NetBeans, Java, and JFreeChart

An introduction for the complete novice (Part 1)

Carl David
Carl.David@uconn.edu

Introduction

This set of "papers" consists of an introduction to JFreeChart and NetBeans as an IDE for Java programming. I attempt herein to ease the learning burden, a little, compared to the misery that I had to suffer through. We are using JDK 1.5.0_03, NetBeans 4.1 (for future work, we've gone over to 5.0), and jfreechart-1.0.0-rc1. I hope its not necessary to mention that all of these programs are free, in the public domain, and can be downloaded and installed on modern PCs.

Please, if you read this and find mistakes, please e-mail me at Carl.David@uconn.edu and I will correct them. Everything you read here has been achieved created by hacking through every piece of material I can find on the WWW concerning the topics being discussed. Orginality is explicitly eschewed. Stealing code from every legal source is, *a posteriori*, acceptable[1].

Starting a New Project

To start a project, left-click File→New Project. Then, after following the menus and naming the project, creating a Main application, etc., you can start coding into this Main application, and use the various build and clean functions, but if you choose to create named Applications, then, inside the project, one can right-click on the project node to get a menu, choose New, and then look for a kind of form which makes sense. In this case, I want a graphical application with swing, and with a graphical part, i.e., one which, when it comes up on the screen, I want it to have a drawing face, perhaps a menu, etc., choose Java GUI Forms→Sample Forms→Application. Notice that this will not be the "Main" application, and therefore, although F9 will compile it, Shift-F6 is needed to execute it, etc..

Perhaps it makes sense to use the Main project instead, but the method I've chosen, at least at the beginning, has been OK except for refactoring, which hasn't worked as I'd expected.

1"Plagiarize,
Let no one else's work evade your eyes,
Remember why the good Lord made your eyes,
So don't shade your eyes,
But plagiarize, plagiarize, plagiarize...
Only be sure always to call it please, "*research*"."

Thank you Tom Lehrer

Before creating a new Application set up your libraries!

Before starting creating applications, right-click on the project, and find the bottom entry, “properties”. Click here, and click on libraries, and then find the jfreechart libraries required and insert them into the classpath:

