**SHEFFIELD HALLAM UNIVERSITY**

Faculty of Arts, Computing, Engineering and Sciences

Student Name: **PHINA ONYINYE ANYALECHI** Student Number: **26006046**

Course Title: **MSC. COMPUTER AND NETWORK ENGINEERING**

Module Title: **OBJECT ORIENTED METHODS**

Tutor: **ALEX SHENFIELD** Module Code: **16-7213**

Assessment Title: **OBJECT ORIENTED DATA ANALYSIS IN C#**

Assessment ID: **1CWK50** Date Submitted: **MARCH, 2016**

**Assignment Brief**

To write an object oriented application in C# that can read a formatted data file and analyse the data contained within. Provided are some example data files containing a month-by-month historical record of the weather file from different locations around the country. I am expected to;

- Design a UML activity design showing the program flow for my application
- Write a C# program that can open any of the provided weather data files and interactively visualise and analyse the data
- Provide well defined test cases and results of the testing procedure.

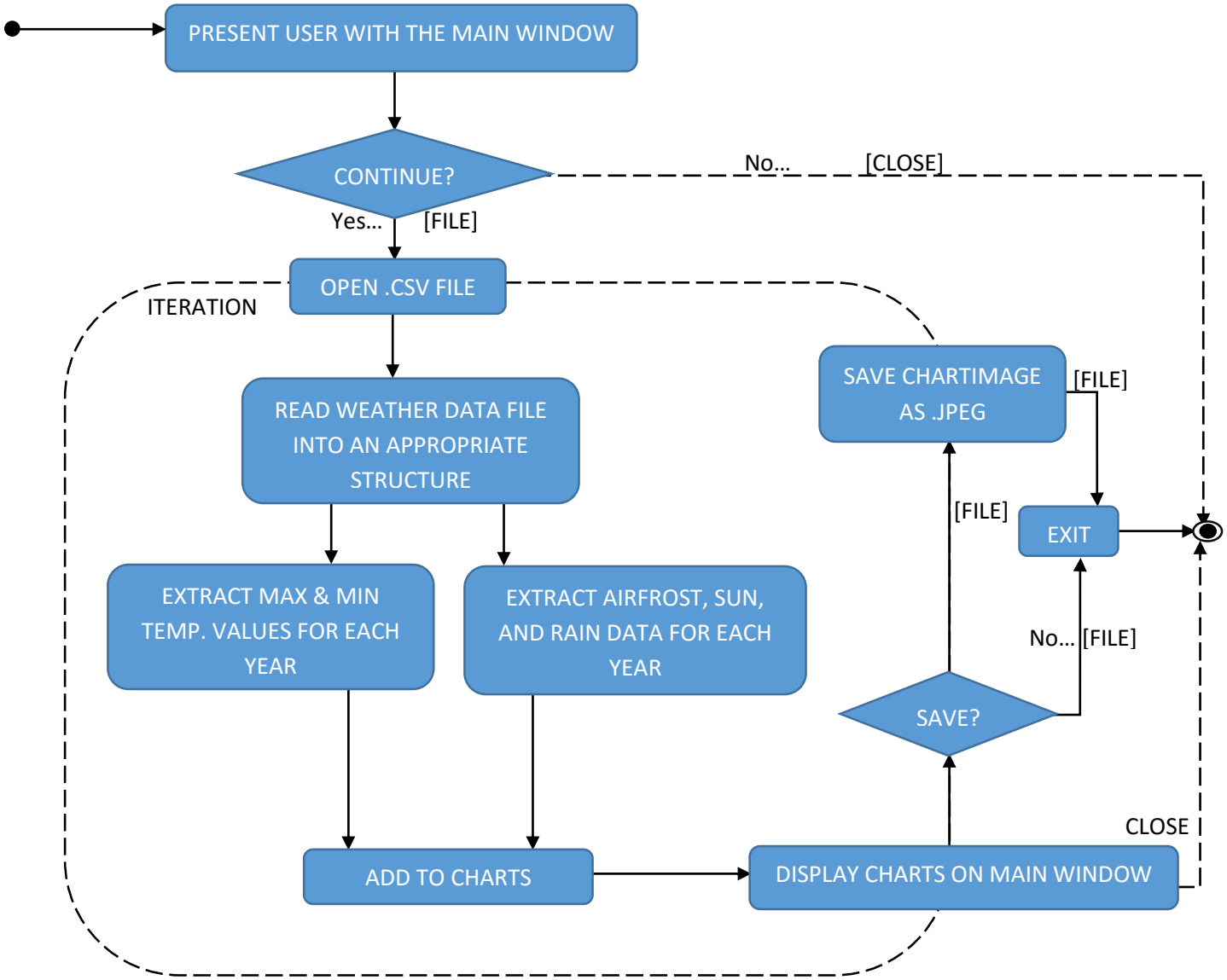



**FIG. 1: UML ACTIVITY DIAGRAM 1**

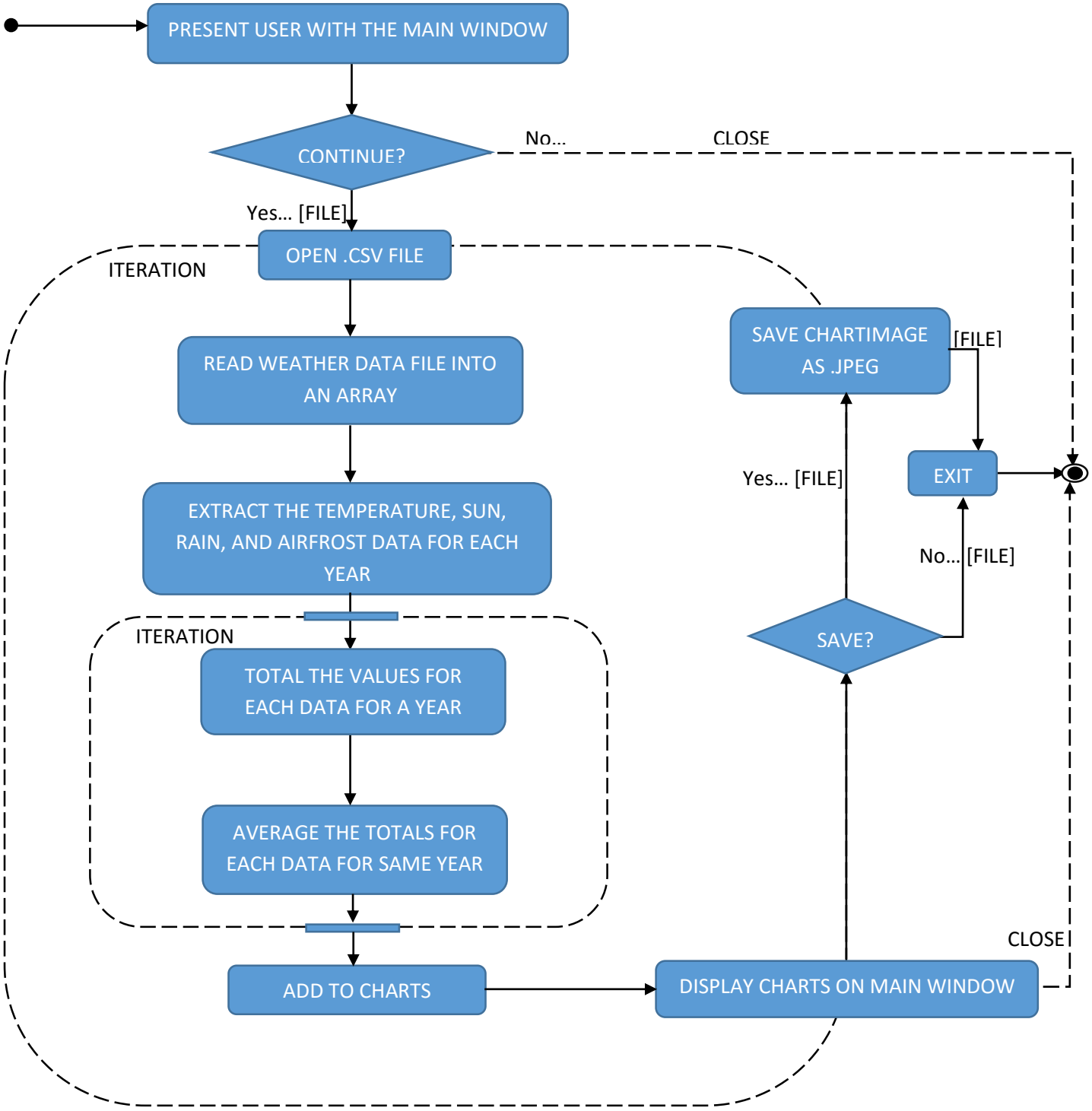



**FIG. 2: UML ACTIVITY DIAGRAM 2**

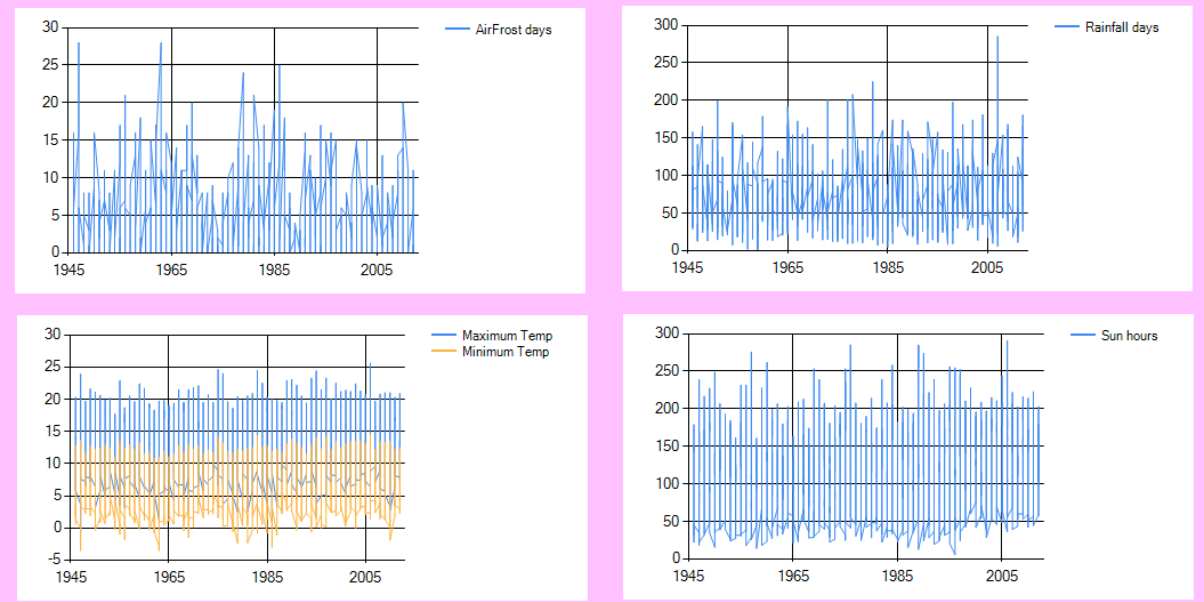
# TEST CASES

## CASE 1

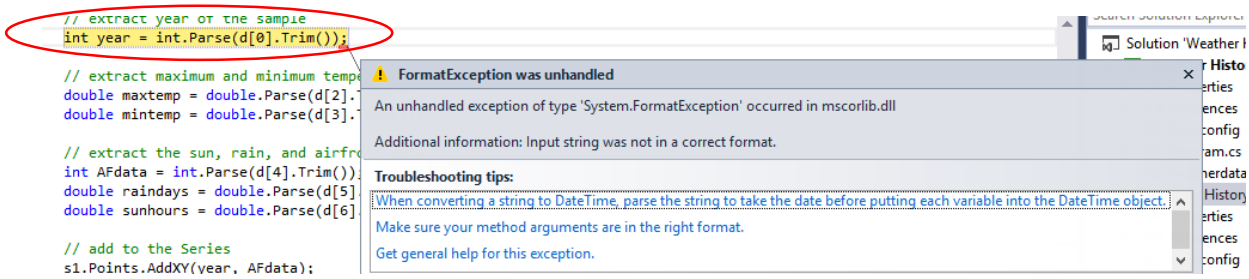
Figure 1 shows a UML activity diagram for WeatherHistory2.cs program. In this program, the chart images should display the weather samples by month, for every year. Below is a table (table 1a) showing the functionality of different weather files and the results for each test performed.

