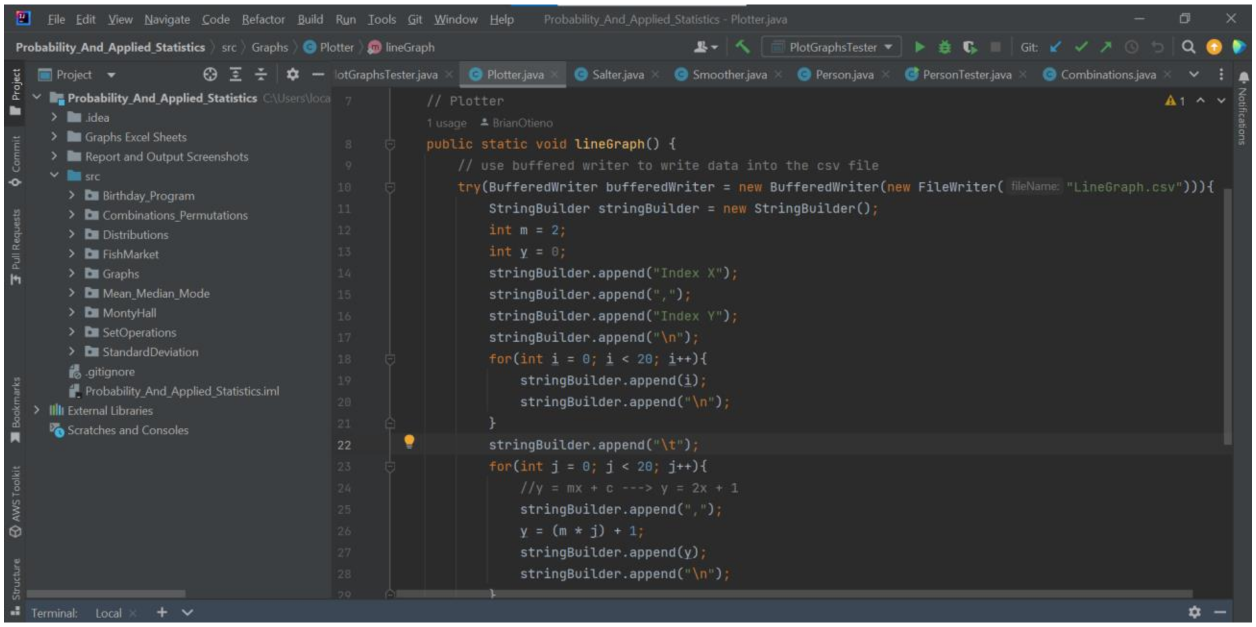
GRAPH SOLUTIONS

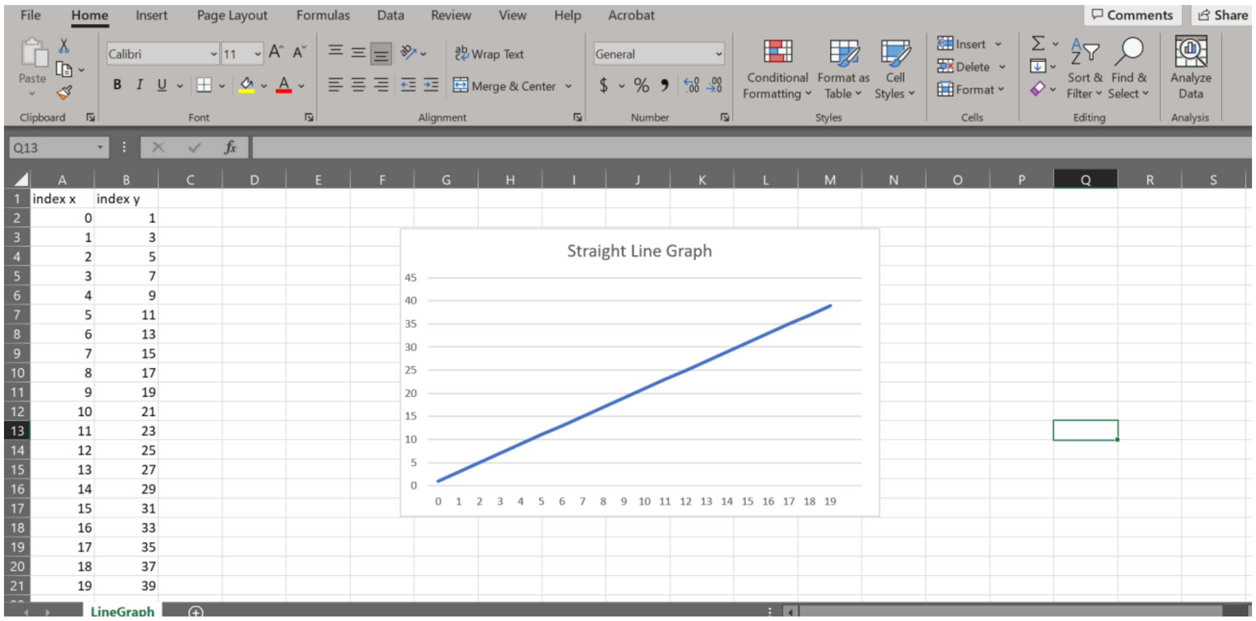
To solve the Graph assignment, I made four classes. One class that will work on salting the graph i.e., Salter Class, one that would plot a straight line (Plotter class), one that would smooth the salted graph (Smoother class) and finally the tester class where the main method will run.

PLOTTER CLASS

For this class, I created a static method called linegraph, which plots the graph y = 2x + 1. A sample from the formulae y=mx +c. I imported the BufferedWriter and FileWriter classes so that I can write and export my data to the Linegraph.csv file which I will use to plot a straight-line graph in excel. I placed the previously mentioned classes in a try catch method so that to efficiently close resources and avoid instances of memory leaks. I also imported the StringBuilder class and used the method append () to provide structure on how my values should appear in the excel spread sheet. There are two for loops in the static method. The first loop