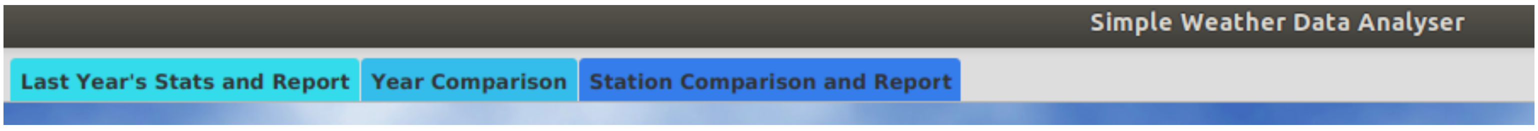
User Guide & Design Strategy:

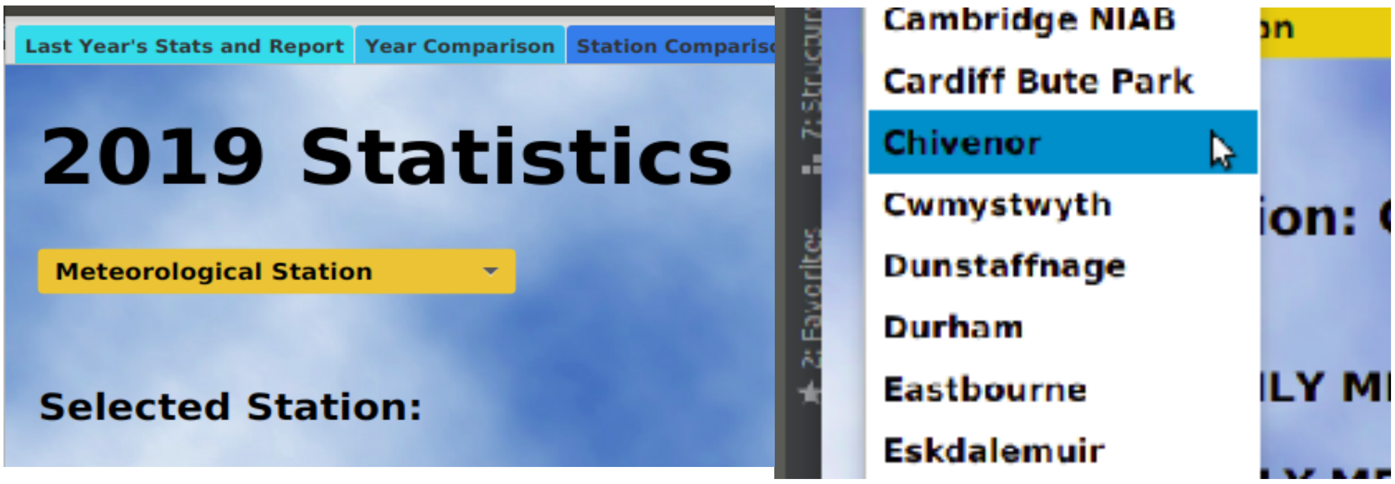
NOTE: All Extra features are labelled in letters from A to H

Basic Features - Implementing Client's (Assignment's) requirements:

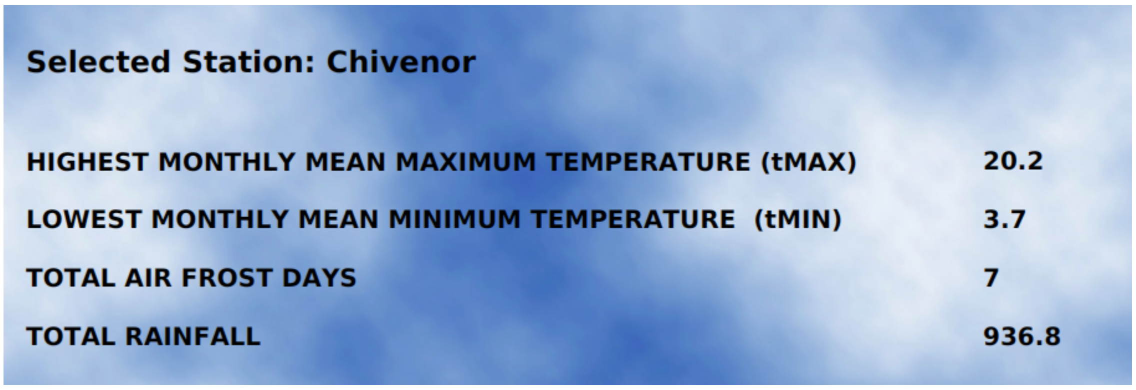
Current "Simple Weather Data Analyser" version has been solely designed as a desktop application and currently does not support mobile phone responsiveness. Nevertheless, the application has been designed with **simplicity and convenience** in mind for the user, thus application's window size can be reduced or expanded. To keep complexity at its bare minimum, the application analyses historic meteorological station data in 3 different GUI's **Tabs**:


Simple Weather Data Analyser
Last Year's Stats and Report
Year Comparison
Station Comparison and Report

1. "Last Year's Stats and Report" (Requirement 1) - Using a GridPane, to ensure a quick access to required information, application shows a snapshot of each Meteorological Station's 2019 statistics (tMax, tMin, Total Air Frost Days, Total Rainfall), which are selectable from a dropdown menu:



For example, selecting "Chivenor" displays:



Selected Station: Chivenor

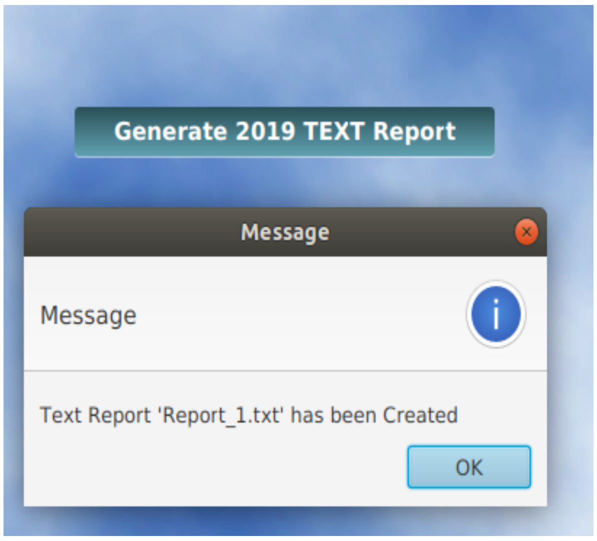
HIGHEST MONTHLY MEAN MAXIMUM TEMPERATURE (tMAX)	20.2
LOWEST MONTHLY MEAN MINIMUM TEMPERATURE (tMIN)	3.7
TOTAL AIR FROST DAYS	7
TOTAL RAINFALL	936.8

