**FunctionPlotter User Manual**

A Java-based collection of programs to plot data, salt data, and smoothen data.

Cachary Tolentino

CSCI-3327

**Table of Contents**

**Software Description3**

Detailed Description3.1

System Requirements3.2

**Installation Guide4**

**Class Overview6**

**Software Description**

A Java-based collection of programs to plot data, salt data, and smoothen data.

**Detailed Description**

The FunctionPlotter program allows the user to generate a csv file with plot point s of a generated quadratic function. The program will initially ask the user for the name that they wish the file to be called. From then the u ser will define up to how many trials (or X) they want plotted. Then the values for the a, b, and c value s in the quadratic function. The program will also provide a salted variant of the original data as well as a smoothen data.

**System Requirements**

* A working device, primarily a desktop or laptop
* An IDE (ex: VSCode, Eclipse, etc…)
* Java JDK (Ver. 17 & up) & JRE (SE 17 & up)

**Installation Guide**

To begin using FunctionPlotter you will need to download two files. One is “QuadraticPlotter.java” and the other is “TestQuadraticPlotter.java” (optional).

A black background with white text

AI-generated content may be incorrect.

After downloading the files, simply move the files to the folder containing your project. Once done, you can open your preferred IDE (for this example we will be using VSCode). Then you can open the folder or the file itself within your IDE.

A screenshot of a computer

AI-generated content may be incorrect.

If you opened the folder containing the files then it should look similar to the image below.

A screenshot of a computer

AI-generated content may be incorrect.

If you only imported the QuadraticPlotter file then you can simply start using the class within your own personal project. Otherwise, if you also imported the TestQuadraticPlotter, then you can open that file and run it.

The result will be displayed on the console and the file will be generated in the folder that contains the program files.

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer menu

AI-generated content may be incorrect.

**A screenshot of a computer

AI-generated content may be incorrect.**

**Class Overview**

TestPlotter Class

**main Function**

The function simply calls on the various functions from the QuadraticPlotter, Salter, and Smoother classes. It creates an instance of each object type and calls on their respective function such as plotData(), salterData(), and smoothenData().

QuadraticPlotter Class

**quadraticFunction Function**

The function simply calculates the quadratic function value (Y). The function requires three parameters: x (double), a (double), b(double), c(double). The function will then return the Y value as type double.

**plotData Function**

The function has no parameters but returns a string value. It will perform four main tasks. It will first ask the user for inputs regarding the quadratic function (inputs for x, a, b, and c values). It will then iterate through the given amount x values and add its corresponding y value by calling the quadraticFunction(). These are then added to a data structure (an arraylist of strings). It will then export the data structure into a csv file user the exporter function of the DataHandler class. Finally it will return the name of the file.

Salter Class

**Salter Constructor**

The salter constructor takes in three parameter values: inputFile, lower, and upper. They are of type File, int and int.

**salter Function**

The salter function has a single parameter: data which is of type ArrayList<String>. The function will return newData which if of type ArrayList<Double>. The function will salt the data by randomly choosing to either add or subtract a unique salt value to the current Y value. It will purposefully skip all the X values (found at odd positions). Each salted data or X values is then added onto newData, once finished iterating and salting data, then newData is returned.

**salterData Function**

The salterData function has no parameters but returns a string value. The function is responsible for parsing the inputted data file using the DataHandler’s parser function. Once the data has been parsed, it will call on the salter() function to salt each Y data value. Finally, the newly salted data is exported using the DataHandler’s exporter function and the function returns the new file name of the salted data.

Smoother Class

**Smoother Constructor**

The smoother constructor has two parameter values: inputFile and value, which are of type File and double.

**Smoother Function**

The function has a single parameter: data which is of type ArrayList<String>. This function is responsible for smoothing the given data set (assumed to be salted). The function does so by using a windowValue variable that is defined in the constructor. It will iterate through the entire data structure containing both the X and Y values (ignoring the X’s). For each Y value, it will find the left most position allowed by the windowValue and the right most position allowed by the windowValue. It will then average all the values within the found range of left and right, this new average value will be used to replace the current Y value. Once all the data values have been iterated, the function simply returns the ArrayList<Double> containing the newly smoothened data.

**smoothenData Function**

The smoothenData function has no parameters and no return value. The function is responsible for parsing the inputted data file using the DataHandler’s parser function. Once the data has been parsed, it will call on the smoother() function to smoothen each Y data value. Finally, the newly smoothened data is exported using the DataHandler’s exporter function and the function returns the new file name of the smoothened data.

DataHandler Class

**parser Function**

The function has a single parameter: dataFile which is of type File. This file contains the none-parsed data. The function will iterate through the file line by line and split each line to their own respective values. The inputs follow the following pattern: “X, Y”. Thus, the parser simply splits at the “,”, and adds the X and Y values from the string[] variable. Once there are no more lines to iterate through, it simply returns the newly created data structure containing the parsed data which is of type ArrayList<Double>.

**writer Function**

The writer function has a single parameter: data which is of type ArrayList<Double>. This function is responsible for adding the X and Y values from the data as string values in the following pattern: “X, Y”. It will iterate through all the data (every pair) and once done it will return the value which is of type ArrayList<String> containing the X and Y values.

**exporter Function**

The exporter function has two parameters: data and fileName which are of type ArrayList<String> and String. The function will export the provided data into a csv file. By using FileWriter, it will add a string in each line in the following pattern: “X, Y”. The function does not return any value.