C. W. David
Department of Chemistry
University of Connecticut
Storrs, Connecticut 06269-3060
(860)486-3217
Carl.David@uconn.edu

This is an attempt to meld JCCKit and NetBeans/Java in the same way as I started with JfreeChart; The idea is helping novices in all three to get up and going. The motivating influence is to get scientific programming rapidly implemented in this new Java/NetBeans environment, using a graphics environment appropriate to generating graphs on the fly (a common scientific requirement).

I should explain that I'm a typical uneducated hacker when it comes to programming, having learned FORTRAN in (about) 1963, progressed on to BASIC, and then through a plethora of languages, each time hacking around in order to make the progress that my (scientific) project required at the time. Java, NetBeans and JCCKit are all free, downloadable pieces of software which allows a novice such as me to proceed in a relatively inexpensive manner; there are alternative free IDE's available, but one has to choose one, and I choose NetBeans. As to JCCKit, it is not well (imho) documented, and if I continue this project (while pursuing a scientific goal concerning water molecular mechanics models) I hope to contact its author and document the scheme for novices such as myself.

So read on, and I hope that I'm not making any mistakes. If you discover one (or more) please e-mail me at Carl.David@uconn.edu and I will correct the material. Please!

```
/*
 * app2.java
 *
 * Created on October 12, 2005, 12:37 PM
 */

package jcctest;

/**
 *
```

```
 * @author  c.w.david
 */
import jcckit.GraphicsPlotCanvas;
import jcckit.data.DataPlot;
import jcckit.data.DataCurve;
import jcckit.util.*;
import jcckit.data.*;
import java.util.Properties;
import javax.swing.JPanel;
import java.awt.*;

public class app2 extends javax.swing.JFrame {

    /** Creates new form app2 */
    public app2() {
        initComponents();
        double inc = 2.5d;
        curve = new DataCurve("cosine");
        curve2 = new DataCurve("cosine squared");
        for (int i = 0;i<100;i++){
            double cos = Math.cos((double)i*inc*Math.PI/180);
            curve.addElement(new DataPoint(i,cos));
            curve2.addElement(new DataPoint(i,Math.pow(cos,2)));
        }
        myPlot = new DataPlot();
        myPlot.addElement(curve);
        myPlot.addElement(curve2);
        myCanvas =  createPlotCanvas();//"true".equals(getParameter("graphics2D")));
        myCanvas.connect(myPlot);
        JPanel myPanel = new JPanel();
        setLayout(new BorderLayout());
        add(myCanvas.getGraphicsCanvas(),BorderLayout.CENTER);
        add(myPanel,BorderLayout.SOUTH );
        pack();
        this.setSize(500, 500);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new app2().setVisible(true);
            }
        });
    }
```

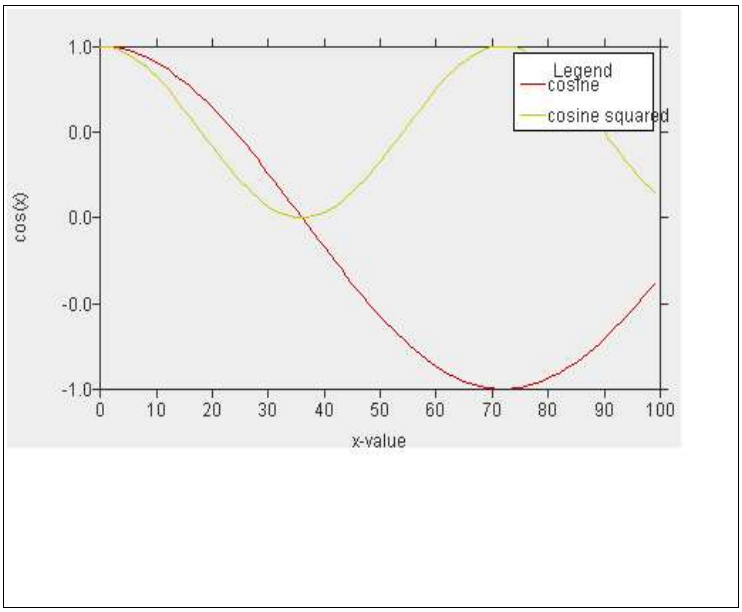
```
   public static GraphicsPlotCanvas createPlotCanvas() {
      Properties props = new Properties();
      ConfigParameters config = new ConfigParameters(new PropertiesBasedConfigData
(props));
      props.put("plot/legendVisible", "false");
      props.put("plot/coordinateSystem/xAxis/minimum", "0.0");
      props.put("plot/coordinateSystem/xAxis/maximum", "100");
      props.put("plot/coordinateSystem/xAxis/axisLabel", "x-value");
      props.put("plot/coordinateSystem/xAxis/ticLabelFormat", "%d");
      props.put("plot/coordinateSystem/yAxis/axisLabel", "cos(x)");
      props.put("plot/coordinateSystem/yAxis/maximum", "1");
      props.put("plot/coordinateSystem/yAxis/minimum", "-1");
      props.put("plot/coordinateSystem/yAxis/ticLabelFormat", "%f.0");
      props.put("plot/curveFactory/definitions", "curve");
      props.put("plot/curveFactory/curve/symbolFactory/attributes/className",
            "jcckit.graphic.ShapeAttributes");
      props.put("plot/curveFactory/curve/symbolFactory/attributes/fillColor",
"0xfe8000");
      props.put("plot/curveFactory/curve/symbolFactory/attributes/lineColor", "0");
      props.put("plot/curveFactory/curve/symbolFactory/size", "0.08");


      return new GraphicsPlotCanvas(config);
   }
   // Variables declaration - do not modify
   // End of variables declaration
   public static DataPlot myPlot;
   public static GraphicsPlotCanvas myCanvas;
   public static  DataCurve curve, curve2;
}
```

The code is presented as an attempt to make one understand the absolute minimum possible, and since the documentation in this pacckage (jcckit) is virtually non-existent, the code is also presented as a hack, pure and simple. Since this is not a “main” method from the point of view of NetBeans, you need F9 to compile, Shift F6 to execute, and Shift-Cntrl-F5 to debug. You can see the incentive to have a “main” class.

props.put("plot/legendVisible", "true"); places the legend in the graph with the names of the functions:

curve = new DataCurve("cosine");
curve2 = new DataCurve("cosine squared");
as indicated inside the quotations marks.

Here is the output from this code (with the minor change:props.put ("plot/legendVisible", "true"); so that we get an appropriate legend):



which certainly looks OK.

This seems enough to get a novice user going, but there will be more as time permits.