| S/N | TEST CASE | RESULT |
|---|---|---|
| 1. | When the formatted sheffieddata.csv file is opened and plotted | Perfect display of sample data on the charts (picture 1). No error messages. |
| 2. | When the unformatted sheffieddata.csv file is opened and plotted | No display of chart. Error message is shown (picture 1a) |
| 3. | When the formatted cambornedata.csv file is opened and plotted | Perfect display of sample data on the charts (picture 2). No error messages. |
| 4. | When the unformatted cambornedata.csv file is opened and plotted | No display of chart. Error message is shown (picture 2a) |
| 5. | When the formatted eastbournedata.csv file is opened and plotted | Perfect display of sample data on the charts (picture 3). No error messages. |
| 6. | When the unformatted eastbournedata.csv file is opened and plotted | No display of chart. Error message is shown (picture 3a) |
| 7. | When the formatted wickairportdata.csv file is opened and plotted | Perfect display of sample data on the charts (picture 4). No error messages. |
| 8. | When the unformatted wickairportdata.csv file is opened and plotted | No display of chart. Error message is shown (picture 4a) |

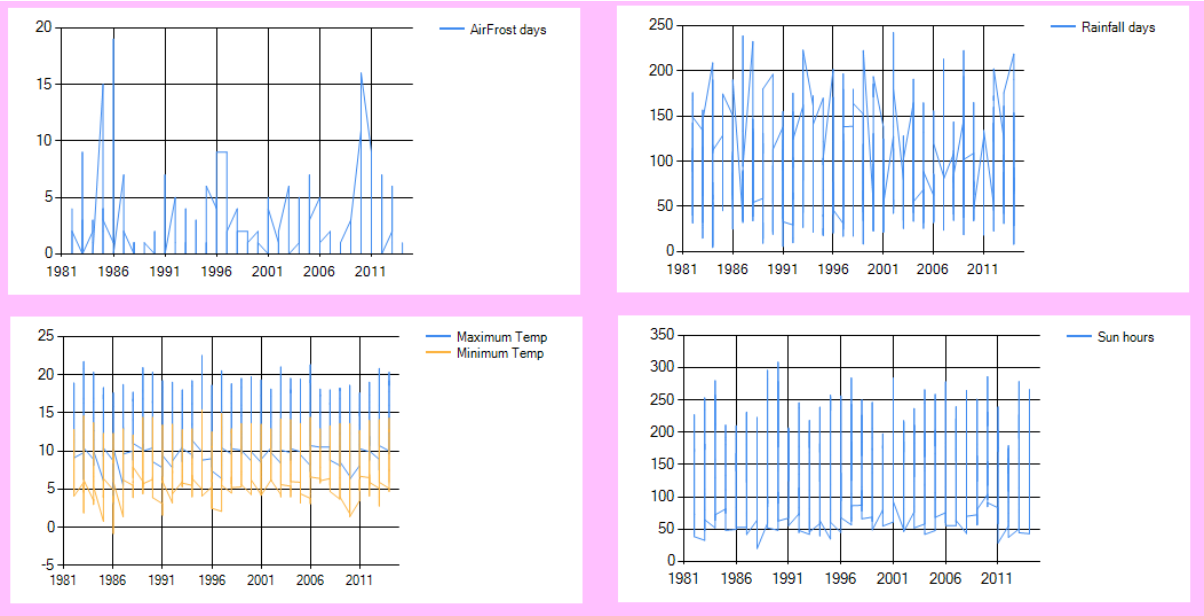
