Research Process

This was a particularly interesting part of the project because I never really referenced libraries that can be found online. The first issue I encountered was actually being able to find the correct jar files to download. When I was first looking for JfreeChart and Apache jar files, it seemed like there were a plethora of different files I could download. This resulted in a bit of confusion but after further googling, I was able to find the proper ones. I also followed this youtube [video](https://www.youtube.com/watch?v=aBONSQ44cnk) to correctly find the zip file necessary in downloading for this project. After this, I had to figure out how to import a jar file to my java project. To do this, I then followed this youtube [video](https://www.youtube.com/watch?v=ujh0ZrAvwTU) on how to import jar files in eclipse. My next step was to verify that the jar file I imported into my java project was actually working correctly. I then did some [googling](https://www.vogella.com/tutorials/JFreeChart/article.html) and found some imports that I could use for my class and was able to verify that it worked!

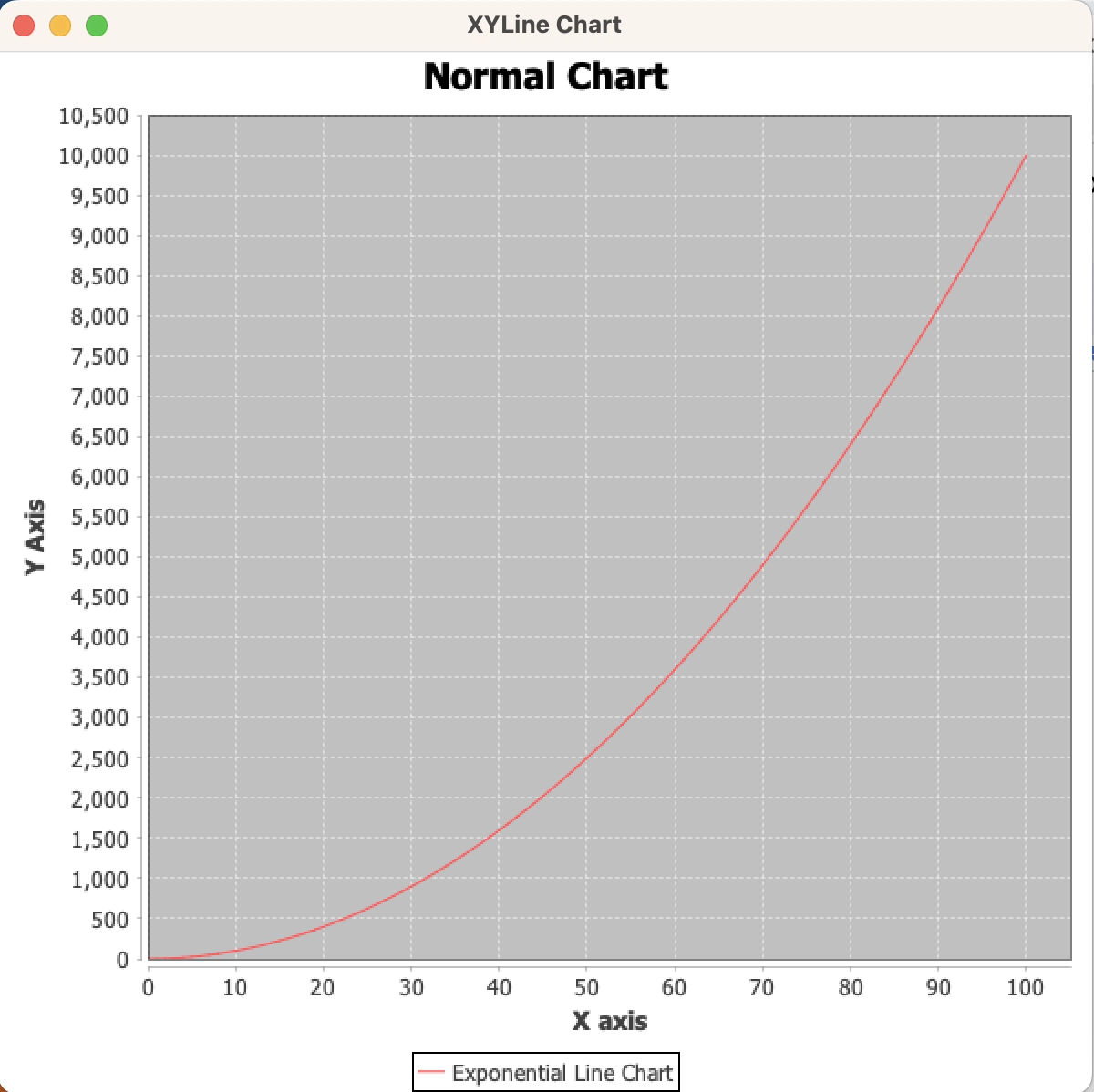
After successfully importing JfreeChart, I had to find the Apache jar file. After some [googling](https://commons.apache.org/proper/commons-math/download_math.cgi), I was able to identify the correct file and use it in my java project. To verify that the library worked with my java project, I looked up some [example imports](https://www.youtube.com/watch?v=MifbwoCumX0) and verified that it worked! Once this was done, the mission was actually figuring out how to set up a chart using JFreeChart and plotting x and y values by using Apache. After some [research](https://bethecoder.com/applications/tutorials/charts/jfreechart/xy-line-chart.html), I was able to make a skeleton for my graph, labels, and empty datasets. I then coded for a bit and started plugging things in to see what worked. I was then able to get a line going but then needed to connect it to the Apache library somehow. After looking through the documentation for the library, I stumbled upon the [Power](https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/analysis/function/Power.html) library and found exactly what I was looking for. After coding for a bit, I was finally able to make an exponential graph.