is for displaying the coordinates for index x over an iteration of 20 items, while the second loop is for getting the y coordinates through the formulae y = 2x + 1.



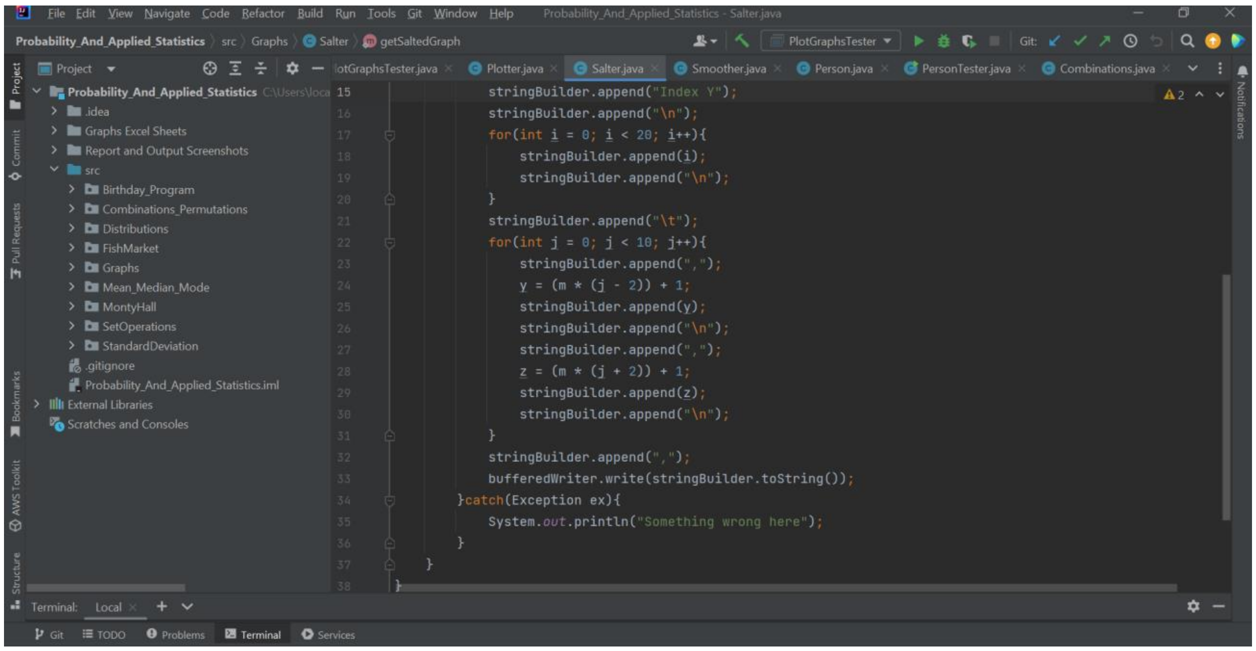
```java
// Plotter
public static void lineGraph() {
    // use buffered writer to write data into the csv file
    try(BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter("LineGraph.csv"))){
        StringBuilder stringBuilder = new StringBuilder();
        int m = 2;
        int y = 0;
        stringBuilder.append("Index X");
        stringBuilder.append(",");
        stringBuilder.append("Index Y");
        stringBuilder.append("\n");
        for(int i = 0; i < 20; i++){
            stringBuilder.append(i);
            stringBuilder.append("\n");
        }
        stringBuilder.append("\t");
        for(int j = 0; j < 20; j++){
            //y = mx + c ---> y = 2x + 1
            stringBuilder.append(",");
            y = (m * j) + 1;
            stringBuilder.append(y);
            stringBuilder.append("\n");
        }
```