**TABLE 1A: FUNCTIONALITY TABLE 1**



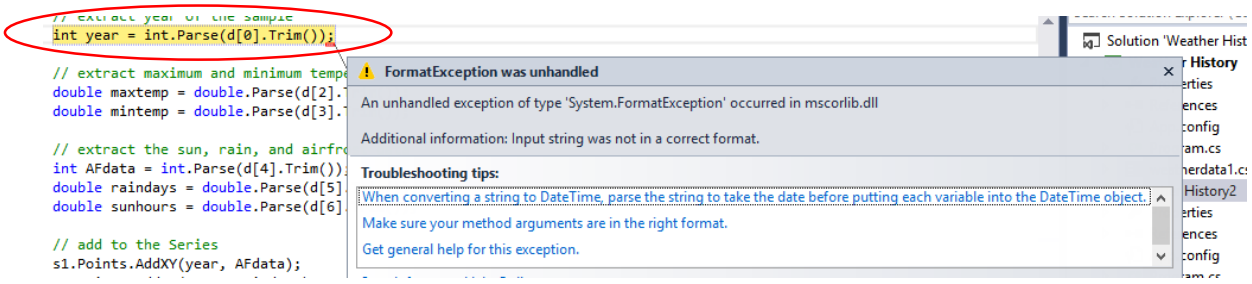

Picture 1: Sheffield yearly weather report



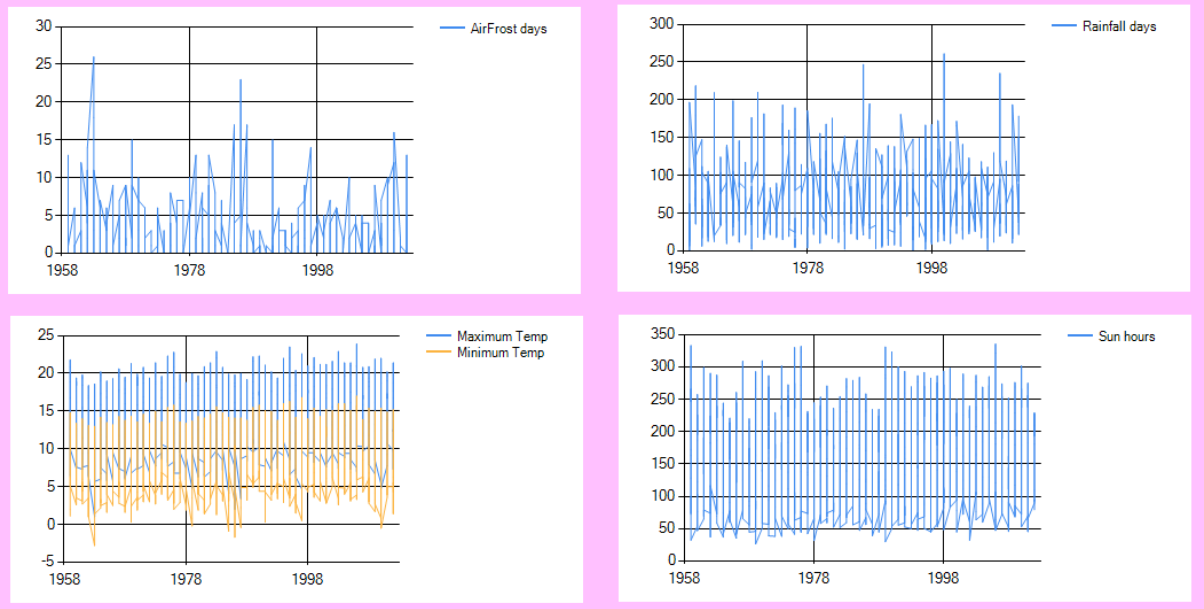

Picture 1a: Error message from unformatted Sheffield file



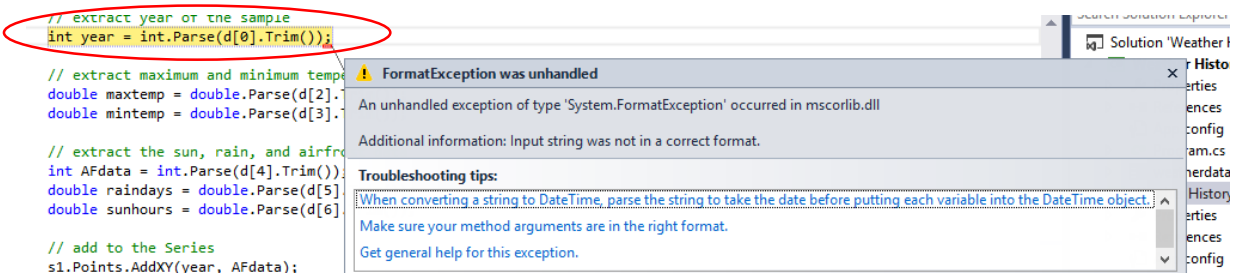

Picture 2: Camborne yearly weather report



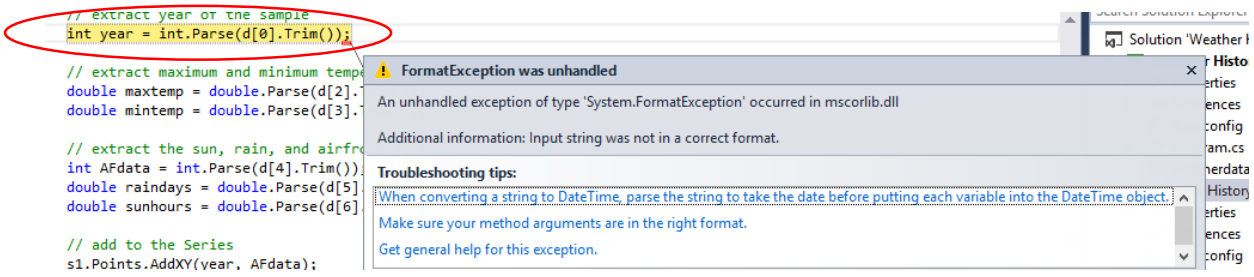

Picture 2a: Error message from unformatted Camborne file



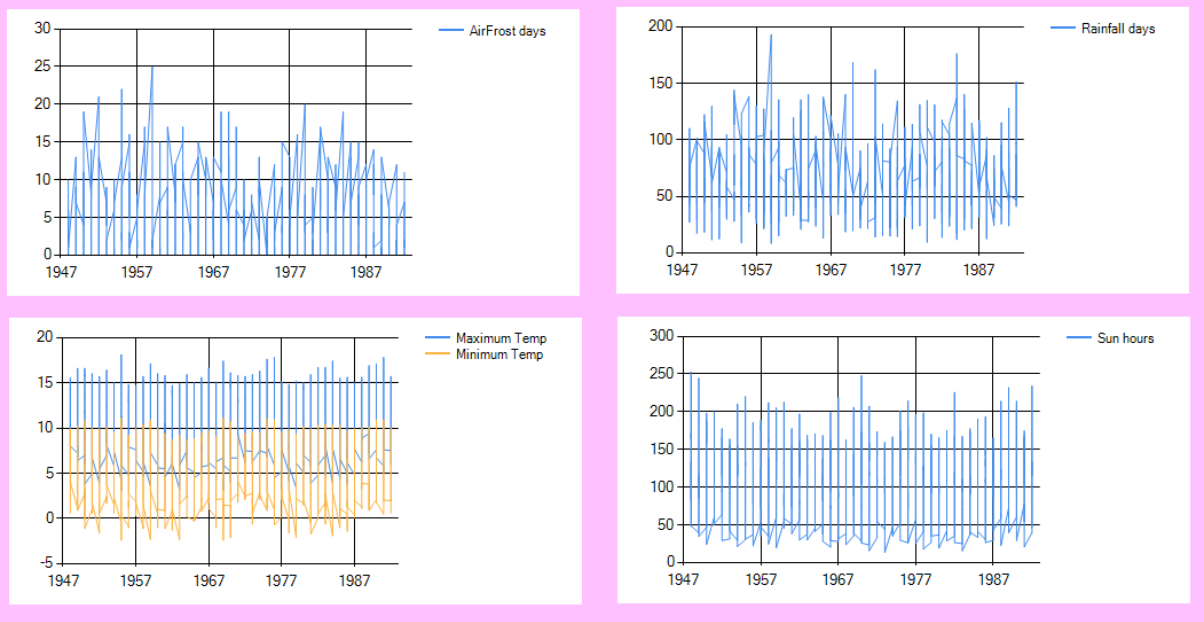

Picture 3: Eastbourne yearly weather report




Picture 3a: Error message from unformatted Eastbourne file




Picture 4a: Error message from unformatted Wick airport file




Picture 4: Wick airport yearly weather report

## SUMMARY

These results show that all unformatted files will not be properly debugged by the C# program. To be noted also, for each error message displayed, the cause of error was mentioned (circled in red), making it efficient for user to fix errors. For instance, all these error messages specified that the error was located in **d[0]**; which is the first column of our data file, it also means that it recognises only the first error.

Whereas, properly formatted files do not pose a problem. Will debug properly and give expected results.