Additional Features in "Last Year's Stats and Report" Tab

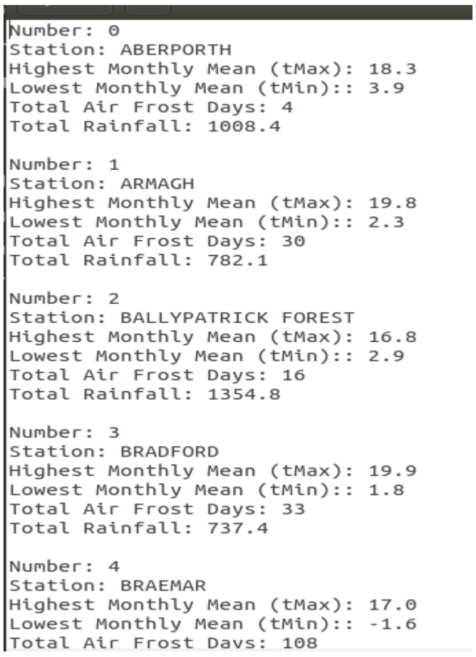
In case the user wanted to examine the data deeper by comparing it to other stations' 2019 statistics, this can be accessed in 2 ways:

A. Generating a Comprehensive Text Report by clicking "Generate 2019 Text Report" Button:

Application creates "Report_1.txt" file and report shows same statistics with different data per each station:

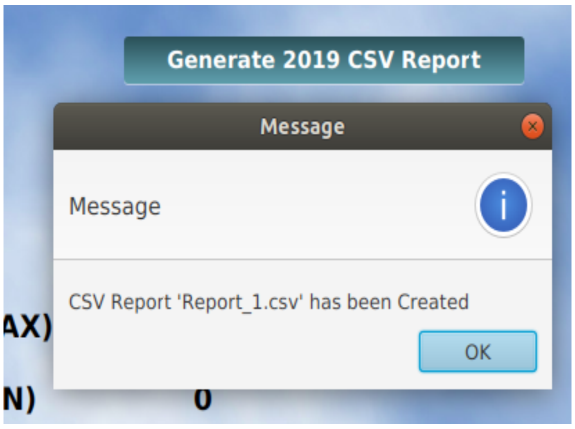


Text Report 'Report_1.txt' has been Created

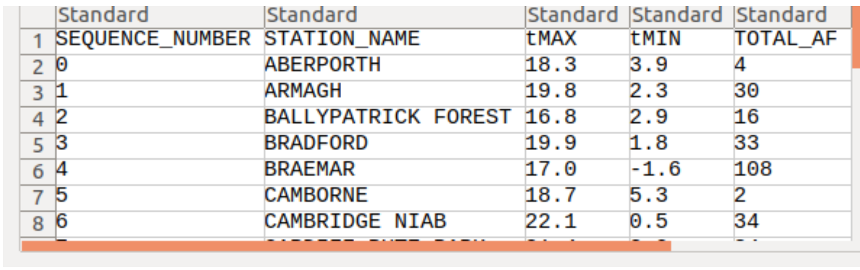


```
Number: 0
Station: ABERPORTH
Highest Monthly Mean (tMax): 18.3
Lowest Monthly Mean (tMin):: 3.9
Total Air Frost Days: 4
Total Rainfall: 1008.4

Number: 1
Station: ARMAGH
Highest Monthly Mean (tMax): 19.8
Lowest Monthly Mean (tMin):: 2.3
Total Air Frost Days: 30
Total Rainfall: 782.1

Number: 2
Station: BALLYPATRICK FOREST
Highest Monthly Mean (tMax): 16.8
Lowest Monthly Mean (tMin):: 2.9
Total Air Frost Days: 16
Total Rainfall: 1354.8

Number: 3
Station: BRADFORD
Highest Monthly Mean (tMax): 19.9
Lowest Monthly Mean (tMin):: 1.8
Total Air Frost Days: 33
Total Rainfall: 737.4

Number: 4
Station: BRAEMAR
Highest Monthly Mean (tMax): 17.0
Lowest Monthly Mean (tMin):: -1.6
Total Air Frost Days: 108
```

B. Otherwise, if the user wanted to conduct further analysis, or read/share the data as a spreadsheet, the same information can be reported in a **different layout,** as a **.csv file by clicking "Generate 2019 CSV Report:**



CSV Report 'Report_1.csv' has been Created

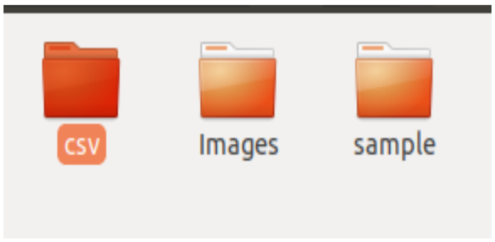


	Standard	Standard	Standard	Standard	Standard
1	SEQUENCE_NUMBER	STATION_NAME	tMAX	tMIN	TOTAL_AF
2	0	ABERPORTH	18.3	3.9	4
3	1	ARMAGH	19.8	2.3	30
4	2	BALLYPATRICK FOREST	16.8	2.9	16
5	3	BRADFORD	19.9	1.8	33
6	4	BRAEMAR	17.0	-1.6	108
7	5	CAMBORNE	18.7	5.3	2
8	6	CAMBRIDGE NIAB	22.1	0.5	34

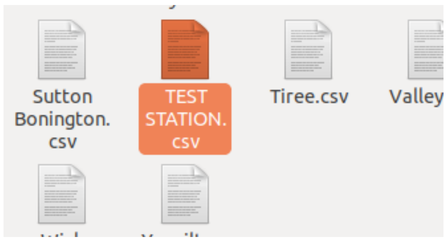
C. Another Additional Feature that applies to the whole Application is the dynamic reading of stations.

