Geno Copertino

Graphing using JFreeCharts and Apache

First, I had to import a bunch of things from the JFreeCharts and Apache libraries. This was necessary to be able to create the plots, and use some of the other features needed to accomplish this piece. I create a class Plot that extends JFrame and then initialize the plot. I name the GUI and set a static variable equaled to 100 as it will be used many times throughout the program to create the array sizes. Here is also where you initialize the size of the plot and ensure that the frame exits when the X button is pressed. Then the panel was created where you label the axises and title.

```
import javax.swing.JFrame;
import javax.swing.JPanel;

import java.awt.BorderLayout;
import java.awt.Color;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;

import org.apache.commons.math3.random.JDKRandomGenerator;
import org.apache.commons.math3.stat.descriptive.moment.Mean;

public class Plot extends JFrame {
    int value = 100; //sets the size of the arrays

    public Plot(int userValue) { //initialize the plot
        super("Graph, Salt, and Smooth using JFreeCharts and Apache"); //names the GUI
        this.value = userValue;
        JPanel panel = createPanel(); //creates the panel
        add(panel, BorderLayout.CENTER);

        setSize(600, 600); //initializes size
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //exits the frame when X button is pressed
        setLocationRelativeTo(null);
    }

    private JPanel createPanel() {
        String title = "Ploting, Salting, & Smoothing a Function"; //sets the title
        String xLabel = "x-axis"; //labels the x-axis
        String yLabel = "y-axis"; //labels the y-axis

        XYDataset dataset = generateDataset(); //calls the generateDataset() method
        JFreeChart chart = ChartFactory.createXYLineChart(title, xLabel, yLabel, dataset); //creates the chart and with labels
        XYPlot myPlot = chart.getXYPlot();
        XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) myPlot.getRenderer();
        renderer.setBaseShapesVisible(true);
        chart.getXYPlot().setBackgroundPaint(Color.WHITE); //sets background color
        myPlot.getRenderer().setSeriesPaint(2, new Color(100, 238, 244)); //sets the color of functions
        return new ChartPanel(chart); //returns chart
    }

```

Then, came the generateDataset() method which contains the bulk of the code. Here we created the x and y arrays and looped through both to ensure they contain values. To generate x's it was just a simple for loop to keep incrementing them by 1 with the bound of 100 which we declared earlier. For y it was similar, but we had to include the function piece. Since, I chose to do y = 5x + 10 I had to make sure each x was getting multiplied by 5 and added by 10 and then added to graph as a whole. Salting the data consisted of a for loop with 3 nested if loops. They basically worked as three different cases and they all invoked sequentially. For example, when rand = 1, a random y values from (0 to 30) + 15 was input into the y array.

```java
private XYDataset generateDataset() {
    JDKRandomGenerator random = new JDKRandomGenerator();
    final XYSeries newGraph = new XYSeries("Original");
    //create two arrays for x and y.
    Double[] x = new Double[value];
    Double[] y = new Double[value];

    double generateX = 0; //create a for loop that adds x values
    for(int i = 0; i < value; i++) {
        x[i] = generateX;
        generateX += 1;
    }
    //adds my function (y = 5x + 10) to the generated dataset
    double generateY = 0;
    for(int i = 0; i < value; i++) {
        generateY = (5 * (x[i]) + (10));
        y[i] = generateY;
        newGraph.add(x[i], y[i]);
    }
    //salts data
    final XYSeries saltData = new XYSeries("Salt");
    double newValue = 0;
    //create a for loop with a random integer between 0-2 so that the y values are randomized
    for(int i = 0; i < value; i++) {
        int rand = random.nextInt(2);
        if(rand == 0) {
            newValue = y[i] + (random.nextInt(30) + 15);
            y[i] = newValue;
            rand = 1;
        }

        if(rand == 1) {
            newValue = y[i] + (random.nextInt(10) + 5 );
            y[i] = newValue;
            rand = 2;
        }

        if(rand == 2) {
            newValue = y[i] + (random.nextInt(25) - 10);
            y[i] = newValue;
            rand = 0 ;
        }


        //adds the new x and y values to the graph using saltData object
        saltData.add(x[i], y[i]);
    }
```

To smooth the data, we try to calculate the mean of each salted y value and make it our new smoothed value using a for loop. Finally, we declare the dataset, run the calculations, and run the program in our Test class.

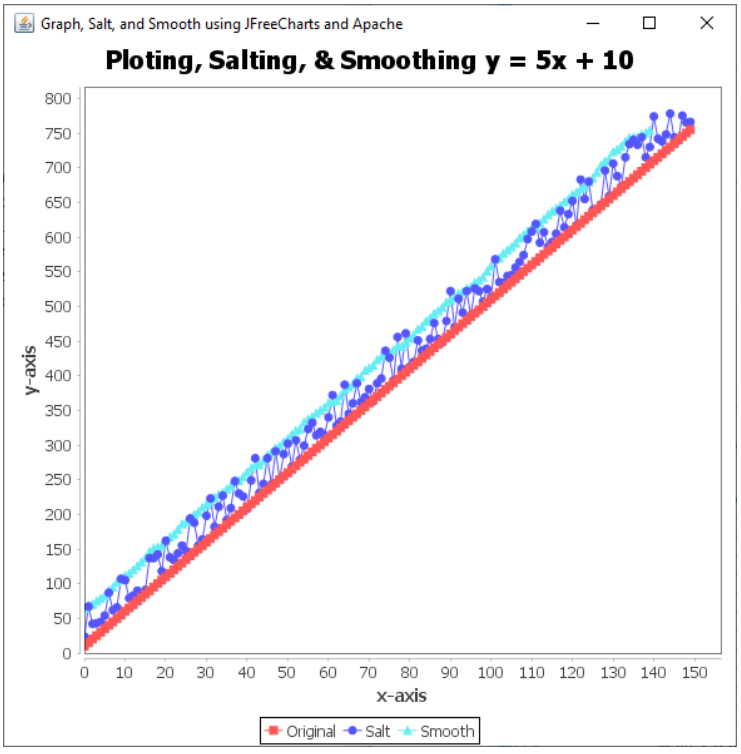
```java
//smooths data
final XYSeries smoothData = new XYSeries("Smooth");

Mean mean = new Mean();
Double[] m = new Double[value - 10];

for(int i = 0; i < value - 10; i++) {
    mean.increment(y[i]);
    mean.increment(y[i + 1]);
    mean.increment(y[i + 2]);
    mean.increment(y[i + 3]);
    mean.increment(y[i + 4]);
    mean.increment(y[i + 5]);
    mean.increment(y[i + 6]);
    mean.increment(y[i + 7]);
    mean.increment(y[i + 8]);
    mean.increment(y[i + 9]);
    mean.increment(y[i + 10]);
    m[i] = mean.getResult();
    mean.clear();
    smoothData.add(x[i], m[i]);
}

final XYSeriesCollection dataset = new XYSeriesCollection();
dataset.addSeries(newGraph);
dataset.addSeries(saltData);
dataset.addSeries(smoothData);

return dataset;
```



Then, we run the Test class and get the three different line graphs.

As you can you see, the red line is our original function y = 5x + 10, which is just a straight line. The salted data is similar, but you can see obvious kinks up and down the y axis. Then, with our smoothed data line it irons out those kinks almost completely.