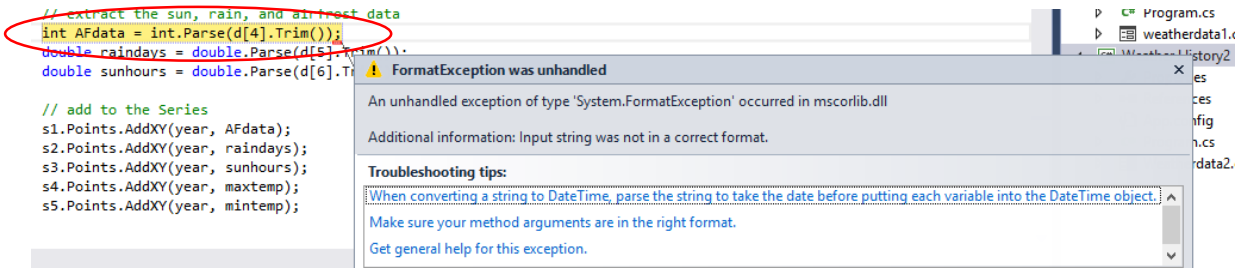
## CASE 2

Below is another table (table 1b) showing cases where the data files were not properly formatted and the outcome for each test.

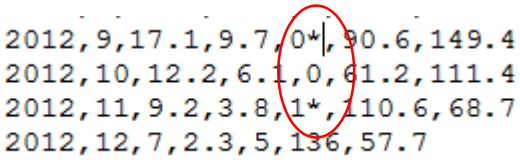
| S/N | TEST CASE | RESULT |
|---|---|---|
| 1. | When the sheffielddata.csv is opened and plotted | No display of chart. Error message shown (picture 5a). Cause of error shown (picture 6a). |

| | | |
|---|---|---|
| 2. | When the cambornedata.csv is opened and plotted | No display of chart. Error message shown (picture 5b). Cause of error shown (picture 6b). |
| 3. | When the eastbournedata.csv is opened and plotted | No display of chart. Error message shown (picture 5c). Cause of error shown (picture 6c). |
| 4. | When the wickairportdata.csv is opened and plotted | No display of chart. Error message shown (picture 5d). Cause of error shown (picture 6d). |

**TABLE 1B: FUNCTIONALITY TABLE 2**

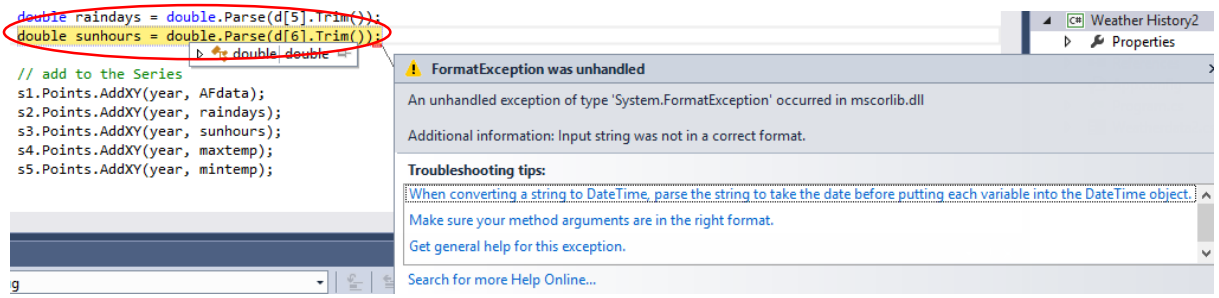



Picture 5a: Error message from partially formatted Sheffield file




Picture 6a: Cause of error located in Sheffield file

In the above case, the error message (picture 5a) showed that the error was located in **d[4]**; which is the fifth column and has the airfrost data. That is verified in picture 6a; a screenshot of the partially formatted area, showing the * symbol that caused the error.

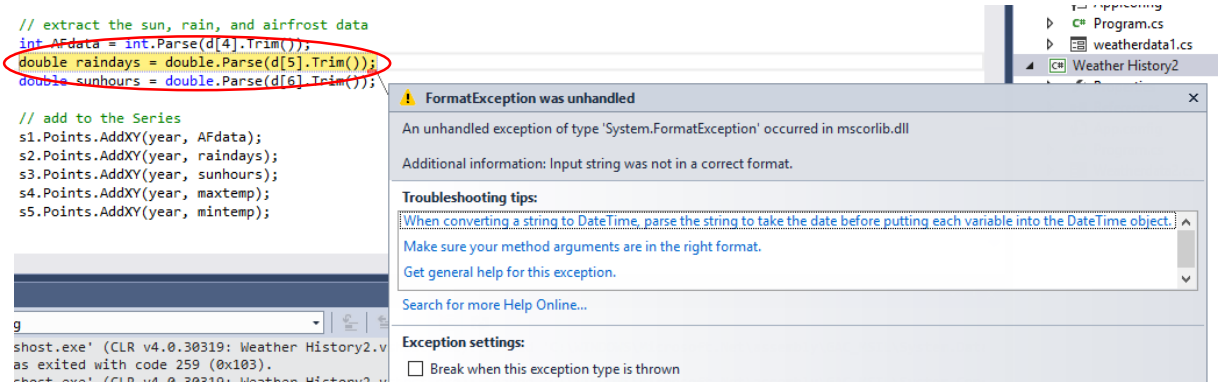



Picture 5b: Error message from partially formatted Camborne file

```
yyyy,mm,tmax,tmin,af,rain,sun,
1978,9,17.5,11.3,0,26.7,---,
1978,10,15.6,10.7,0,20.4,---,
1978,11,12.6,7.6,0,56.3,---,
1978,12,9.2,5,5,276.7,---,
1979,1,6.5,0.9,13,134.8,---,
1979,2,6.7,1.9,5,133,---,
1979,3,8.8,3.6,2,143.8,105,
```

Picture 6b: Cause of error located in Camborne file

In this case, the error message (picture 5b) showed that the error was located in **d[6]**; which is the seventh column and has the sun data. That is verified also in picture 6b; a screenshot of the partially formatted area, showing the --- symbol that caused the error.