Using Java Files.walk method by mkyong (2018) any **.csv** file compiled in a format same as the provided stations for this project, can be added in the "**csv**" folder and the station's name and data will be included in all application's station selection dropdown menus, while its data will be handled same as for other stations, thus creating for the user an **effortless and seamless** system for updating or analysing different stations:

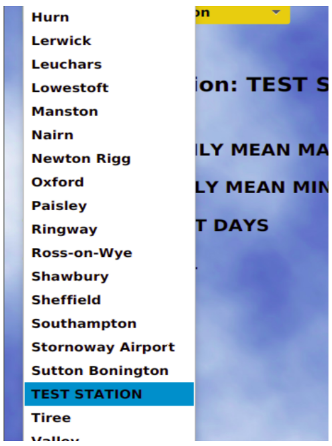
1.



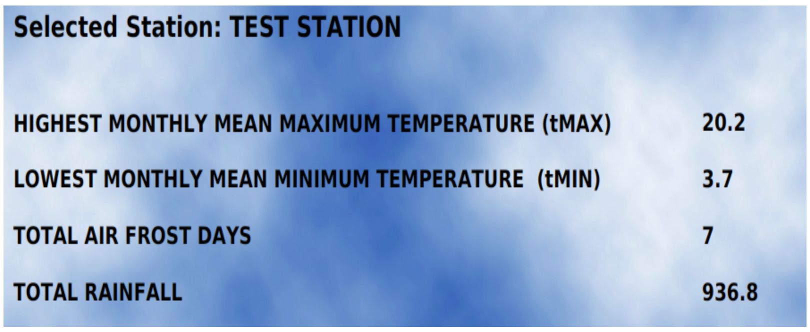
2.



3



4.

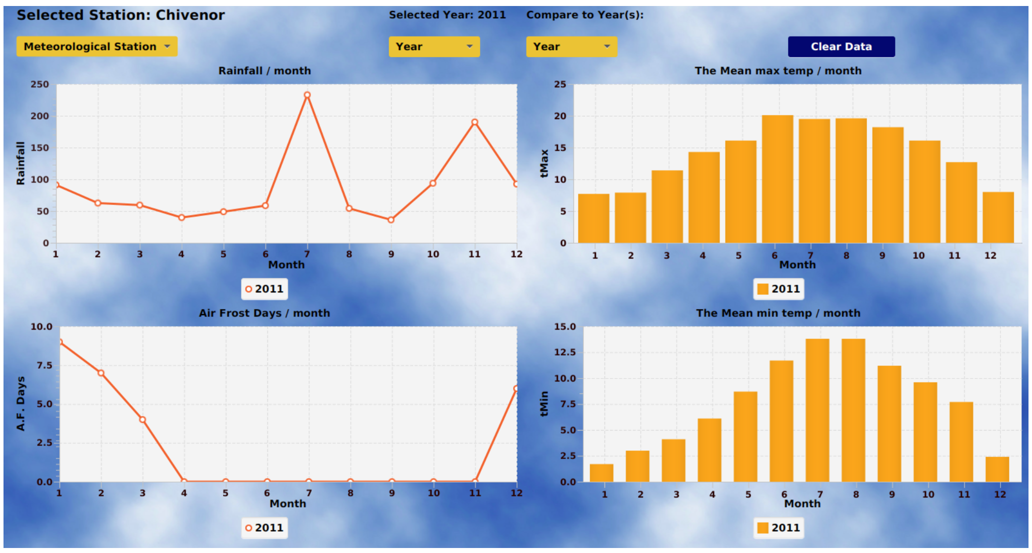


Selected Station: TEST STATION

HIGHEST MONTHLY MEAN MAXIMUM TEMPERATURE (tMAX)	20.2
LOWEST MONTHLY MEAN MINIMUM TEMPERATURE (tMIN)	3.7
TOTAL AIR FROST DAYS	7
TOTAL RAINFALL	936.8

(Chivenor.csv was copied and renamed to TEST STATION.csv to demonstrate the example).

2. "Year Comparison" (Requirement 2) This section allows the user to view each station's tMax, tMin, Air frost Days and Rainfall historic figures in more depth than in previous tab, by being able to display the data for every available year and month over 4 charts:

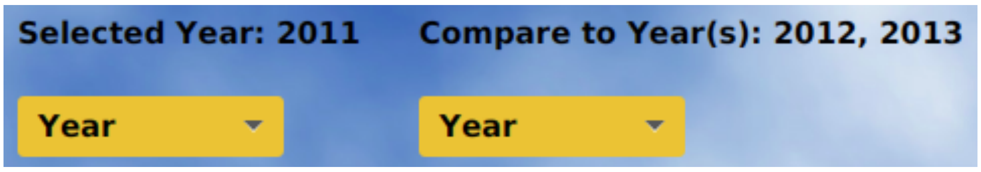


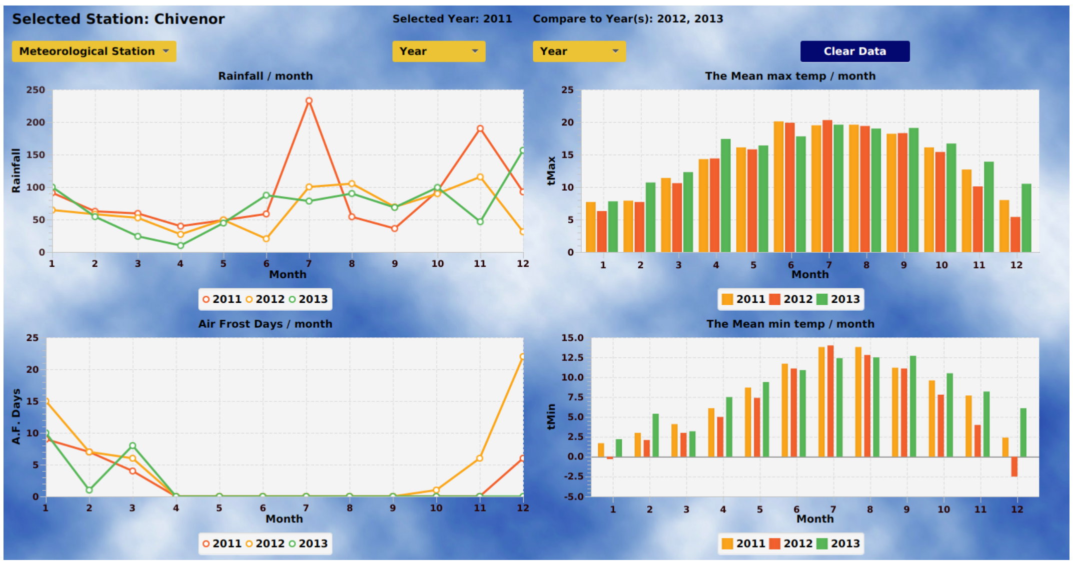
Both temperature related datasets have been chosen to be displayed as column Bar-charts to emphasise the comparison relationship, based on Abela's (2006) data visualisation framework for good practices. Additionally, Air Frost Days are shown on Line-chart for better visual when data is empty, while Rainfall Line-chart also fits Abela's principles, in showing change over time.