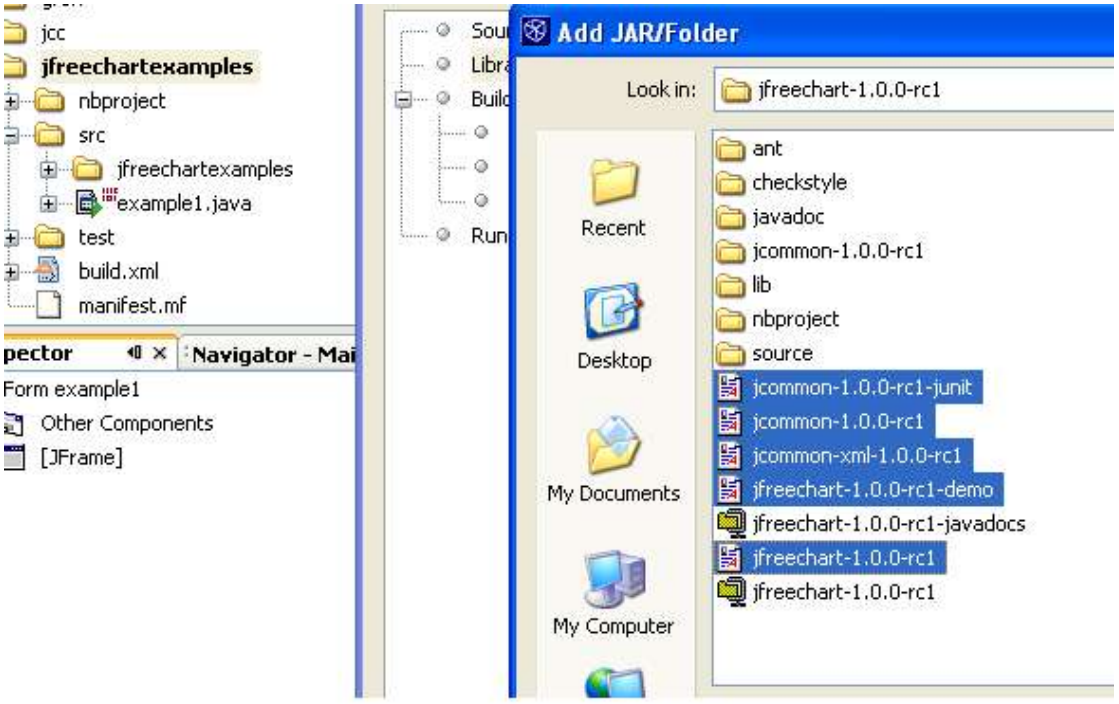

Illustration 1Figure 1

(notice “Libraries” is highlighted, and then following the next menu’s choices, locate the jar files necessary, wherever you’ve placed them).

Creating a new Application inside an existing Project with Graphical Interface

Now, when you right-click on your project and choose New, you can get a choice included in Figure 2 (see below).

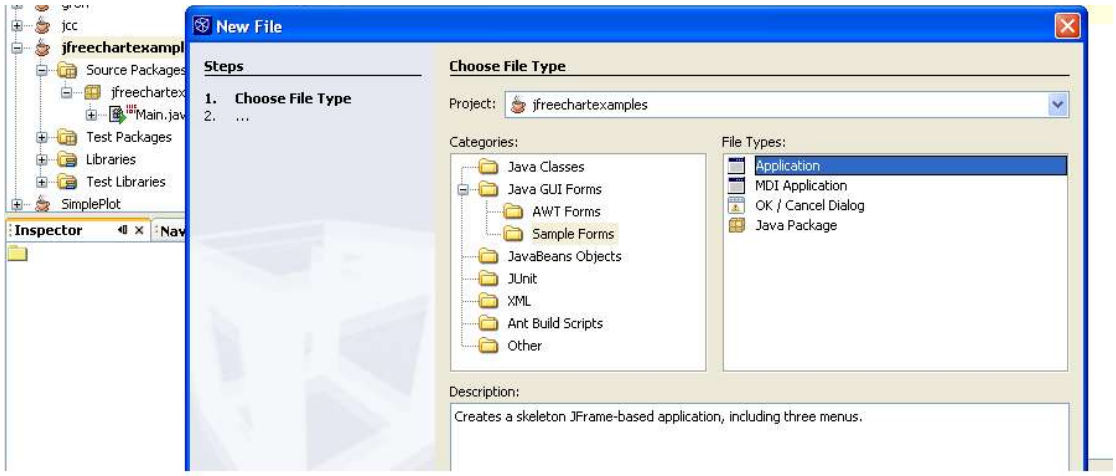

Illustration 2Figure 2

Just follow the menus from here.

Just remember that there will be no Main, and therefore shift-F6 executes the current program, etc.. Just check the right-clicks on the program's name in the Project window to see your options.

Example 1

Example 1 will consist of a cosine plot, in the easiest manner possible (I think).

Here is my “example1” code:

```
/*
 * example1.java
 *
 * Created on September 22, 2005, 11:47 AM
 */

/**
 *
 * @author  david
 */

import org.jfree.*;
import org.jfree.data.xy.*;//needed for XYSeries, could have called directly
import org.jfree.chart.*;//needed for createXYLineChart
import java.math.*;//needed for cosine
import java.awt.image.*;//needed for Buffered Image
import javax.swing.*;

public class example1 extends javax.swing.JFrame {

    /** Creates new form example1 */
    public example1() {
        initComponents();
        System.out.println("After initComponents");
        y_of_x = new double[n_points];
        x = new double[n_points];
        XYSeries series = new XYSeries("Cos(x) versus x");
```

```
    for (int i = 0;i< n_points;i++){//calculate the data to be plotted
        y_of_x[i] = Math.cos(i*Math.PI/180);
        series.add((double)i,y_of_x[i]) ;//add the computed values to the series
    }
    XYDataset dataset = new XYSeriesCollection(series);
    JFreeChart chart = ChartFactory.createXYLineChart(
            "Cos(x) versus x",
            "x",
            "cos(x)",
            dataset,
            org.jfree.chart.plot.PlotOrientation.VERTICAL,
            true,
            false,
            false);
    BufferedImage image = chart.createBufferedImage(400,500);
    jLabel1.setIcon(new ImageIcon(image));
    this.setSize(500, 600);
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
// </editor-fold>

private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new example1().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JMenuItem aboutMenuItem;
private javax.swing.JMenuItem contentsMenuItem;
private javax.swing.JMenuItem copyMenuItem;
private javax.swing.JMenuItem cutMenuItem;
private javax.swing.JMenuItem deleteMenuItem;
private javax.swing.JMenu editMenu;
private javax.swing.JMenuItem exitMenuItem;
private javax.swing.JMenu fileMenu;
private javax.swing.JMenu helpMenu;
private javax.swing.JLabel jLabel1;
private javax.swing.JMenuBar menuBar;
private javax.swing.JMenuItem openMenuItem;
private javax.swing.JMenuItem pasteMenuItem;
private javax.swing.JMenuItem saveAsMenuItem;
```

```
    private javax.swing.JMenuItem saveMenuItem;
    // End of variables declaration
    public static int n_points = 500;
    public static double[] y_of_x,x;//we will plot y(x) versus x
}
```