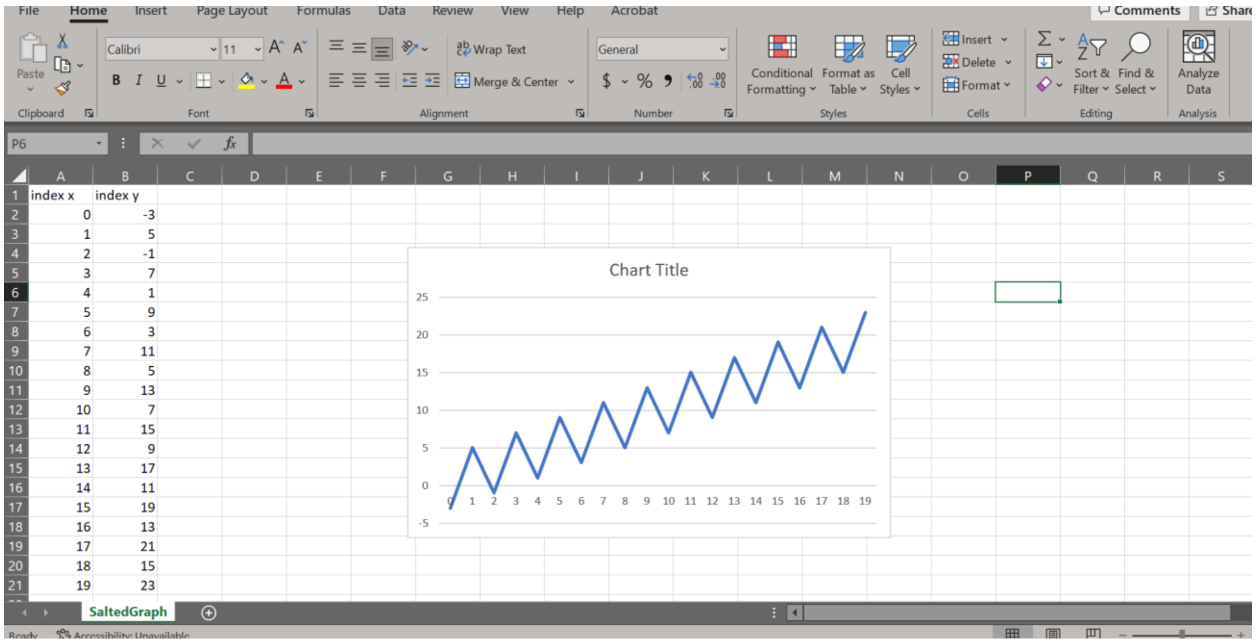
index x	index y
0	1
1	3
2	5
3	7
4	9
5	11
6	13
7	15
8	17
9	19
10	21
11	23
12	25
13	27
14	29
15	31
16	33
17	35
18	37
19	39