Charts can be cleared by clicking

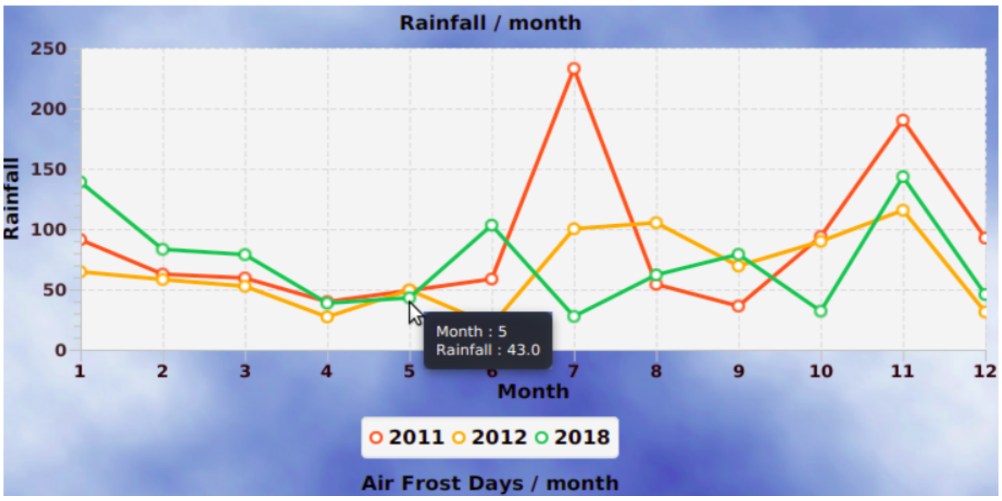
Additional Features in "Year Comparison" Tab

D. Users can compare every station's year/month against other years via below dropdown menus:



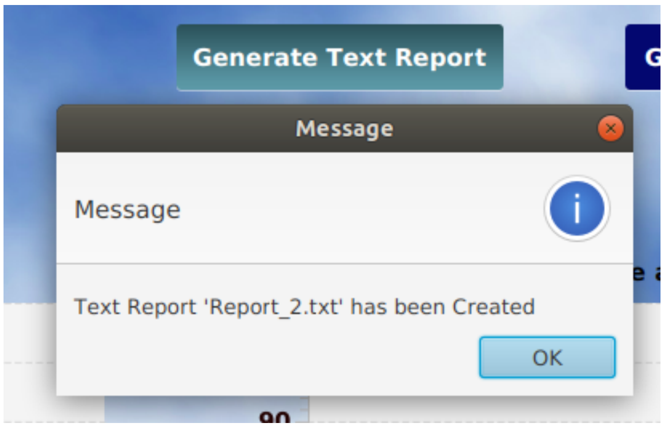
Result: 

E. To increase visual clarity and data comparison precision, hover labels have been added to chart nodes using Tooltip. Method adopted from ItachiUchiha (2018):



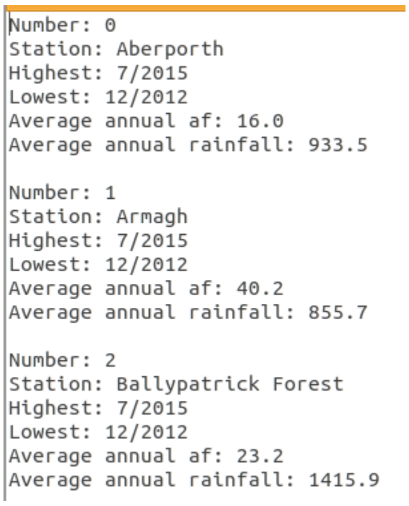
3. "Station Comparison and Report" - (Requirement 3)

This section works similarly as the extra features implemented in "Last Year's Stats and Report" tab, however the reported statistics include Average-Annual Rainfall and Average-Annual Air Frost Days, also year and month for highest tmax and lowest tmin. The basic requirement is fulfilled by clicking:



Message

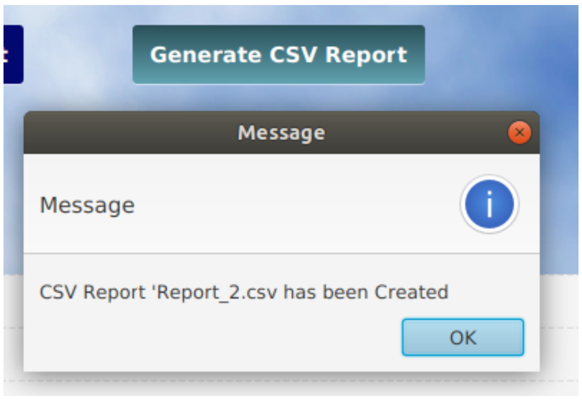
Text Report 'Report_2.txt' has been Created



```
Number: 0
Station: Aberporth
Highest: 7/2015
Lowest: 12/2012
Average annual af: 16.0
Average annual rainfall: 933.5

Number: 1
Station: Armagh
Highest: 7/2015
Lowest: 12/2012
Average annual af: 40.2
Average annual rainfall: 855.7

Number: 2
Station: Ballypatrick Forest
Highest: 7/2015
Lowest: 12/2012
Average annual af: 23.2
Average annual rainfall: 1415.9
```