Notice that I've highlighted code which was created by NetBeans. Also, some NetBeans generated code is hidden (folded), to allow faster traversals of one's code. Here is the folded code which has been generated using the Design button to graphically create the code for a panel inside the face of the application:

```
    private void initComponents() {
        jLabel1 = new javax.swing.JLabel();
        menuBar = new javax.swing.JMenuBar();
        fileMenu = new javax.swing.JMenu();
        openMenuItem = new javax.swing.JMenuItem();
        saveMenuItem = new javax.swing.JMenuItem();
        saveAsMenuItem = new javax.swing.JMenuItem();
        exitMenuItem = new javax.swing.JMenuItem();
        editMenu = new javax.swing.JMenu();
        cutMenuItem = new javax.swing.JMenuItem();
        copyMenuItem = new javax.swing.JMenuItem();
        pasteMenuItem = new javax.swing.JMenuItem();
        deleteMenuItem = new javax.swing.JMenuItem();
        helpMenu = new javax.swing.JMenu();
        contentsMenuItem = new javax.swing.JMenuItem();
        aboutMenuItem = new javax.swing.JMenuItem();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        jLabel1.setText("<=picture");
        getContentPane().add(jLabel1, java.awt.BorderLayout.CENTER);

        fileMenu.setText("File");
        openMenuItem.setText("Open");
        fileMenu.add(openMenuItem);

        saveMenuItem.setText("Save");
        fileMenu.add(saveMenuItem);

        saveAsMenuItem.setText("Save As ...");
        fileMenu.add(saveAsMenuItem);

        exitMenuItem.setText("Exit");
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });

        fileMenu.add(exitMenuItem);

        menuBar.add(fileMenu);

        editMenu.setText("Edit");
        cutMenuItem.setText("Cut");
        editMenu.add(cutMenuItem);

        copyMenuItem.setText("Copy");
```

```
        editMenu.add(copyMenuItem);

        pasteMenuItem.setText("Paste");
        editMenu.add(pasteMenuItem);

        deleteMenuItem.setText("Delete");
        editMenu.add(deleteMenuItem);

        menuBar.add(editMenu);

        helpMenu.setText("Help");
        contentsMenuItem.setText("Contents");
        helpMenu.add(contentsMenuItem);

        aboutMenuItem.setText("About");
        helpMenu.add(aboutMenuItem);

        menuBar.add(helpMenu);

        setJMenuBar(menuBar);

        pack();
    }
```

In the future this folded code will be omitted. It is generated by the system, either during initialization or during construction of the graphical interface, and can be folded out of view for easier comprehension of the program's text.

To create *our* code, one starts with a definition (at the bottom, following the definitions induced by the NetBeans IDE):

```
    public static int n_points = 500;
    public static double[] y_of_x,x;//we will plot y(x) versus x
```

thinking that we will store the plot's abscissa in a vector. It turns out to be unnecessary, but so what[2]. Next, we create an instance of these variables, and attempt to create a jfreechart series instance. This fails, since we do not have jfreechart available for use yet,

```
        y_of_x = new double[n_points];
        x = new double[n_points];
        XYSeries series = new XYSeries("Cos(x) versus x");
```

so we need

```
import org.jfree.data.xy.*;//needed for XYSeries, could have called directly
```

or we could have written "org.jfree.data.xy.XYSeries" explicitly. This is easier, but finding oneself inside the various folders takes some getting used to.