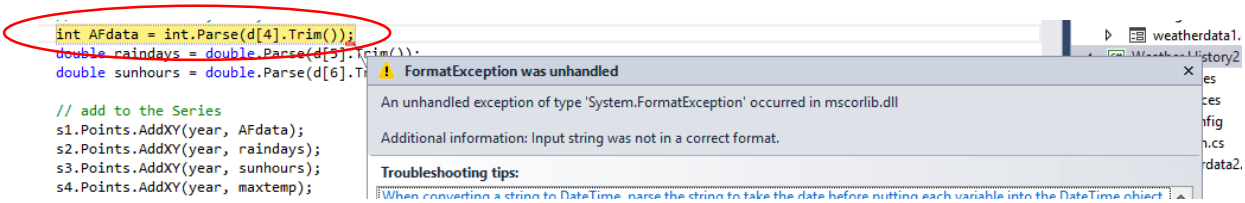



Picture 5c: Error message from partially formatted Eastbourne file

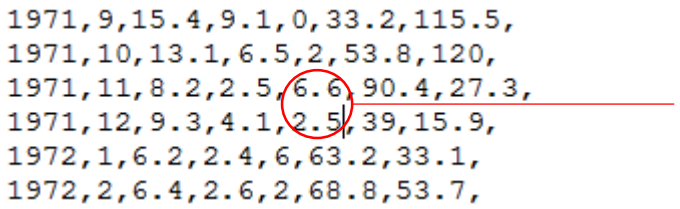
```
1970,9,18.4,12.4,0,113.8,195.3,
1970,10,15.1,9.6,0,28.1,128.5,
1970,11,12.1,7,1,209.9,68.1,
1970,12,7.3,3.2,7,58.1*,58,
1971,1,7.8,3.9,6,89.9,56.7,
1971,2,8.3,3.4,0,25.5,93.5,
```

Picture 6c: Cause of error located in Eastbourne file

In this case, the error message (picture 5c) showed that the error was located in **d[5]**; which is the sixth column and has the rain data. That is verified also in picture 6c; a screenshot of the partially formatted area, showing the * symbol that caused the error.




Picture 5d: Error message from partially formatted Wickairport file




Picture 6d: Cause of error located in Wickairport file

In the above case, the error message (picture 5d) showed that the error was located in **d[4]**; which is the fifth column and has the airfrost data. That is also verified in picture 6d; a screenshot of the partially formatted area, showing the wrong format of values which caused the error.

## CASE 3

During the functionality testing with the partially formatted files, I also tested, varying the location of the error.

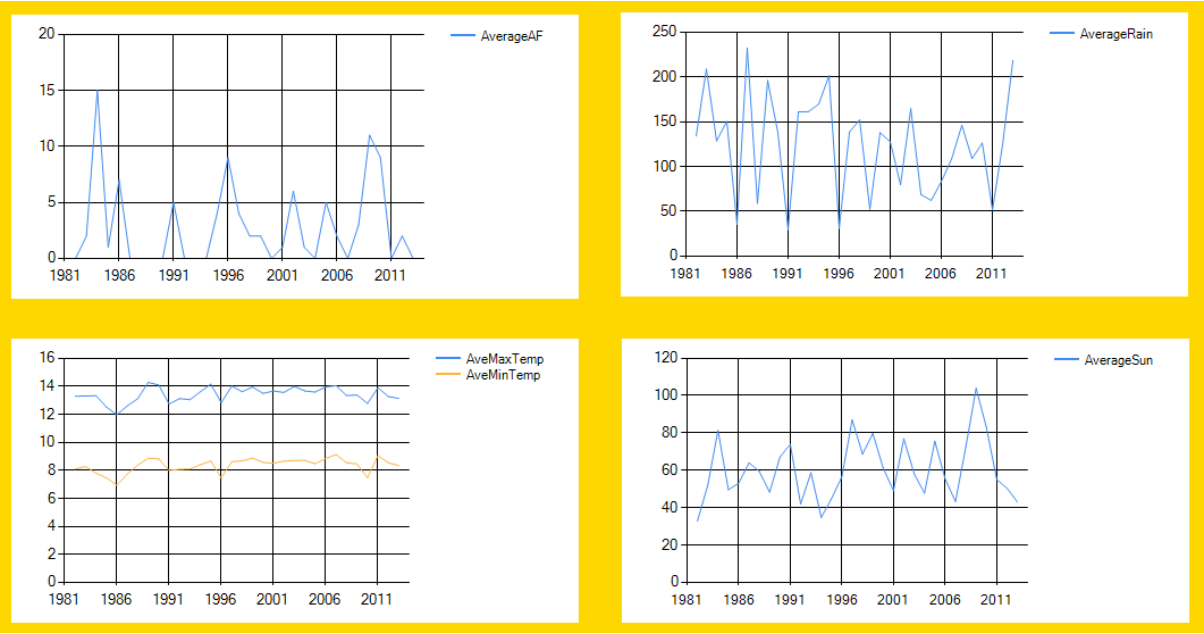
From picture 6a; for sheffield, the error was present towards the end of the data file, previous values were neglected. It took a while, but an error message was eventually diplayed.

From picture 6b; for camborne, the error was detected at the start of the data file, and an error message was promptly displayed.

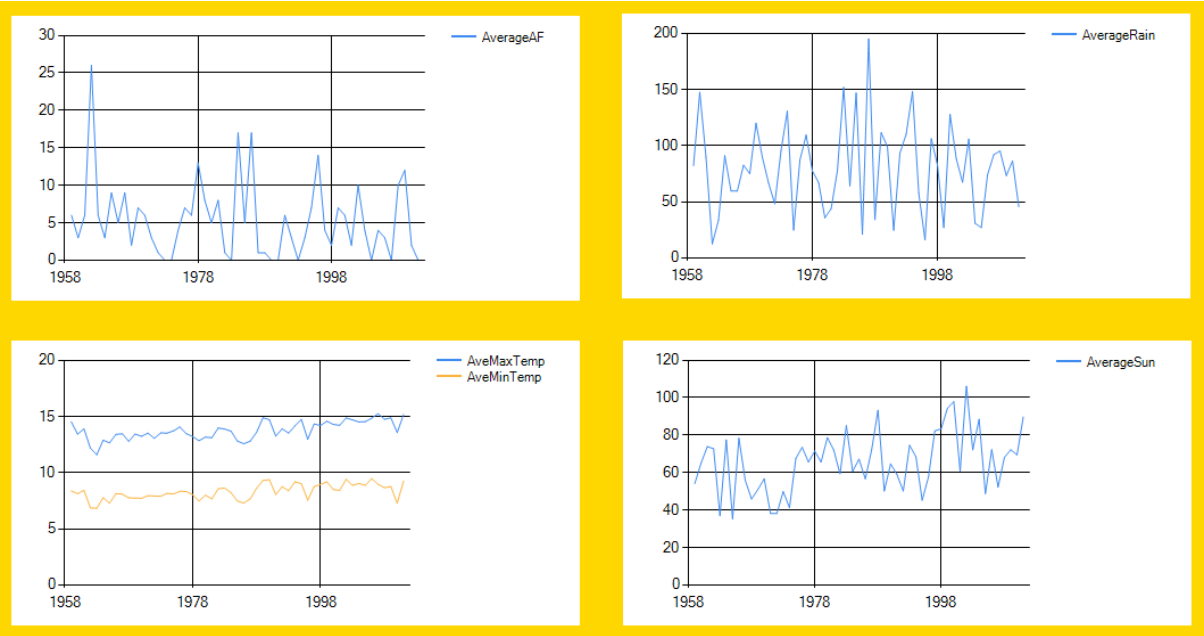
From picture 6c & 6d; for Eastbourne and Wickairport respectively, the error was positioned in the middle of the data file. It took a while also, but eventually an error message was displayed.