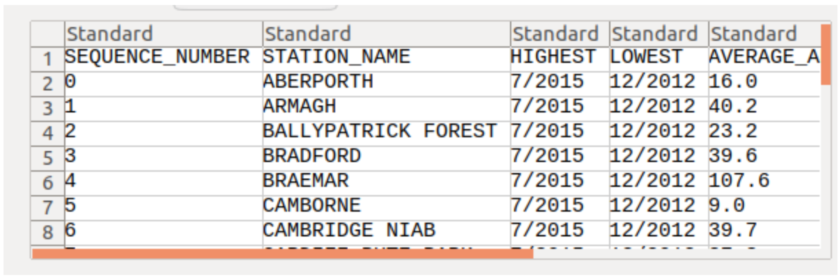
Additional Features in "Station Comparison and Report" Tab

F. **CSV format report can be generated instead of Text for the same purpose as in additional feature B.**



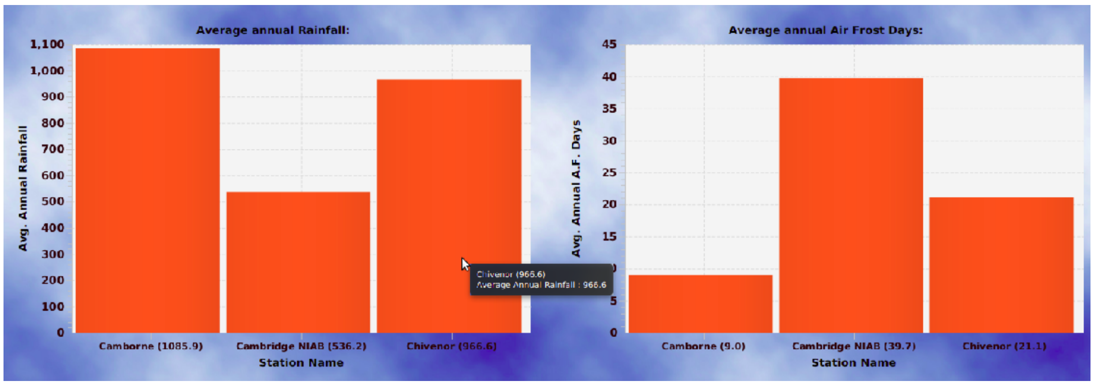
Message

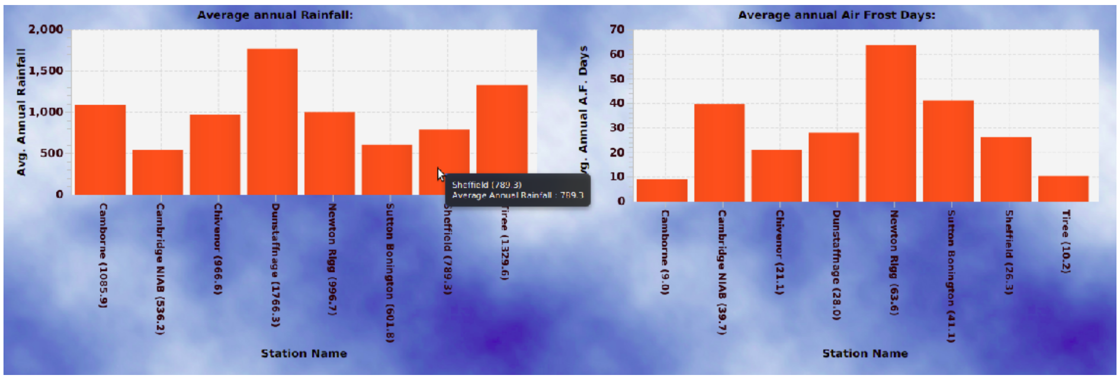
CSV Report 'Report_2.csv has been Created

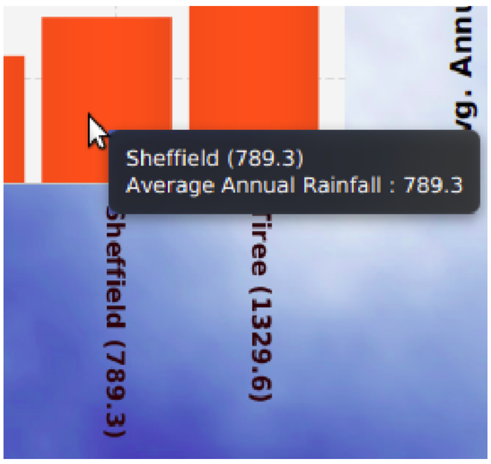


	Standard	Standard	Standard	Standard	Standard
1	SEQUENCE_NUMBER	STATION_NAME	HIGHEST	LOWEST	AVERAGE_A
2	0	ABERPORTH	7/2015	12/2012	16.0
3	1	ARMAGH	7/2015	12/2012	40.2
4	2	BALLYPATRICK FOREST	7/2015	12/2012	23.2
5	3	BRADFORD	7/2015	12/2012	39.6
6	4	BRAEMAR	7/2015	12/2012	107.6
7	5	CAMBORNE	7/2015	12/2012	9.0
8	6	CAMBRIDGE NIAB	7/2015	12/2012	39.7

G. **In this section the user can compare all stations in the** "csv" folder based on Average-Annual Rainfall and Average-Annual Air Frost Days. The hover labels (additional feature **E.**) have also been added for extra precision, however all station names, also include their node's value.

