Java Apache & JFreeCharts Documentation

Welcome to the Java Libraries documentation! This documentation will show you all the methods, explanations of the methods I wrote and how I formatted my program to do what it needs to do. Below details the classes, the methods and everything inserted from the libraries and Java stuff too.

If you haven’t, please visit <https://byronproject2.carrd.co/>.

*The first class is PlotSaltSmoothData.java.*

A computer screen shot of a program

Description automatically generated

As you see above, the package is inserted (yeah, this was a test, and I never started over…so its name idk.) Then Apache, jfreeCharts and javax.swing and Random are imported into the program. A ton of private series are created. There is a separate one for plotting, salting and smoothing data. DescriptiveStatistics was imported here, but I never found a use for it.

**Method – plotQuadraticFormula()**

A computer screen shot of a program

Description automatically generated

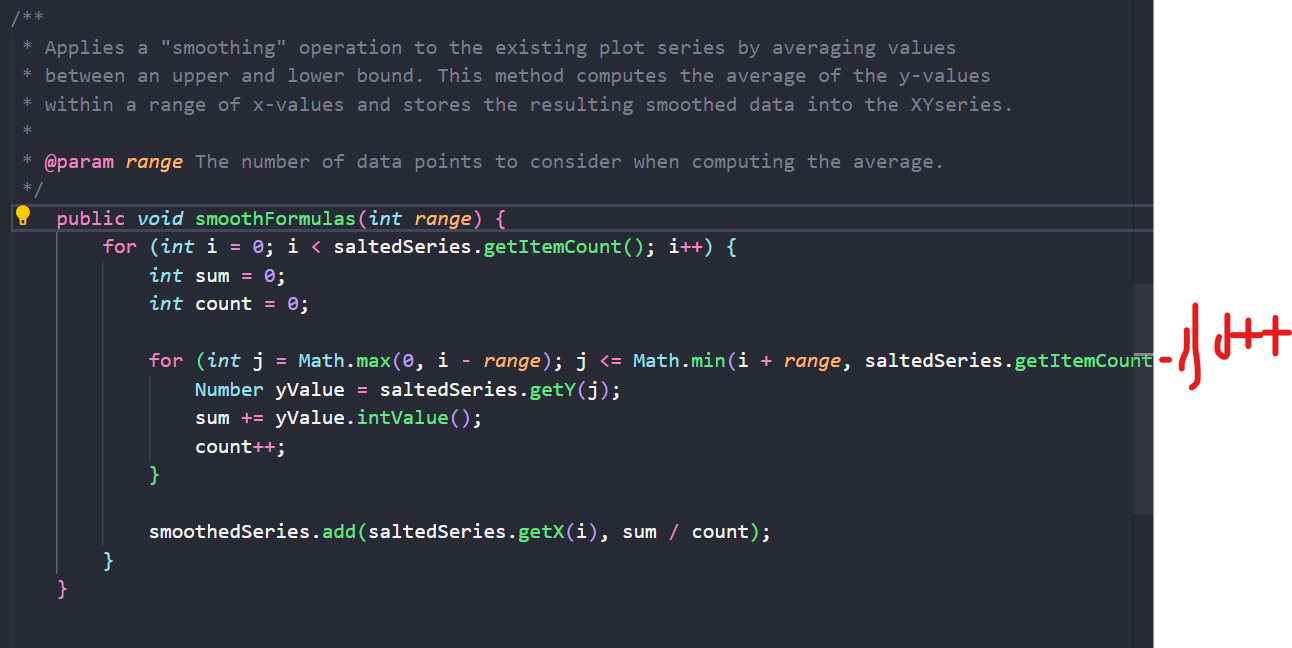
This method plots x values from -50 to 50. It is very similar to my method in the regular Java program. However, the difference is instead of adding values to an Array List, it uses the JfreeCharts XYSeries. It adds the x values and the new y values based off the quadratic.

**Method - saltFormulas(int bound)**

A computer screen shot of a program code

Description automatically generated

This method first clears the saltedSeries (just in case). It runs a loop through all the plot series values to prevent a wrong iteration in case of the range being changed. Then, it puts the x values from the plot series into a Number variable, and it puts the y values into a Number variable too. GetX and getY seemed new here, but they work like the normal array list command “.get”. The ogY variable held the original y values. The salt variable held the salted Y values based off a random integer from a parameter bound subtracted by 100. The X values along with the salted y values were added to the salted series.



I could not get the entire method in, so I drew the end of the for-loop. This method accepts a range. It averages values around each data point within that range, looking at the low and high bounds. Math.max and Math.min were used here to take the smallest and largest numbers. The y values from the salted series were stored in the Y value number variable. They were added to the sum. After every computation in the for loop, counted was added to. The x values and smoothed y values (sum / count) were added to the smoothed XY series.

**Methods - getPlotSeries(), getSaltedSeries(), and getSmoothedSeries().**

A computer screen shot of text

Description automatically generated

These three methods were used to return the plot series, salt series and smooth series. The getter methods drag the values over to the GraphMaker.java class to use these values in the graph creation process.

*The second class is GraphMaker.java.*

It starts with the jfree imports to make the graph or “chart” as well as the color import and the scanner import for the last part of the project.

*A computer screen shot of a program code

Description automatically generated*

A plotSaltSmoothData object is created to use the getters showed before to retrieve the data from the three XY series.

**Method – Graph()**

A screenshot of a computer program

Description automatically generated

A computer screen shot of code

Description automatically generated

This method’s purpose is to create a graph of the function. In this case, it is quadratic. It does not perform any salting or smoothing. This graph has a purple line. The color was created using RGB colors. That part was cool, like I made it purple. The function was named here, the axis was labelled, the plot orientation was set, the legend, tooltips and URLs. The second part was the outside of the chart. The size was set, the dimensions, the title above and the locations. It was just a ton of commands played with here.

**Method – Graph2()**

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

This method is like the one above except the graph is plotted then salted. This is the salted graph. The line is blue. The name is “Salted Quadratic Function Plot.” It was salted by 100 here. The bottom stuff is just a ton of stuff I was playing with once again. What is funny is I was not getting a salted graph at first, so I was playing with all those additions, yet it was because I didn’t call the function.

**Method - Graph3()**

A computer screen shot of a program

Description automatically generated

A computer screen shot of code

Description automatically generated

The third method to graph the third smoothed graph is the same. It is called the plotted, salted and smoothed XY series. This ends up producing the smoothed graph. It does all the same steps, creating the chart/graph. The only difference is it has a pink line. However, the process was fun to create!

**Method – run()**

A computer screen shot of a program code

Description automatically generated

This is the run method that runs the program. It prompts the user to press 1, 2 or 3 (or 4 to exit) to pick the graph of their choosing. 1 being the original plotted graph, 2 being the salted graph, and 3 being the smoothed graph. It uses a simple scanner to do this.

The last class is the “Tester.java” class which just runs the run() method.

A computer screen shot of a program code

Description automatically generated

By this point in my coding endeavors, I learned “Hey, don’t make your main method busy.” So, I ended up making a ton of classes and only using the main method class to run the engine! The end.