SALTER CLASS

For this class, I created a static method which plots the salted graph of y = 2x + 1. I imported the BufferedWriter, FileWriter and StringBuilder classes just like the way I did on the Plotter Class. My data was being exported to the SaltedGraph.csv file, which I will use to plot the graph in excel. The only difference in implementation I made in this class was altering the formulae to y = 2*(x - 2) + 1 and y = 2 * (x + 2) + 1 in the respective for loops so that the output when plotted would appear in a zig zag pattern as demonstrated below to show that the graph is being salted.



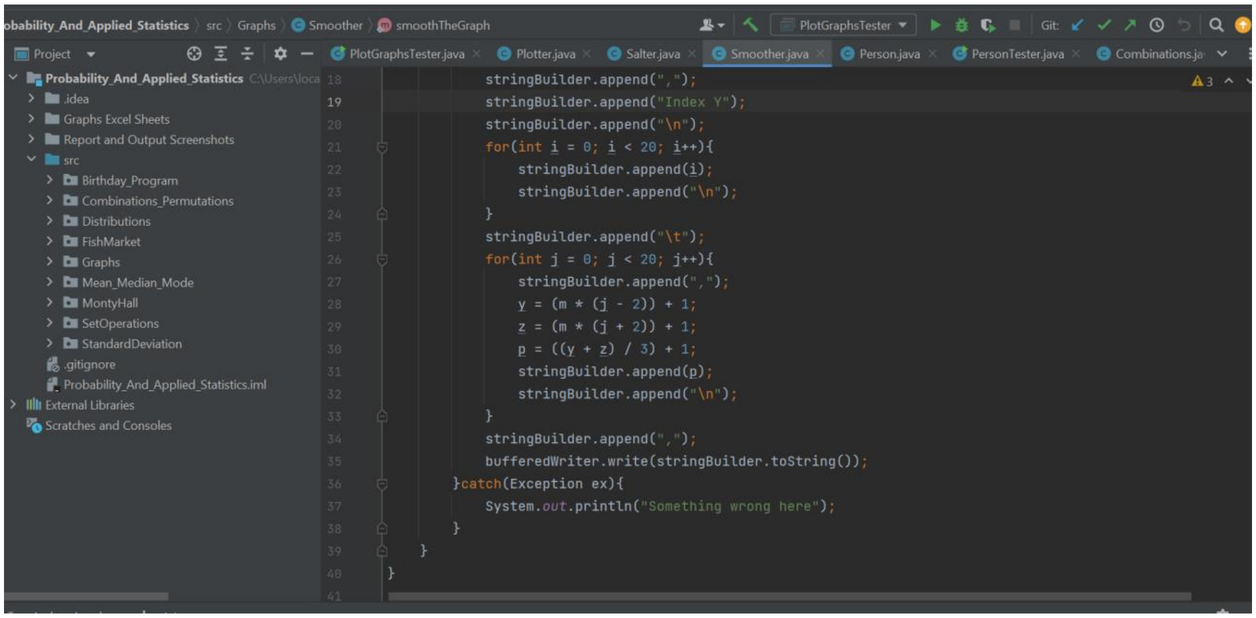
```java
            stringBuilder.append("Index Y");
            stringBuilder.append("\n");
            for(int i = 0; i < 20; i++){
                stringBuilder.append(i);
                stringBuilder.append("\n");
            }
            stringBuilder.append("\t");
            for(int j = 0; j < 10; j++){
                stringBuilder.append(",");
                y = (m * (j - 2)) + 1;
                stringBuilder.append(y);
                stringBuilder.append("\n");
                stringBuilder.append(",");
                z = (m * (j + 2)) + 1;
                stringBuilder.append(z);
                stringBuilder.append("\n");
            }
            stringBuilder.append(",");
            bufferedWriter.write(stringBuilder.toString());
        }catch(Exception ex){
            System.out.println("Something wrong here");
        }
    }
}
```



