Using JFreeCharts and Datasets to make a Chart of a Plotter, Salter, and Smoother in Java

Melvin Vazquez Andujar

April 26, 2024

**Abstract:**

This project is very similar to the Java project in that the logic and most of the math is done using the same methods and classes. However, the main difference for this program is that we use JFreeCharts to make a chart for the plotter, salter, and smoother. The chart is a graph that gets shown to the user when the program runs and, in the chart, it shows all three graphs. Plus, the way the data gets stored for this program is in a data series and not in an ArrayList like it was in the Java Program.

Table of Contents

[Results: 2](#_Toc165133229)

[JFreeChartProgram Class: 5](#_Toc165133230)

[Creating JFreeChartProgram with External Libraries: 5](#_Toc165133231)

[Methods: 5](#_Toc165133232)

[Conclusion: 6](#_Toc165133233)

[Bibliography: 7](#_Toc165133234)

# Results:

A graph showing a graph

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

# JFreeChartProgram Class:

## Creating JFreeChartProgram with External Libraries:

Creating the JFreeChart class for this program was difficult in that learning how to use XY series, and XY Series collection because it was weird since it is very similar to each other in how it works and was confusing me in which one to call at times. However, once I did the first method and it worked, it made sense to me as to how XY series is for one data set for whatever I specify it for, while XY series collection is like a database that stores the series I created in each method. This makes it easier to call the data and use it when I need to in other methods. Additionally, figuring out how to do the use charts and create frame that gets the chart to print out the code was difficult to do. However, I was able to find a website that explained and gave very clear directions on how to do the charts functions.

### Methods:

#### plotterAPI(double min, double max, double increments):

This method is very similar to my Java program in that it takes three parameters to set the x to a min and end at the maximum in a for loop. Inside that for loop I run the equation that I use until I get to the max x that it should stop at. I am using the same equation which is . The difference with this code is that instead of an ArrayList I used a XY series to store the values of x and y and once I was done getting all data points, I added the complete series into the collection and called the CSV method to print the data into a csv.

#### saltedData(double min, double max):

Same as the method above I used my code from my java program, the only difference is that I made another series for the salted data, and I called the plotter series inside the loop to do the portion where I have to mess up the data. The min and max is for the range I ask the user to input where it will add/subtract a number from min to the max given inclusive. Once the code was done similar to the plotterAPI I add the values into the salter series, add the completed series into collection and call the salter csv.

#### smoothedData(int windowVal, int smoothingTimes):

Similar to the last two methods I use the same code from java in which I use multiple for loops to try and smooth the data back to how the original looks from looking at values left and right depending on how many the user wants the code to look at. It will get the average and add that to the smoother data series. At first this program was a bit difficult for me because I was overthinking the program and trying to use a new series for average values and such, but then I realized I could store the result in a variable and that variable gets added to smoother series. Once the code finished I added the completed series to the collection and called the smoother csv.

#### plotterCSV():

This is just a copy and paste of my java csv code, the only thing I changed was the way to access the XY series in that it is different how you get the value of an index and different in how you check the size of the data series. Other than that, everything stayed the same and I use file to create a csv file and printwrite to write the data points into the csv.

#### salterCSV():

This is just a copy and paste of the method above, the only thing that changes is the data series being called and changed to the salter series

#### smootherCSV():

This is just a copy and paste of the plotterCSV method above, the only thing that changes is the data series being called and changed to the smoother series

#### boundsAndRange():

This method uses a Scanner to get user input on the parameters being used in the plotter, salter, and smoother methods. The user inputs the values for each parameter and then the method checkBounds is called to make sure the values make sense and will work in the code.

#### checkBounds(double start, double end, double increment, double min, double max):

In this method I use the parameters that we get from Bounds to check whether it makes logical sense, so it checks that the start is not greater than the end, that it is incrementing by more than 0, and that min is not greater than max. If all is good then this method calls the plotterAPI(start, end, increment), saltedData(min, max), and smoothedData(5,1) with the parameters that were inputted by the user.

#### XYLineChart():

This method I thought would be the hardest since I did not know how to do charts at all, but once I found the website that helped, this code became very easy. The only hard part was figuring out the ChartFrame and how I would use it in the code. This code just labels the axis and gives the chart a title. After that we use JFreeChart built in method to create a frame with the labels and title. We also used the XY series collection that stored all the series called dataset. This dataset uses all the series and values inside to graph it in the frame.

# Conclusion:

In conclusion this was a very frustrating program compared to all the other due to difficulties in getting a jar file for JFreeCharts. I tried to follow their installation guide and tried to use maven, but nothing worked, and I was stuck. I was able to ask a friend that had an older version of JFreeCharts to send me the jar file and I was able to finally start this program. Other than the technical issues, this was one of the more interesting programs to do because of the way external libraries work. I think that this taught me the importance of external libraries and how it makes coding much easier since built in methods do parts of the program for me. For example, I was looking at videos for Apache and found how quick I could do a math problem through the built-in methods they have for different levels of math. They have differentiation, derivatives, etc. I enjoyed learning more about external libraries and is something I will definitely be looking into more as I involve myself deeper in the coding field. Other than that, this program opened my eyes to different ways of coding, being more efficient in the way that I code, and more optimal ways of doing research for specific cases and figuring out how to use Eclipse on a deeper level through build paths, jar files, and csvs.

# Bibliography:

Agrawal, Vikas. “Random Numbers, Distributions and Statistics Using Apache Commons Math Library in Java Eclipse.” *YouTube*, YouTube, 13 Mar. 2015, www.youtube.com/watch?v=MifbwoCumX0.

baeldung. “Introduction to Apache Commons Math.” *Baeldung*, 8 Jan. 2024, www.baeldung.com/apache-commons-math.

Galgani, Francesco. “Introduction to Jfreechart.” *Baeldung*, 10 Mar. 2024, www.baeldung.com/jfreechart-visualize-data.

Minh, Nam Ha. “Using JFreechart to Draw XY Line Chart with XYDataset.” *Using Jfreechart to Draw XY Line Chart with XYDataset*, 2019, www.codejava.net/java-se/graphics/using-jfreechart-to-draw-xy-line-chart-with-xydataset.