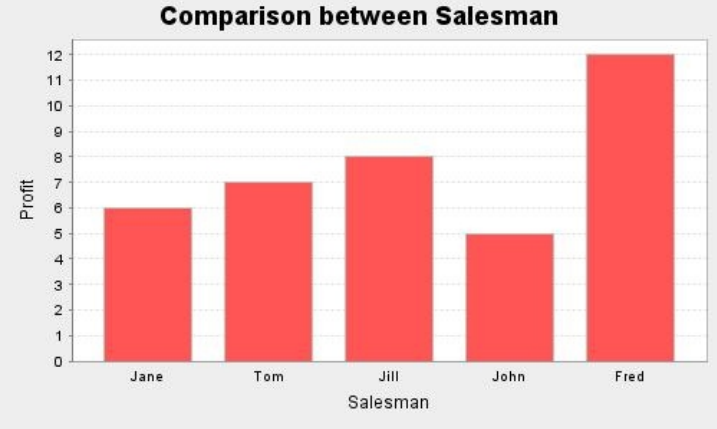
3. Bar Chart Example

Suppose that we want to construct a bar graph which compares the profits taken in by the following salesmen: Jane, Tom, Jill, John, Fred.

Code:

```
public class BarChartExample {
    public static void main(String[] args) {
        // Create a simple Bar chart
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.setValue(6, "Profit", "Jane");
        dataset.setValue(7, "Profit", "Tom");
        dataset.setValue(8, "Profit", "Jill");
        dataset.setValue(5, "Profit", "John");
        dataset.setValue(12, "Profit", "Fred");
        JFreeChart chart = ChartFactory.createBarChart("Comparison between Salesman",
        "Salesman", "Profit", dataset, PlotOrientation.VERTICAL,
        false, true, false);

        try {
            ChartUtilities.saveChartAsJPEG(new File("C:\\chart.jpg"), chart, 500, 300);
        } catch (IOException e) {
            System.err.println("Problem occurred creating chart.");
        }
    }
}
```

Graph:



Explanation:

To define a data set for a bar graph use an object of class `DefaultCategoryDataset`.
`DefaultCategoryDataset dataset = new DefaultCategoryDataset();`

Values can be added to the data set using the setValue() method.
`dataset.setValue(6, "Profit", "Jane");`
The first argument specifies the level of profit achieved by Jane. The second argument specifies what will appear in the legend for the meaning of a bar.

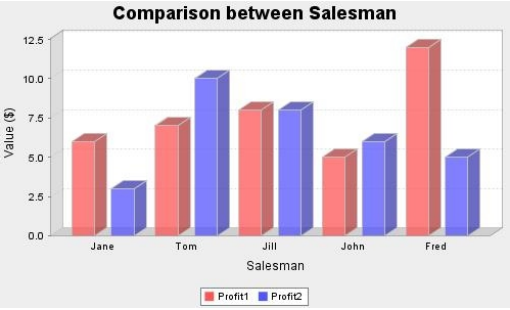
To generate a bar graph object of class `JFreeChart`, the method createBarChart() of `ChartFactory` is used. It takes the same set of arguments as that required by createXYLineChart(). The 1st argument denotes the title of the graph, the second the label for the x-axis, the third the label for the y-axis.
`JFreeChart chart = ChartFactory.createBarChart("Comparison between Salesman", "Salesman", "Profit", dataset, PlotOrientation.VERTICAL, false, true, false);`
Modification: As was the case with pie charts, it is possible to display the bars in 3D using the createBarChart3D() method.