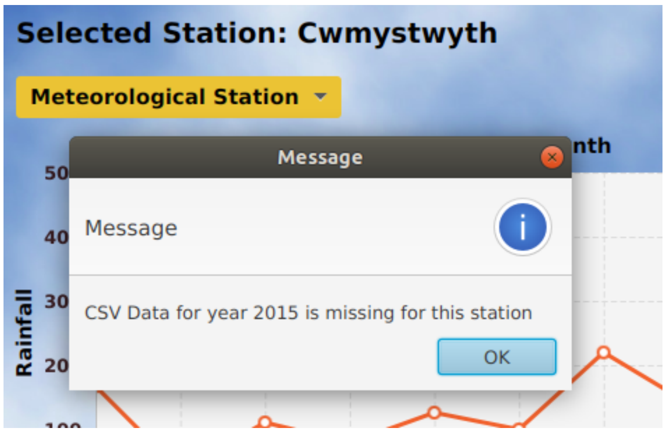





Sheffield (789.3)
Average Annual Rainfall : 789.3

H. Empty Data - handling Alert Boxes. (Robustness)

When a station is selected and its CSV file has no data for that particular year, alert box is displayed.


Selected Station: Cwmystwyth

Meteorological Station

Message

Message

CSV Data for year 2015 is missing for this station

OK

Implementation & Brief Algorithmic Description:

This JavaFX Application consists of the following:

1. **csv** folder - as described in User Guide & Design Strategy section, contains all the input data, which the algorithm utilises to perform calculations and extract requested information.

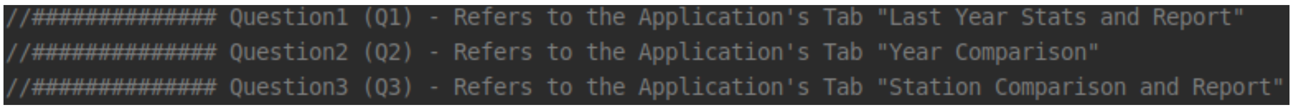
2.**Images** folder- contains the background image, which sets the application's theme.

3.**sample** folder - contains the following 4 files, which create the whole application:

"Main.java" - sets the scene, connects the **"sample.fxml"** via FXMLLoader, which holds all containers and objects created via JavaFX ScieneBuilder and also connects the **"style.css",** which applies CSS properties to the created elements and makes the application more aesthetically appealing to the user.

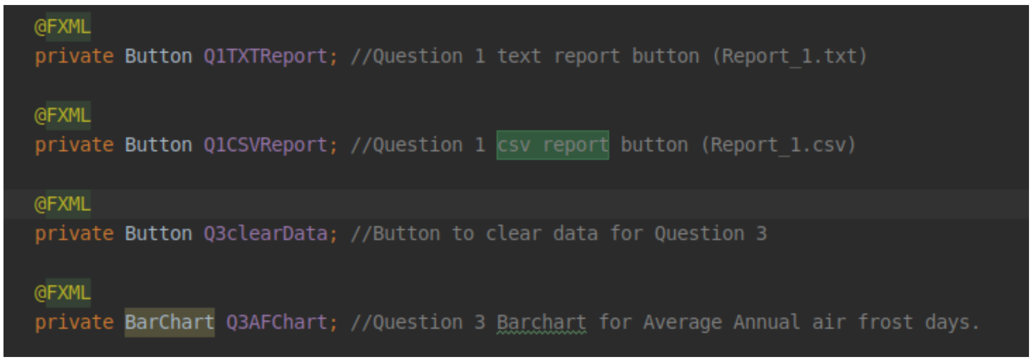
"Controller.java" - fully implements the functionality part of the application. The file contains highly detailed comments, which describe how the algorithm functions and what all the functions and lines of code are responsible for.

Requirements (Questions) have been addressed in variable names and algorithm's comments as follows:



```
//############## Question1 (Q1) - Refers to the Application's Tab "Last Year Stats and Report"
//############## Question2 (Q2) - Refers to the Application's Tab "Year Comparison"
//############## Question3 (Q3) - Refers to the Application's Tab "Station Comparison and Report"
```

The Created FXML object data is injected in the variables declared in **"Controller.java".**



```
@FXML
private Button Q1TXTReport; //Question 1 text report button (Report_1.txt)

@FXML
private Button Q1CSVReport; //Question 1 csv report button (Report_1.csv)

@FXML
private Button Q3clearData; //Button to clear data for Question 3

@FXML
private BarChart Q3AFChart; //Question 3 Barchart for Average Annual air frost days.
```

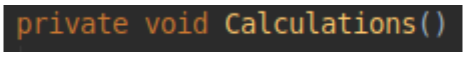
All 3 requirements have dedicated Controller class variables, in order to adequately utilise function scope.

```
//CONTROLLER CLASS VARIABLES:
```

The algorithm has been refactored and repetitive computation parts have been redefined into functions to maintain efficiency and code quality. Functions are defined from **line 168 to line 557** (between **csvReader()** and **initialize()** methods).

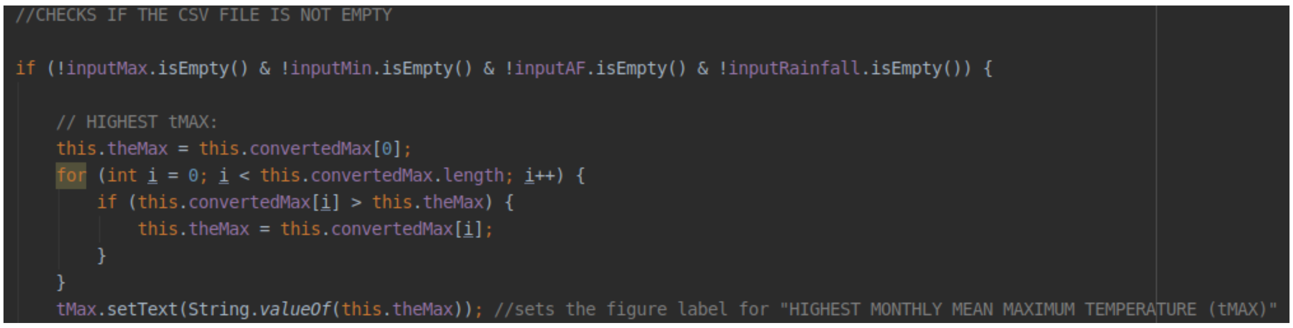
```
private void csvReader(String year, String StationName)
```

Parameters take year and station's name Strings, **dynamically** reads any file ending with **".csv"** in the csv folder, specified path, extracts <u>tMax,tMin,Air frost days and Rainfall</u> data, according to the provided year(or all years) and station's name. Then it reads the file line by line, splits them into columns, separating by "," character, the lines are essentially arrays of Strings, which are indexed and populate List type class variables. Method then converts the data to array of doubles, stores it in designated Class Variables, to allow mathematical calculations to be performed.