Next, we want to generate the data.

```
for (int i = 0;i< n_points;i++){//calculate the data to be plotted
            y_of_x[i] = Math.cos(i*Math.PI/180);
            series.add((double)i,y_of_x[i]) ;//add the computed values to the series
}
```

Notice the construction of the "for (){}" group. Also notice that we need to add the statement

```
import java.math.*;//needed for cosine
```

2 In future incarnations, we'll consolidate the code omiting the y_of_x[] vector, but it doesn;t hurt to leave it in here.

in order to achieve the use of the cosine (and other) functions[3].

Next, we follow an algorithm to create out plot:

```
XYDataset dataset  = new XYSeriesCollection(series);
JFreeChart chart = ChartFactory.createXYLineChart(
    "Cos(x) versus x",
    "x",
    "cos(x)",
    dataset ,
    org.jfree.chart.plot.PlotOrientation.VERTICAL,
    true,
    false,
    false);
```

Finally, we display the chart in the simplest possible manner:

```
BufferedImage image = chart.createBufferedImage(400,500);
jLabel1.setIcon(new ImageIcon(image));
this.setSize(500, 600);
```

Where did jLabel1 come from? It came from the “Design” tab's functioning. We picked out jLabel and then touched the center of our displayed layout, and, lo and behold, there we are, its all done. If one looks at the “Properties” of jLabel1, one can find its “Text” and edit it to what one “<=picture”, which makes our output just a little more professional looking than the default value.

Saving one's work

For protection, we now click on the source code in the Project window, copy it, and paste it back into the source folder. It is created as example1_1 which we re-factor and re-name as example1_original (or something like that). Then we can fiddle with example1 and later re-factor/re-name its future conent to example2, using the same skeleton code. Actually, this hasn't worked as I'd expected, and perhaps you can suggest an alternative scheme.

Two Graphs on the Same Page

For scientific work, one often needs present two (or more) graphs sharing the common ordinate but different abscissas. “example2” is my choice of how to do this.

```
/*
 * example2.java
 *
 * Created on September 22, 2005, 11:47 AM
 */

/**
 *
 * @author  david
 */

import org.jfree.*;
import org.jfree.data.xy.*;//needed for XYSeries, could have called directly
import org.jfree.chart.*;//needed for createXYLineChart
```

3 Or write “java.math.Cos”.

```
import org.jfree.data.xy.XYSeriesCollection;
import org.jfree.chart.plot.*;
import org.jfree.chart.axis.*;
import org.jfree.chart.renderer.xy.*;
import java.math.*;//needed for cosine
import java.awt.image.*;//needed for Buffered Image
import javax.swing.*;
import java.awt.Font;

public class example2 extends javax.swing.JFrame {

    /**
     * Creates new form example2
     */
    public example2() {
        initComponents();
        System.out.println("After initComponents");
        y_of_x = new double[n_points];
        x = new double[n_points];
        XYSeries series1 = new XYSeries("Cos(x) versus x");
        XYSeries series2 = new XYSeries("Cos^2(x) versus x");
        for (int i = 0;i< n_points;i++){//calculate the data to be plotted
            y_of_x[i] = Math.cos(i*Math.PI/180);
            series1.add((double)i,y_of_x[i]) ;//add the computed values to the series
            series2.add((double)i,Math.pow(y_of_x[i],2)) ;
        }
        XYDataset dataset1 = new XYSeriesCollection(series1);
        XYDataset dataset2 = new XYSeriesCollection(series2);
        CombinedDomainXYPlot parent = new CombinedDomainXYPlot(new NumberAxis("x-angle
argument"));
        XYItemRenderer renderer1 = new StandardXYItemRenderer();
        XYItemRenderer renderer2 = new StandardXYItemRenderer();
        XYPlot subplot1 = new XYPlot(dataset1,null, new NumberAxis("Cos(x)"),renderer1);
        NumberAxis axis1 = (NumberAxis)subplot1.getRangeAxis();
        axis1.setTickLabelFont(new Font("Monospaced", Font.PLAIN,7));
        axis1.setLabelFont(new Font("SansSerif", Font.PLAIN,7));
        axis1.setAutoRangeIncludesZero(false);
        parent.add(subplot1,1);
        XYPlot subplot2 = new XYPlot(dataset2,null, new NumberAxis(null),renderer2);
        NumberAxis axis2 = (NumberAxis)subplot2.getRangeAxis();
        axis2.setTickLabelFont(new Font("Monospaced", Font.PLAIN,11));
        axis2.setLabelFont(new Font("SansSerif", Font.PLAIN,9));
        axis2.setAutoRangeIncludesZero(false);
        parent.add(subplot2,1);

        JFreeChart chart = new JFreeChart("Cos(x) versus x", parent);


        BufferedImage image = chart.createBufferedImage(400,500);
        jLabel1.setIcon(new ImageIcon(image));
        this.setSize(500, 600);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
```

```
    */
   //folded  code omitted
   private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
      System.exit(0);
   }

   /**
    * @param args the command line arguments
    */
   public static void main(String args[]) {
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new example2().setVisible(true);
         }
      });
   }

   // Variables declaration - omitted

   // End of variables declaration
   public static int n_points = 500;
   public static double[] y_of_x,x;//we will plot y(x) versus x
}
```