## CASE 4

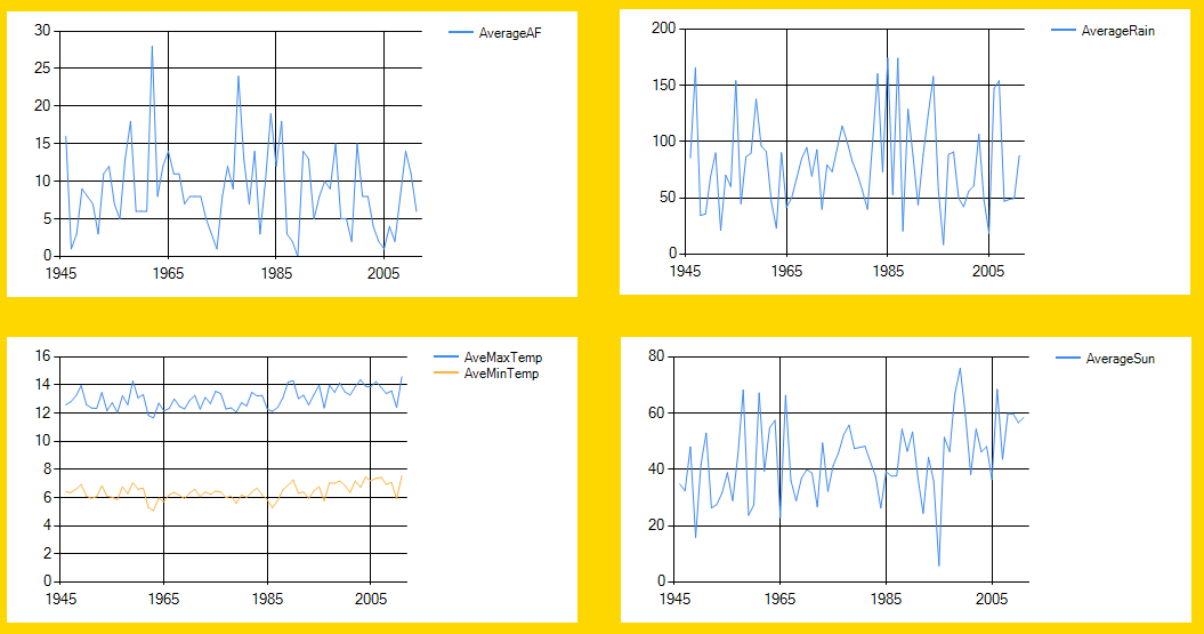
Figure 2 shows a UML activity diagram for WeatherHistory.cs program. In this program, the chart images should display the average values for the weather samples by month, for every year. The display is a trending image for each sample in a year. Tests were carried out for the formatted files and partially formatted files only. Below are the results.