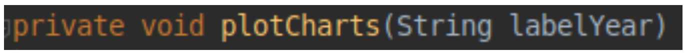


```
private void Calculations()
```

For Requirement (Question) 1 ("Last Year Stats and Report" Tab) calculates and assigns data to class variables, the variables are then used when generating a **.txt** or **.csv** report, also this method sets the **Labels**, which display Station's highest tmax, lowest tmin, total air frost days, total rainfall. Method loops through the converted double arrays, (populated with data from csvReader()) and by comparing indexed values determines **highest tMax and lowest tMin**, also by summing-up values, determines **Total Rainfall and Air Frost Days.** Example:

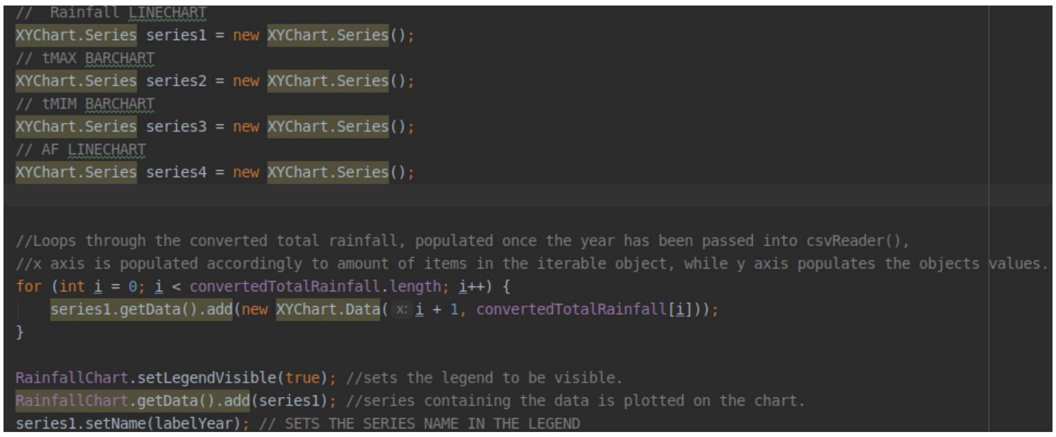


```
//CHECKS IF THE CSV FILE IS NOT EMPTY

if (!inputMax.isEmpty() & !inputMin.isEmpty() & !inputAF.isEmpty() & !inputRainfall.isEmpty()) {

    // HIGHEST tMAX:
    this.theMax = this.convertedMax[0];
    for (int i = 0; i < this.convertedMax.length; i++) {
        if (this.convertedMax[i] > this.theMax) {
            this.theMax = this.convertedMax[i];
        }
    }
    tMax.setText(String.valueOf(this.theMax)); //sets the figure label for "HIGHEST MONTHLY MEAN MAXIMUM TEMPERATURE (tMAX)"
```

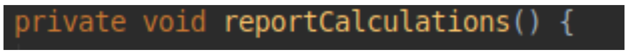


```
private void plotCharts(String labelYear)
```

For Requirement (Question) 2 ("Year Comparison" Tab) plotCharts() - takes year data either from "**Meteorological Station**", "**Selected year**" or "**Compare to Year(s)**" Event handlers. Based on the passed in **year**, plotCharts() Creates Total Rainfall, tMax, tMin, Air frost Days, Charts. Initial empty charts are created in ScieneBuilder as FXML and injected in Controller.java variables. Example below: (See source "Controller.java" file for further details)

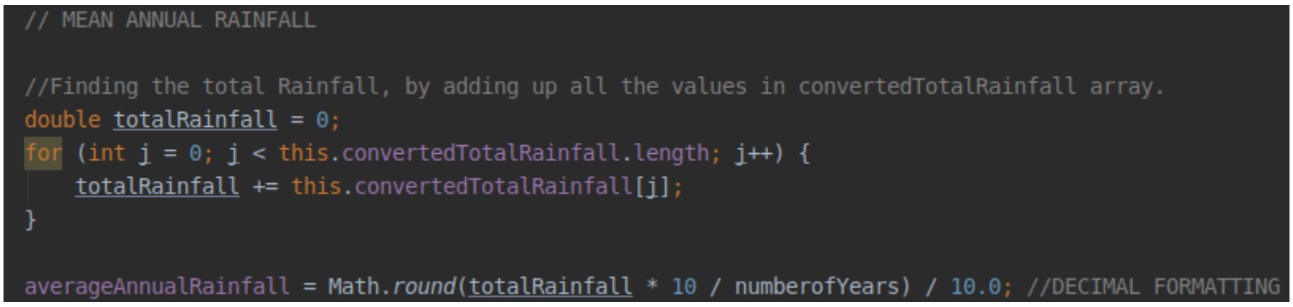


```
//  Rainfall LINECHART
XYChart.Series series1 = new XYChart.Series();
// tMAX BARCHART
XYChart.Series series2 = new XYChart.Series();
// tMIM BARCHART
XYChart.Series series3 = new XYChart.Series();
// AF LINECHART
XYChart.Series series4 = new XYChart.Series();


//Loops through the converted total rainfall, populated once the year has been passed into csvReader(),
//x axis is populated accordingly to amount of items in the iterable object, while y axis populates the objects values.
for (int i = 0; i < convertedTotalRainfall.length; i++) {
    series1.getData().add(new XYChart.Data( x: i + 1, convertedTotalRainfall[i]));
}

RainfallChart.setLegendVisible(true); //sets the legend to be visible.
RainfallChart.getData().add(series1); //series containing the data is plotted on the chart.
series1.setName(labelYear); // SETS THE SERIES NAME IN THE LEGEND
```