To see what the code is doing, we first look at the "for(){}" loop:

```
for (int i = 0;i< n_points;i++){//calculate the data to be plotted
         y_of_x[i] = Math.cos(i*Math.PI/180);
         series1.add((double)i,y_of_x[i]) ;//add the computed values to the series
         series2.add((double)i,Math.pow(y_of_x[i],2)) ;
      }
```

The cos(x) and $\cos^2(x)$ are part of the java.Math.* classes which can be imported or explictly coded.

Next, we define two rather than one XYDataset:

```
      XYDataset dataset1 = new XYSeriesCollection(series1);
      XYDataset dataset2 = new XYSeriesCollection(series2);
```

Each dataset is addresses a different function. They are going to be combined together using a new kind of plot:

```
CombinedDomainXYPlot parent = new CombinedDomainXYPlot(new NumberAxis("x-angle argument"));
```

Then we define a renderer for each of the datasets, e.g.,

```
XYItemRenderer renderer1 = new StandardXYItemRenderer();
```

Penultimately, we set up each series to be added:

```
 XYPlot subplot1 = new XYPlot(dataset1,null, new NumberAxis("Cos(x)"),renderer1);
      NumberAxis axis1 = (NumberAxis)subplot1.getRangeAxis();
      axis1.setTickLabelFont(new Font("Monospaced", Font.PLAIN,7));
      axis1.setLabelFont(new Font("SansSerif", Font.PLAIN,7));
      axis1.setAutoRangeIncludesZero(false);
      parent.add(subplot1,1);
```

First, we define subplot1, defining the axis label ("Cos(x)"), and associate the proper renderer which will give rise to a unique color for this particular suplot (among other things). We define this plot's axis and set up some axis properties, ending by adding the

subplot to the CombinedDomainXYPlot (parent). The weight (1) could have been omitted, or changed to have this plot bigger (or smaller) that its brother.

Of course, we do the same with the second series of data points, $\cos^2(x)$, but this time we change some properties so that we can tell what we are doing.

Finally, we repeat the code from example1 to display the code.

```
JFreeChart chart = new JFreeChart("Cos(x) versus x", parent);


    BufferedImage image = chart.createBufferedImage(400,500);
    jLabel1.setIcon(new ImageIcon(image));
    this.setSize(500, 600);
```

In the next example, we'll use a different way of displaying our graph.

