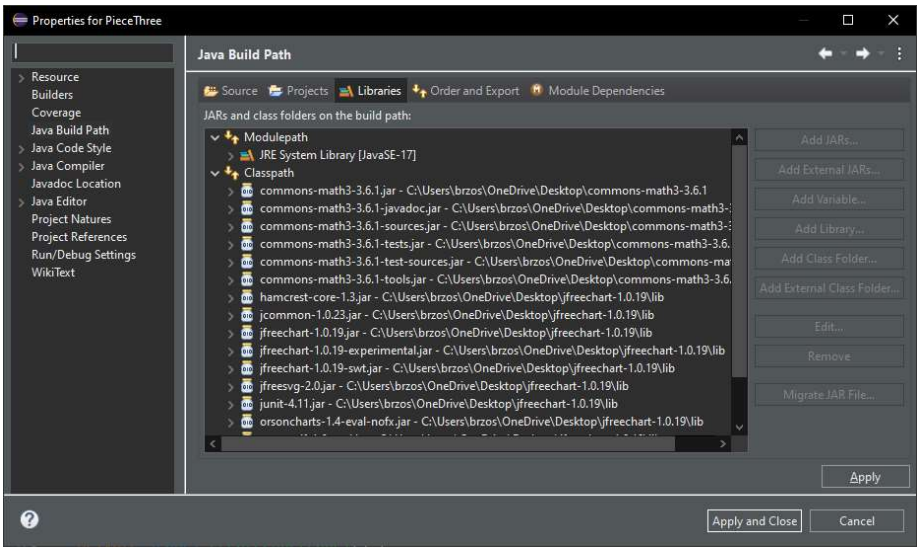
Piece 2

Plot, Salt, Smooth Data

With JFreeCharts and Apache Stats Libraries

At first, this piece of the project was giving me a lot of trouble because I was not importing the two libraries to the classpath. Instead, I was having the .jar files imported into the modeulepath. This was the first time that I was importing a library therefore, it took me a few minutes to determine that maybe it should be in the classpath instead of the module path. After moving JFreeCharts and Apache stats library to the classpath, I was able to import anything from those libraries so that I could complete the assignment. However, first I needed to understand how both the libraries work so that I could program anything with them.



Prior to showing the graph of my results, I wanted to explain how I was able to program this piece of the project and get it to plot the original, the salted, and the smooth graphs of the function y = 9x + ½. I chose to do the same function in piece 3 where I used octave to graph the original and then also graph the salted version as well as the smoothed graph. First, I extended Plot to JFrame and I would then create the JFreeChart. The documentation on JFreeCharts was very helpful since I was able to easily create the JFreeChart that I wanted. I was able to set the size to 800 x 800 and allowed the button on the top right (top left for MAC users) to close the application.

```
public Plot(int userValue) { //initialize the JFreeChart window
    super("Piece 2"); //names the GUI to "Piece 2"
    this.value = userValue;
    JPanel panel = createPanel(); //creates the panel
    add(panel, BorderLayout.CENTER);

    setSize(800, 800); //initializes the size to 800 x 800
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //allow the button to close the graph
    setLocationRelativeTo(null);
}
```

Next, I would make a method that would create the panel for the chart. Using JPanel, I was able to create the panel for the graph. This method allowed me to set the title of the chart, set the x-axis title, and the y-axis title. I was able to create a new XYDataset dataset equal to the generateDataset() method which would fill in the data regarding the original graph, the salted, and smoothed graphs. After initializing the plot, I was able to set a few other

```
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 includes the title, xLabel, yLabel, and the dataset.
    XYPlot myPlot = chart.getXYPlot();
    XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) myPlot.getRenderer();
    renderer.setBaseShapesVisible(true);
    chart.getXYPlot().setBackgroundPaint(Color.WHITE); //sets the background color
    myPlot.getRenderer().setSeriesPaint(2, new Color(50, 250, 150)); //sets the color of functions
    return new ChartPanel(chart); //returns the chart
}
```

things such as the color of the background in the graph which I set to white. I also was able to change the individual colors of the plotted functions as well.

The generateDataset() method is another method and the first few lines are relatively self-explanatory. You are able to see that a random number generator is initialized as well as a graph variable. Then, I was able to create the Arrays for the x and y variables. This would allow me to have the coordinates for the point that I would be graphing. I would create a for loop so that I can populate the x[] and y[] arrays with the proper values. After I added values to both x[] and y[], I was able to add x[i] and y[i] to the graph.

To salt the data, I created a for loop that would have three if statements. In these if statements, if the random value is 0, 1, or 2 then, it would add a random value to the y value. This would be different values for the three if statements. To smooth the data, I created a for loop that would add 10 increments from the mean. Finally, I added the original graph, the saltData graph, and the smoothData graph to the dataset. Within my tester class, I was able to simply call the plot from the SwingUtilities library in javax.swing. When doing this final step, you can set the size of the x-axis. I personally set my graph to 150 x values because I wanted to have the functions shown more visibly. Below is my screenshot with the results of this piece of the project.