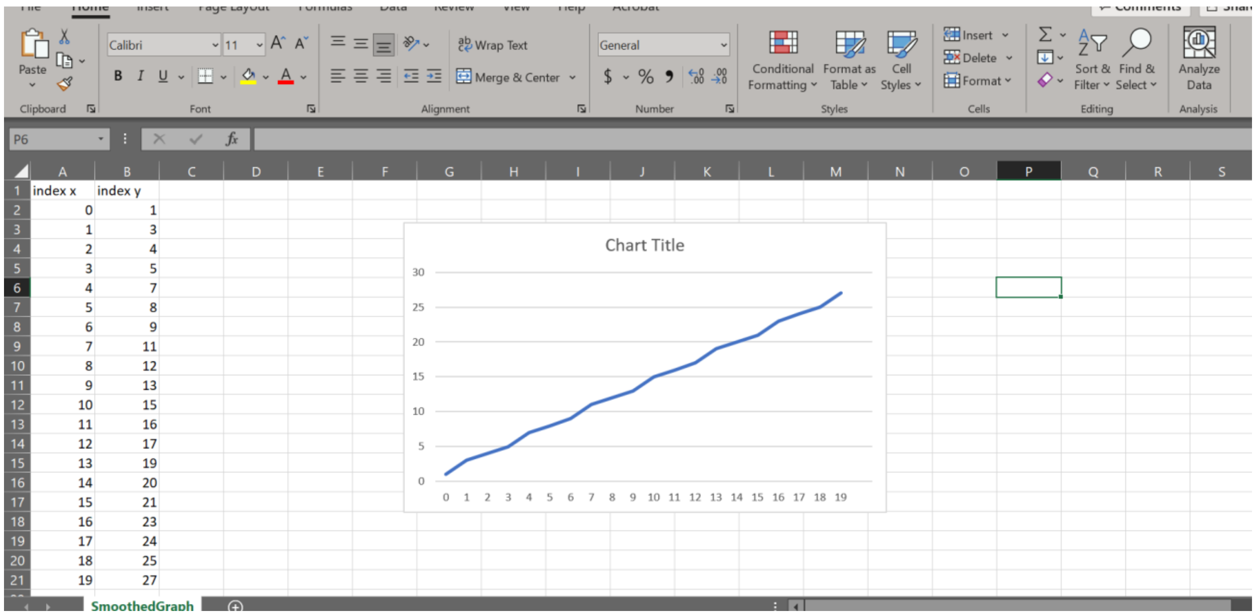
index x	index y
0	-3
1	5
2	-1
3	7
4	1
5	9
6	3
7	11
8	5
9	13
10	7
11	15
12	9
13	17
14	11
15	19
16	13
17	21
18	15
19	23

SMOOTHED CLASS

For this class, I created another static method just like the two previous classes. I imported the BufferedWriter, FileWriter and StringBuilder classes so that I can write and export my data to the SmoothedGraph.csv file. Since the objective was to smoothen the salted graph, I decided to alter the formula I used to plot the salted graph. I changed the implementation to p = ((y + z) / 3) + 1, with the z being the variable that represents the second formula of y = 2*(x +2) +1. This is because in Java, it would result in an error, having two different formulas, sharing the same variable name. The excel sheet below, shows how the graph resembles after going through the smoothing process.



