Project 2 Extra Credit Considerations

When going through part 1 of this project it almost felt as if there was a lack of content within it, especially with PSS 3. So I decided to create a JFreeCharts GUI application that utilizes all parts of Project 2 Part 1. The Classes that do this are located in the folder located at the File Path: src\main\java\ExtraCredit

- **ChartAppCreator.java** | File Path: src\main\java\ExtraCredit\ChartAppCreator.java
  - ***graphApplication():*** The objective of this method is to utilize all parts of Project 2 Part1, All this method intakes though is the String title as there are other classes that act as pop up windows that utilize data.
  - The first chunk of this method is dedicated to setting up the Series line attributes that will be utilized later on with the graphing functionality of this method. This is followed by a section setting up the applications JFrame, textbox for the Initial-X value, and a dropdown menu to choose which function you would like to graph.
  - Following this is a section dedicated creating the different Jbutton functionalities of this application. These buttons include
    - **Graph Button:** which takes the initial-x value and graphs the selected function from the drop-down menu and adds it to a base dataset arraylist. If the selected function is the exponential function, then a pop-up window will ask you to enter the value of *n* before it graphs the function. It
      - I decided to have a quick JOptionPane pop up window rather than a typical JTextField since there is another function that would not utilize the value, making it so that the initial window was cleaner.
    - **Salting Button:** This button takes the baseDataset provided by the graphing button and calls the salting method from FunctionPlot.java, then adds that dataset to a SaltedDataSet array list.
    - **Smoothing Button:** Prior to the smoothing button there is a JTextField that requires the user to input a Window Value integer. This button then takes that window value and saltedDataSet array list and calls the smoothing function from FunctionPlot.java and sets the return value to the smoothedDataSet array list.
    - **Clear Button:** This button is simply used to clear the graph and corresponding datasets so that the user may create a new graph.
    - **Export Button:** The export button calls the exportWindowPop() method from the ExportWindow Class

    - The final portion of this is used to create a JPanel that holds all the previously mentioned buttons and text fields and organize them properly within it. The final line of code being used to pack it all properly within the window on launch.
- **ExportWIndow.java** | src\main\java\ExtraCredit\ExportWindow.java
    - ***exportWindowPop():*** This is a pop up window that requires more than just a JOptionPane. This is due to the fact that it intakes all 3 datasets and Graph Title from graphApplication(). Using these values, it calls on the Datahandler.java method csvOverWriter() and the ChartImgCreator.java method groupChartApp(). It creates three separate CSV files for the respective datasets using the csvOverWriter(), and creates a group chart PNG using the groupChartApp().
    - It has a JtextField asking for you to enter a file path that you would like to export these files to. Once you hit the button it creates a new folder in this location named after the graph title, and it holds the 3 csv files and the graph PNG as well.
- **GraphTitleWindow.java** | src\main\java\ExtraCredit\GraphTitleWindow.java
    - ***guiGraphTitleWindow():*** This is a simple method that is used to prompt the user on what they would like their graph to be named prior to the gui application starting up. It is made up of a JTextfield and a Start app button that is used to pass along the title information the graphApplication() method.

Test outputs of this application are stored in the file path: ProjectOutput\ExtraCreditOutput

Maven

This project used Maven's standard structure in order to manage its source code, resources, and a few of its outputs. While I did not heavily customize my build system it is important to note that Maven allows for extremely easy integration of external libraries/API's via the pom.xml. Some examples of libraries used in this project were JFreeCharts and JUnits. JfreeCharts allowing me to graph my csv charts, and JUnits allowing me to test my StatsLibrary.java outputs for correct outputs. The code for that can be seen at this file path: src\test\java\StatsTest.java