Modification:

It is possible to introduce more than one set of bars to the same graph.
This can be introduced with the following modification:

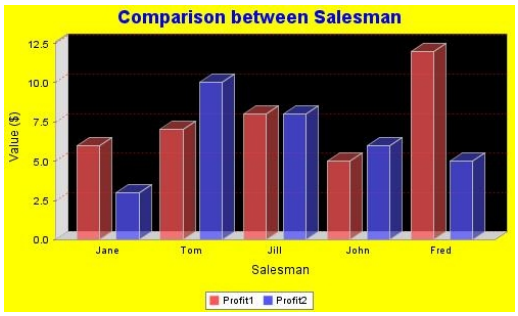
```
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.setValue(6, "Profit1", "Jane");
dataset.setValue(3, "Profit2", "Jane");
dataset.setValue(7, "Profit1", "Tom");
dataset.setValue(10, "Profit2", "Tom");
dataset.setValue(8, "Profit1", "Jill");
dataset.setValue(8, "Profit2", "Jill");
dataset.setValue(5, "Profit1", "John");
dataset.setValue(6, "Profit2", "John");
dataset.setValue(12, "Profit1", "Fred");
dataset.setValue(5, "Profit2", "Fred");
// Profit1, Profit2 represent the row keys
// Jane, Tom, Jill, etc. represent the column keys

JFreeChart chart = ChartFactory.createBarChart3D( "Comparison between Salesman",
"Salesman", "Value ($)", dataset, PlotOrientation.VERTICAL, true, true, false );
```

Add extra bar



One thing that might be worthwhile is to adjust the appearance of the graph (e.g. colour).

```
chart.setBackgroundPaint(Color.yellow);     // Set the background colour of the chart
chart.getTitle().setPaint(Color.blue);      // Adjust the colour of the title
CategoryPlot p = chart.getCategoryPlot();   // Get the Plot object for a bar graph
p.setBackgroundPaint(Color.black);          // Modify the plot background
p.setRangeGridlinePaint(Color.red);         // Modify the colour of the plot gridlines
```



Modify Chart Appearance

4. Time Series Example

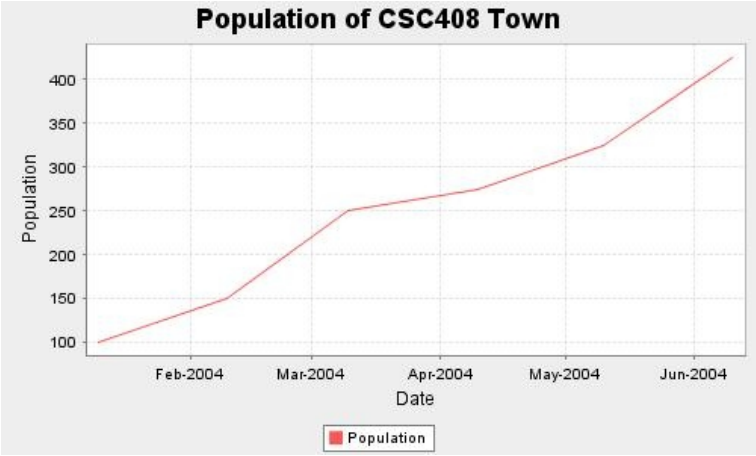
Suppose that we want to monitor the population of a small town over monthly intervals.

Code:

```
public class TimeSeriesExample {
    public static void main(String[] args) {
        // Create a time series chart
        TimeSeries pop = new TimeSeries("Population", Day.class);
        pop.add(new Day(10, 1, 2004), 100);
        pop.add(new Day(10, 2, 2004), 150);
        pop.add(new Day(10, 3, 2004), 250);
        pop.add(new Day(10, 4, 2004), 275);
        pop.add(new Day(10, 5, 2004), 325);
```

```
        pop.add(new Day(10, 6, 2004), 425);

        TimeSeriesCollection dataset = new TimeSeriesCollection();
        dataset.addSeries(pop);

        JFreeChart chart = ChartFactory.createTimeSeriesChart(
            "Population of CSC408 Town",
            "Date",
            "Population",
            dataset,
            true,
            true,
            false);

        try {
            ChartUtilities.saveChartAsJPEG(new File("C:\\chart.jpg"), chart, 500, 300);
        } catch (IOException e) {
            System.err.println("Problem occurred creating chart.");
        }
    }
}
```

Graph:



Explanation:

A Time Series represents a single set of values over some time period.
To define a data set for a time series graph use an object of class `TimeSeries`.
The second argument to the constructor defines the type of time period that this series represents.
Valid values are classes that extend `org.jfree.data.time.RegularTimePeriod` and can be found under the `org.jfree.data.time` package (e.g. `org.jfree.data.time.Day`).

```
TimeSeries pop = new TimeSeries("Population", Day.class);
```

JFreeChart graphs plot data sets that implement `org.jfree.data.general.DataSet`.
For Time Series charts, there are x and y axes, so we need to construct an `XYDataset`.
Class `TimeSeriesCollection` implements the `XYDataset` interface.

```
TimeSeriesCollection dataset = new TimeSeriesCollection();
```

The addSeries() method is used to add a `TimeSeries` to the collection.

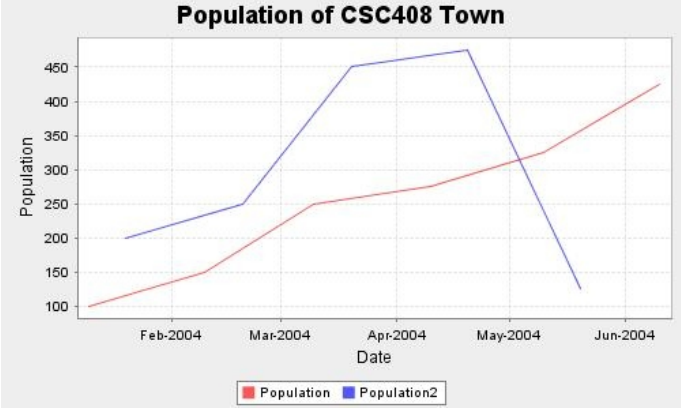
```
dataset.addSeries(pop);
```

Modification:

Like with bar graphs and XY charts, it is possible to plot two different `TimeSeries` instances on the same chart.

```
TimeSeries pop2 = new TimeSeries("Population2", Day.class);
pop2.add(new Day(20, 1, 2004), 200);
pop2.add(new Day(20, 2, 2004), 250);
pop2.add(new Day(20, 3, 2004), 450);
pop2.add(new Day(20, 4, 2004), 475);
pop2.add(new Day(20, 5, 2004), 125);
dataset.addSeries(pop2);
```

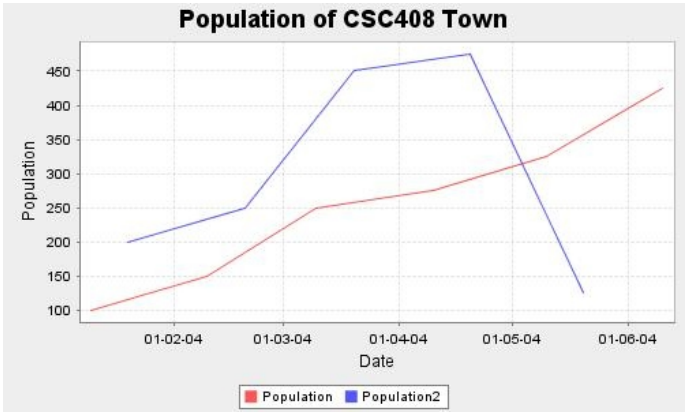
Add extra time line



We can override the display format of the date on the domain axis by casting it to a `DateAxis`.

```
XYPlot plot = chart.getXYPlot();
DateAxis axis = (DateAxis) plot.getDomainAxis();
axis.setDateFormatOverride(new SimpleDateFormat("dd-MM-yyyy"));
```

Date Modification



References

http://www.jfree.org/jfreechart/jfreechart-0.9.21-install.pdf
http://www.javaworld.com/javaworld/jw-12-2002/jw-1227-opensourceprofile.html
http://www.informit.com/guides/content.asp?g=java&seqNum=74
http://tools.devchannel.org/devtoolschannel/04/03/12/1634222.shtml