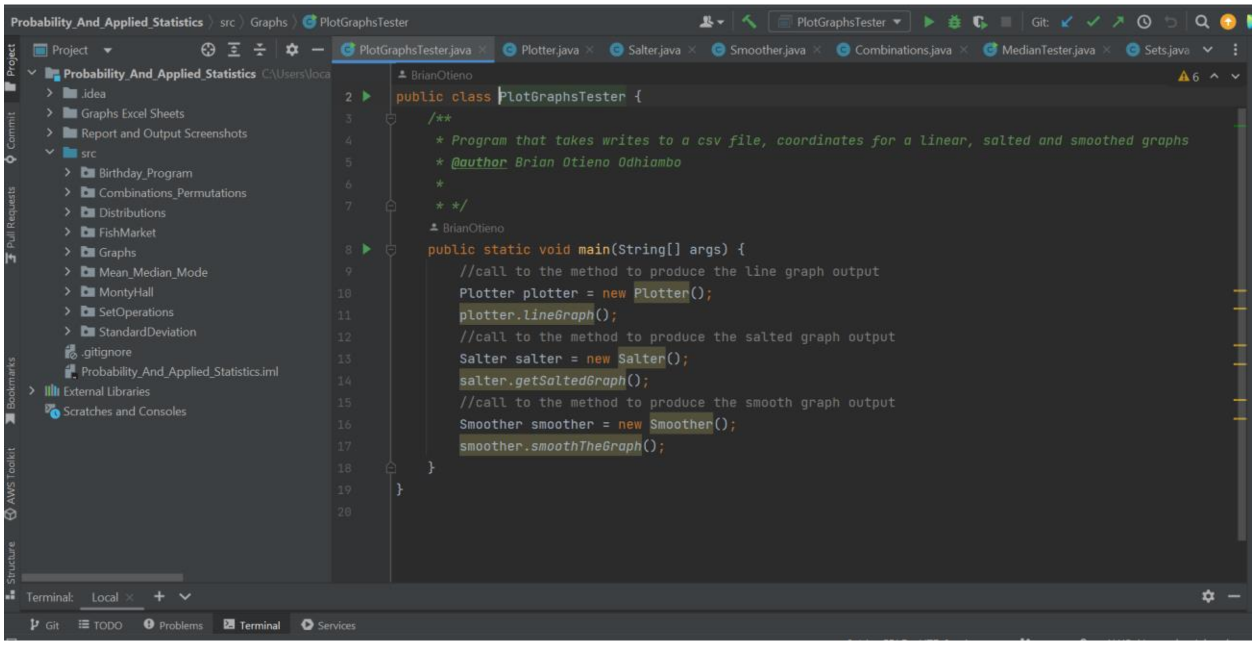
```
            stringBuilder.append(",");
            stringBuilder.append("Index Y");
            stringBuilder.append("\n");
            for(int i = 0; i < 20; i++){
                stringBuilder.append(i);
                stringBuilder.append("\n");
            }
            stringBuilder.append("\t");
            for(int j = 0; j < 20; j++){
                stringBuilder.append(",");
                y = (m * (j - 2)) + 1;
                z = (m * (j + 2)) + 1;
                p = ((y + z) / 3) + 1;
                stringBuilder.append(p);
                stringBuilder.append("\n");
            }
            stringBuilder.append(",");
            bufferedWriter.write(stringBuilder.toString());
        }catch(Exception ex){
            System.out.println("Something wrong here");
        }
    }
}
```



index x	index y
0	1
1	3
2	4
3	5
4	7
5	8
6	9
7	11
8	12
9	13
10	15
11	16
12	17
13	19
14	20
15	21
16	23
17	24
18	25
19	27

THE MAIN METHOD

Finally, The PlotGraphTester class is where I placed the main method, that will have calls to the three different classes. By running the code below, the objects of the classes created were used to call the static methods, which result to the code getting compiled and the 3 respective excel documents were created with X and Y coordinates.



```java
public class PlotGraphsTester {
    /**
     * Program that takes writes to a csv file, coordinates for a linear, salted and smoothed graphs
     * @author Brian Otieno Odhiambo
     *
     * */
    public static void main(String[] args) {
        //call to the method to produce the line graph output
        Plotter plotter = new Plotter();
        plotter.lineGraph();
        //call to the method to produce the salted graph output
        Salter salter = new Salter();
        salter.getSaltedGraph();
        //call to the method to produce the smooth graph output
        Smoother smoother = new Smoother();
        smoother.smoothTheGraph();
    }
}
```