The next task was figuring out how to salt the data by using Apache. It was relatively similar to the original exponential graph, but now I had to add random values so the graph has a more volatile look to it. After some [googling](https://www.baeldung.com/java-generating-random-numbers#1-orgapachecommonsmath3randomrandomdatagenerator), I was able to find a class from the Apache library that allowed me to generate random values for the salter method. Once this was done, I added random values for every even index, and then I subtracted random values for every odd index. Ultimately creating a jagged line as it expanded exponentially.

The final task was figuring out how to smooth the data by using JFreeChart. I still used the same method for graphing the data as I did for the original plot and the salted plot. Although, smoothing the data definitely required some thorough googling and search. After some extensive research, I was finally able to find the [MovingAverage](https://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/time/MovingAverage.html) class in the JFreeChart api. After doing some troubleshooting, I was finally able to get the jagged line to smooth. This was definitely the hardest part of this section due to working with libraries that I was unfamiliar with.

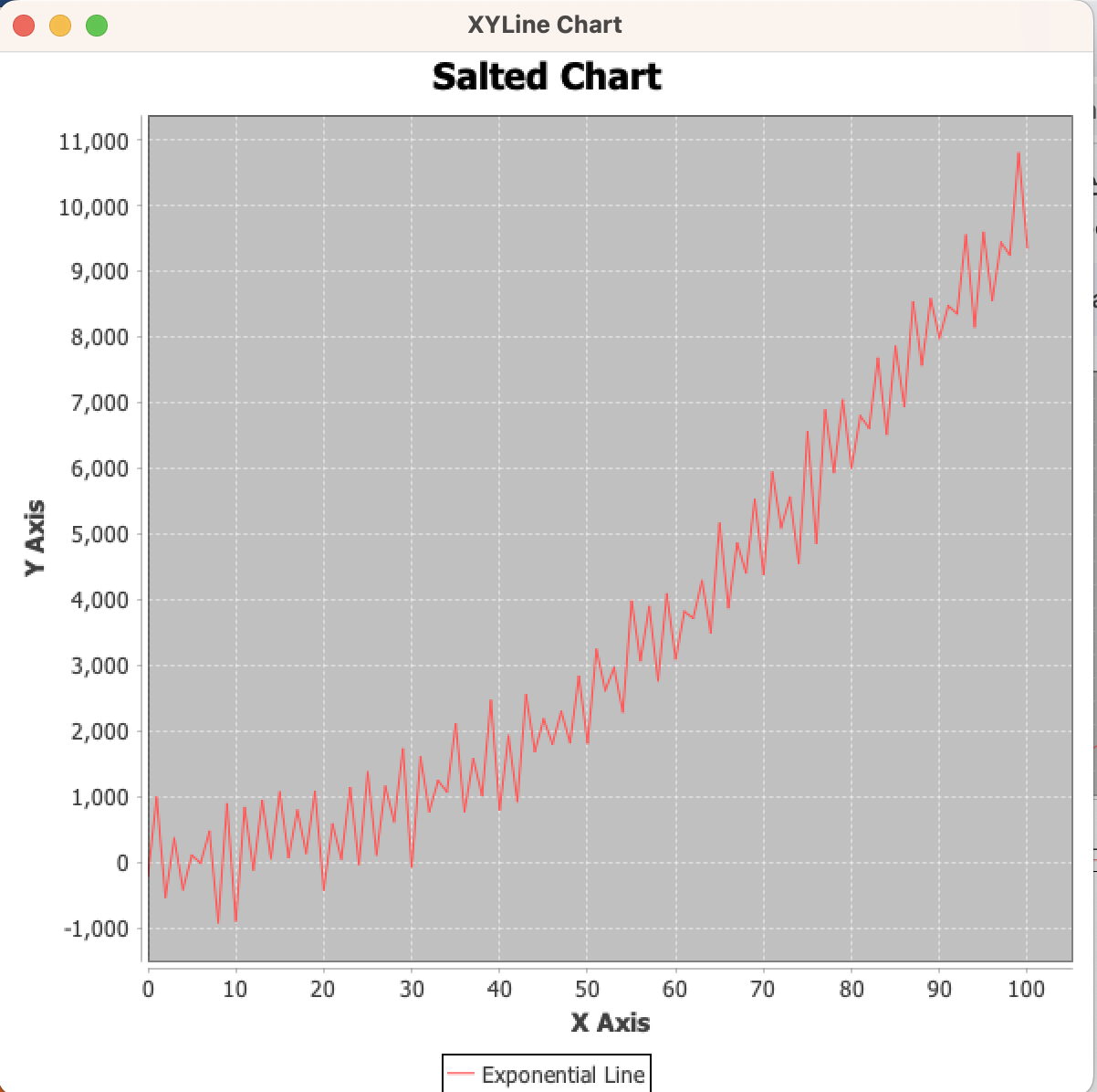
Results

Normal Chart



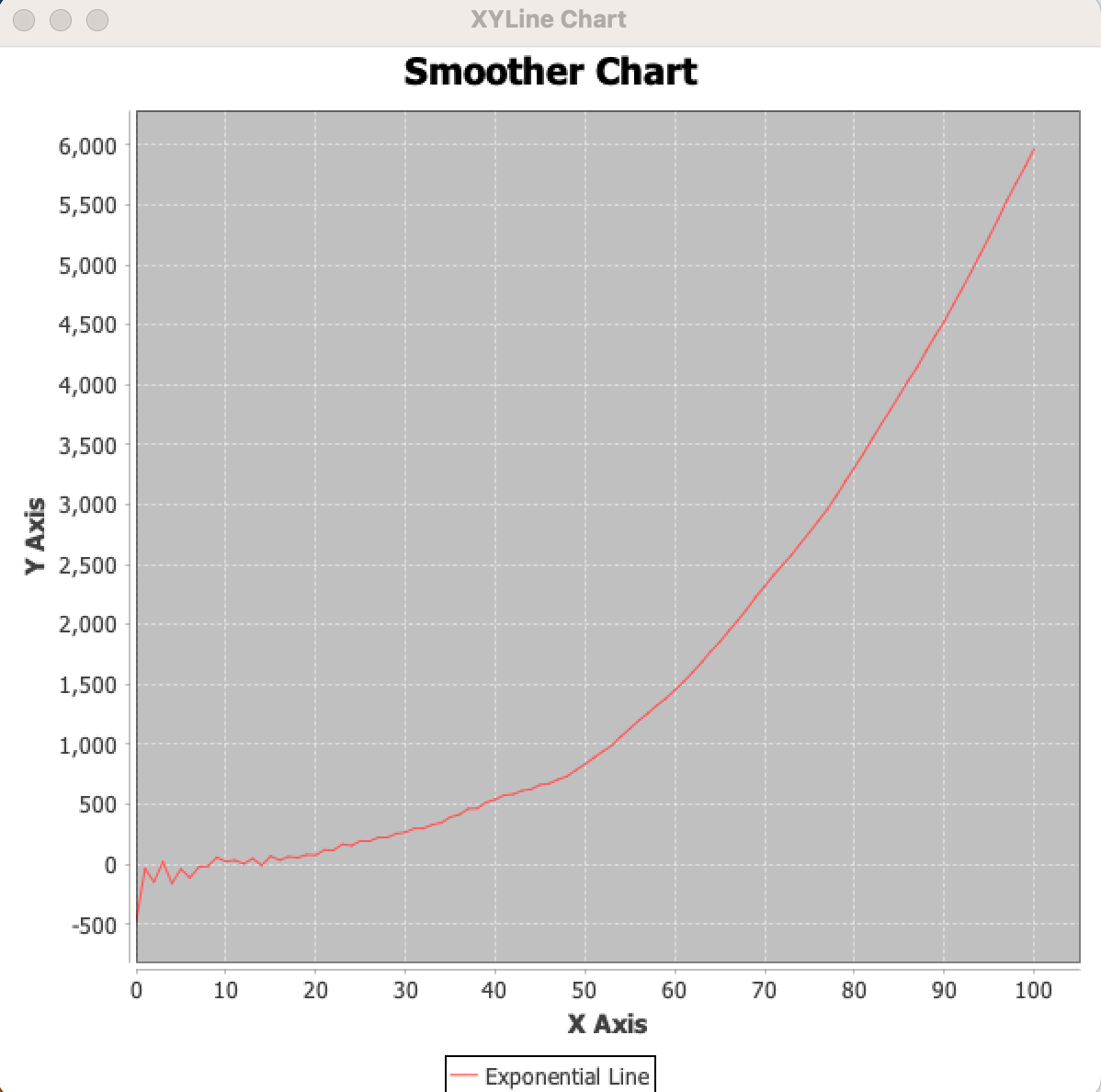
This is the normal chart I made using JfreeChart and Apache libraries. As you can see, the line is exponential as it goes from 0 to 100 in the x axis. For each value in the x axis, every y value was . Ultimately creating exponential y values that ranged from 0 to 10,000.

Salted Chart



This is the salted chart I made by using JfreeChart and Apache libraries. What makes this a salted chart is the volatility that the line expresses as it goes from 0 to 100. The jaggedness was created by subtracting random values for every even y value in the graph. Then, I did something similar for every odd y value and added a random number to it. The random numbers ranged between 10 to 1000, and created the noise that you can see in the graph above.

Smoother Chart



The purpose of this graph was to be able to smooth the salted values that we created in the previous graph. As you can see, I was able to smooth the graph for all values past 10. I did this by using the MovingAverage class in the JFreeChart library and was able to calculate the average for every set of 50 y values. Ultimately leading to the smoothed chart that you can see above.