Adding javadocs

In the NetBeans IDE, one of the great conveniences is the idea that when one types "org."[4] a popup window shows up with choices for the next element, such that clicking on "jfree" and then adding a period yields "org.jfree." with yet another popup box, etc.. These boxes are accompanied by explanatory javadocs boxes if the appropriate foreign (to NetBeans and Java) javadocs have been placed in the appropriate libraries. After some trial and error, the following worked:

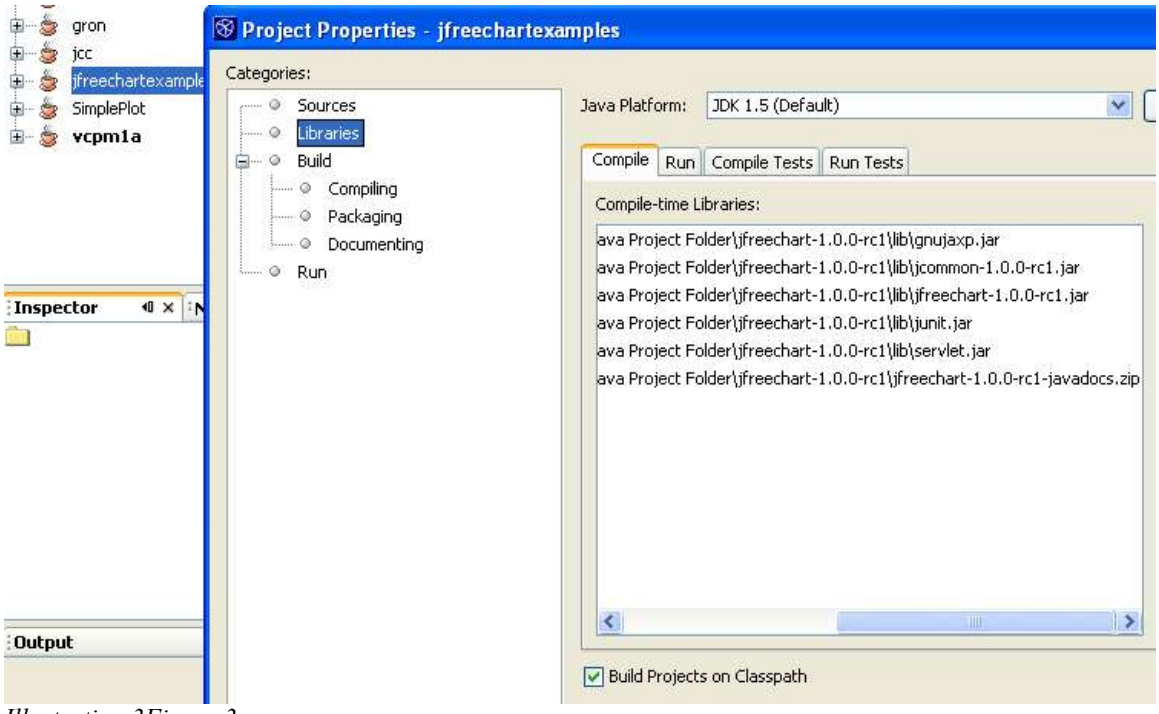

Illustration 3Figure 3

where a right-click on the appropriate project followed by "Properties" followed by "Libraries" allowed us to add the docs zip file, and suddenly everything worked. Since I

4 With the period at the end!

added this same zip file to the libraries under Tools, I'm not sure what's going on, and the NetBeans documentation is not too clear, but a little trial and error worked for me and will, presumably, work for you.

Example2a

We here change the past example to a briefer form:

```
XYDataset dataset1 = new XYSeriesCollection(series1);
XYDataset dataset2 = new XYSeriesCollection(series2);
CombinedDomainXYPlot parent = new CombinedDomainXYPlot(new DateAxis(""));
XYItemRenderer renderer1 = new StandardXYItemRenderer();
```

Notice that we've chosen a different scheme for displaying the graph, i.e., using a ChartPanel:

```
XYItemRenderer renderer2 = new StandardXYItemRenderer();
XYPlot subplot1 = new XYPlot(dataset1,null, new NumberAxis(null),renderer1);
XYPlot subplot2 = new XYPlot(dataset2,null, new NumberAxis(null),renderer2);
parent.add(subplot1);
parent.add(subplot2);
JFreeChart chart = new JFreeChart("Cos(x) and Cos^2(x) versus x", parent);

/* commented out
BufferedImage image = chart.createBufferedImage(400,500);
jLabel1.setIcon(new ImageIcon(image));
this.setSize(500, 600);
 **/
ChartPanel myChart = new ChartPanel(chart);
this.setSize(500, 600);
this.setContentPane(myChart);
```

And we've finished with our introduction. Good luck.