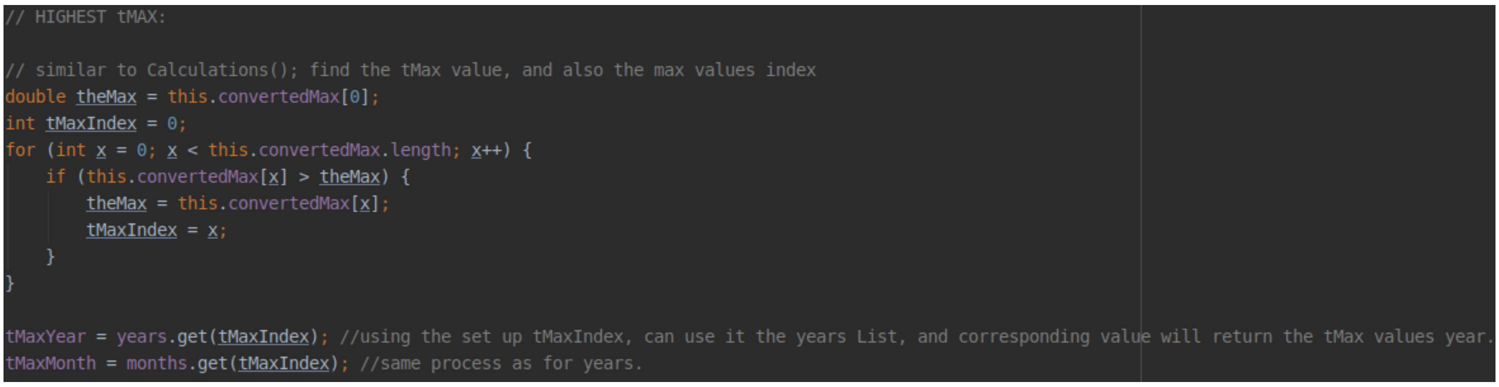


```
private void reportCalculations() {
```

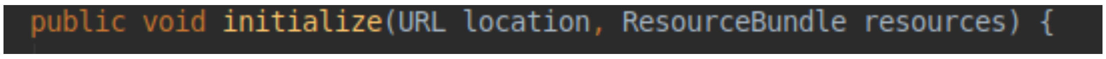
For Requirement (Question) 3 ("Station Comparison and Report" Tab) calculates for each station's **tMax's & tMin's year/month**, also calculates in **weighted averages for Air Frost Days and Rainfall**. Method assigns this data to dedicated class variables, from which the data is used when writing the **report into .csv or .txt.**
Examples below: (See source "Controller.java" file for further details)



```
// MEAN ANNUAL RAINFALL

//Finding the total Rainfall, by adding up all the values in convertedTotalRainfall array.
double totalRainfall = 0;
for (int j = 0; j < this.convertedTotalRainfall.length; j++) {
    totalRainfall += this.convertedTotalRainfall[j];
}

averageAnnualRainfall = Math.round(totalRainfall * 10 / numberofYears) / 10.0; //DECIMAL FORMATTING
```

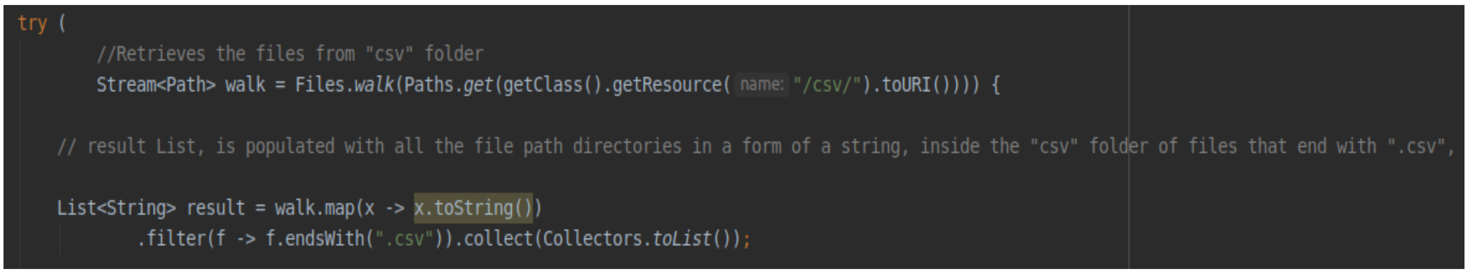


```
// HIGHEST tMAX:

// similar to Calculations(); find the tMax value, and also the max values index
double theMax = this.convertedMax[0];
int tMaxIndex = 0;
for (int x = 0; x < this.convertedMax.length; x++) {
    if (this.convertedMax[x] > theMax) {
        theMax = this.convertedMax[x];
        tMaxIndex = x;
    }
}

tMaxYear = years.get(tMaxIndex); //using the set up tMaxIndex, can use it the years List, and corresponding value will return the tMax values year.
tMaxMonth = months.get(tMaxIndex); //same process as for years.
```



```
public void initialize(URL location, ResourceBundle resources) {
```

The **initialize()** method runs once the application is started. The algorithm accesses all the **csv** folder files and uses **Files.walk** to capture all their file path directories as Strings:



```
try (
        //Retrieves the files from "csv" folder
        Stream<Path> walk = Files.walk(Paths.get(getClass().getResource( name: "/csv/").toURI()))) {

    // result List, is populated with all the file path directories in a form of a string, inside the "csv" folder of files that end with ".csv",

    List<String> result = walk.map(x -> x.toString())
            .filter(f -> f.endsWith(".csv")).collect(Collectors.toList());
```

Then stores them in a list, extracts station names into a **csvNames** String list, loops through it, and each **station** becomes a MenuItem object with event handlers, populated in **Station Selection** dropdown menus. The defined calculation, chart plotting and File Writer functions for report creating are all called via Event Handlers **(setOnAction)**, which are also implemented on **Year Selection** Dropdown MenuItems, also on **Report** and **Clear Data** buttons, along with **label hover Mouse (Tooltip) events.**

References:

mkyong, 2018. *Java Files.Walk Examples*. [online] Mkyong.com. Available at: <https://mkyong.com/java/java-files-walk-examples/> [Accessed 5 April 2020].

Abela, A. (2006). Visualise data. [online] Better Evaluation. Available at: https://www.betterevaluation.org/en/rainbow_framework/describe/visualise_data [Accessed 24 Feb. 2020].

ItachiUchiha, 2014. *Tooltip On Line Chart Showing Date*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/25892695/tooltip-on-line-chart-showing-date> [Accessed 5 April 2020].