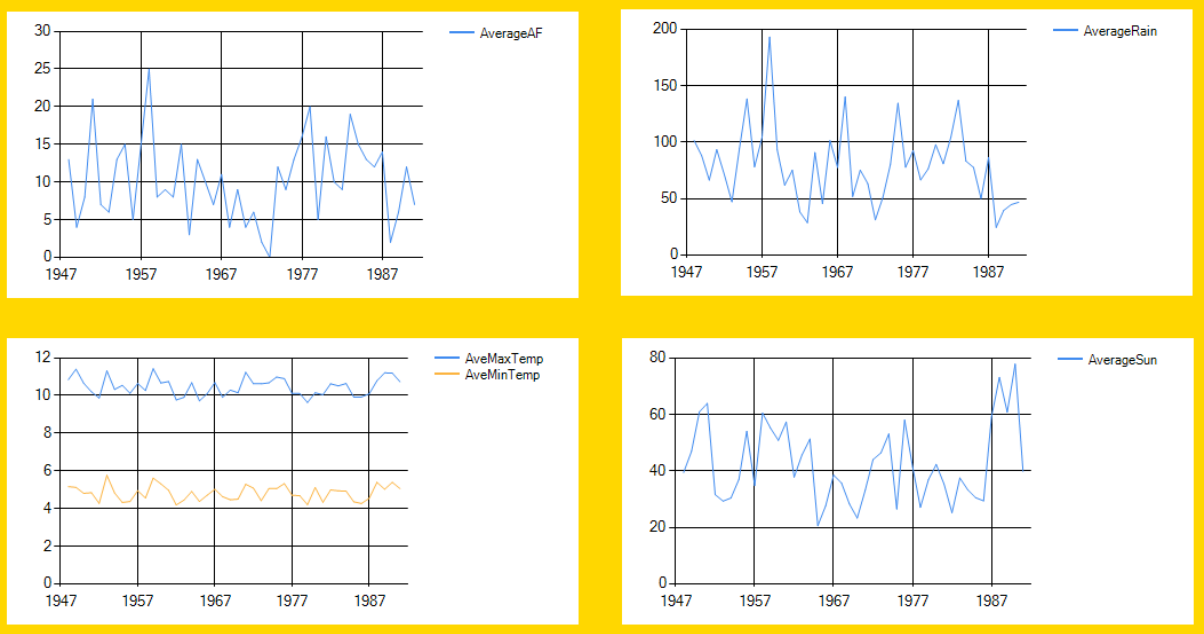

Picture 7a: The average weather report for Camborne




Picture 7b: the average weather report for Eastbourne




Picture 7c: The average weather report for Sheffield




Picture 7d: The average weather report for Wickairport

## SUMMARY (OUTCOME OF TESTS IN CASE 4)

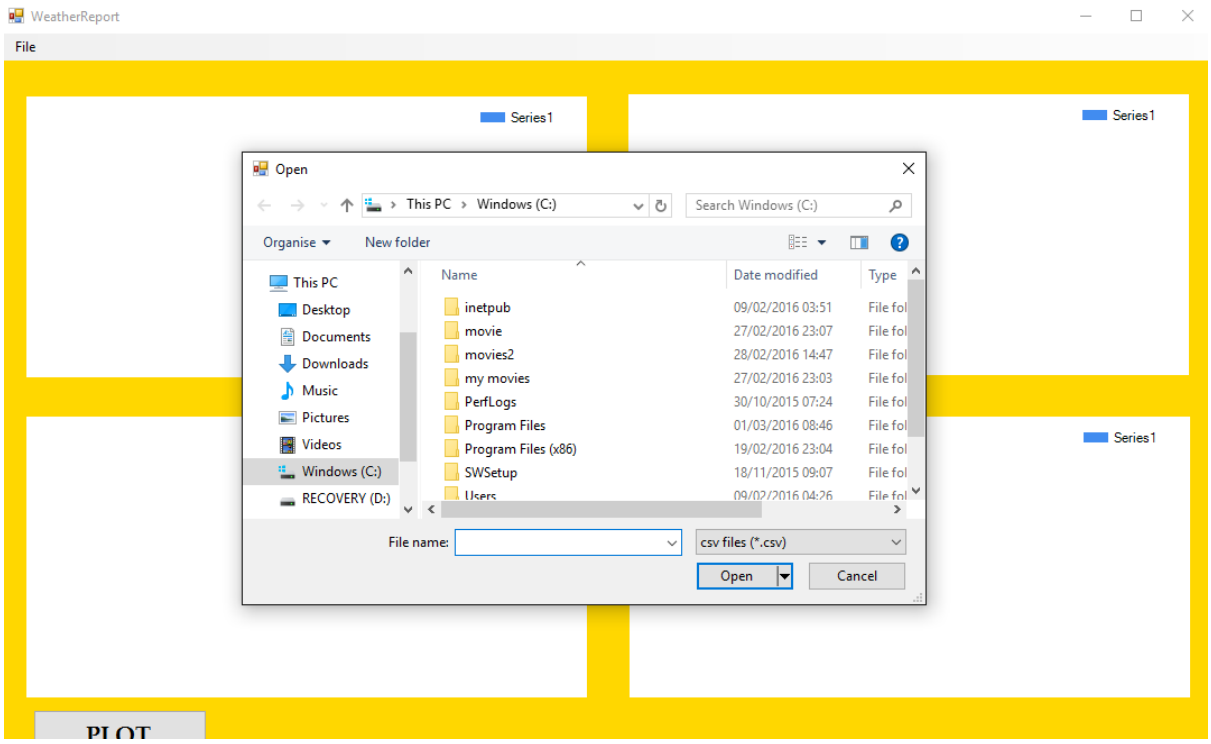
The above pictures show that for the formatted files, the expected outputs were gotten. Whereas, for the partially formatted files, the results were exactly the same as in CASE 2 above.

## CASE 5

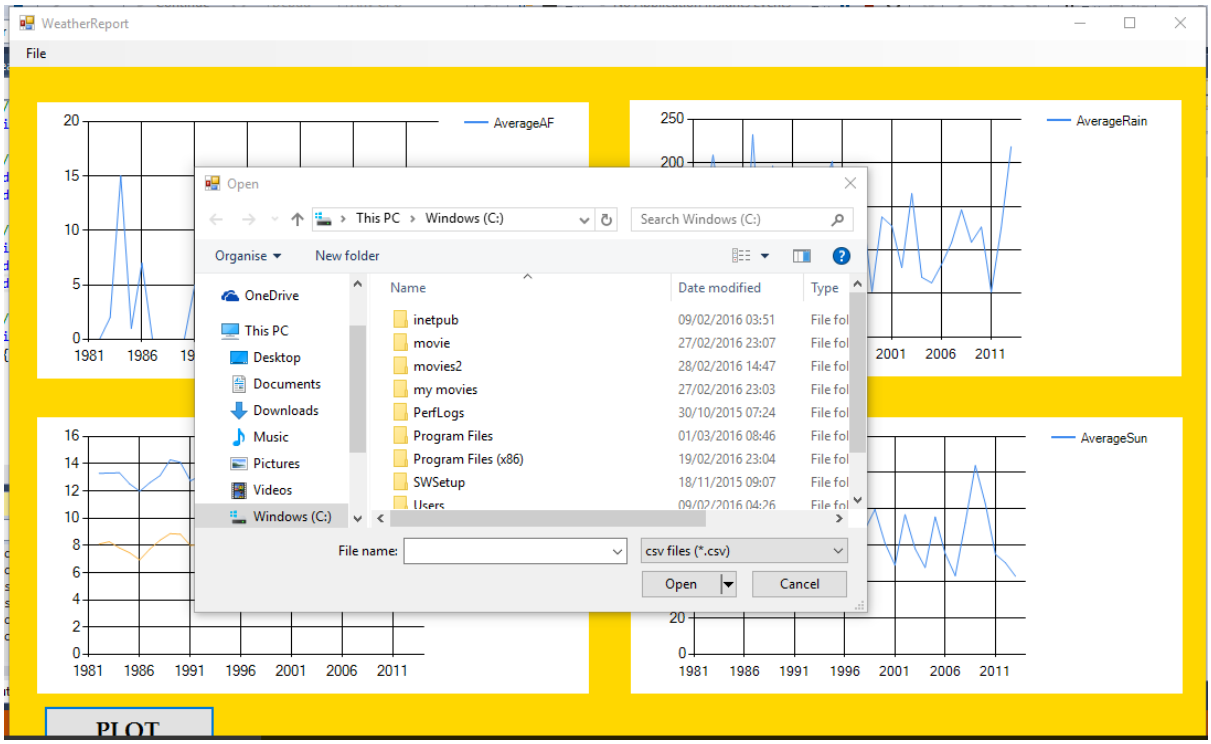
I also tested for the functionality of the written code;

a. To open any desired .csv file and display on chart; worked perfectly.
b. While code is still debugging and having displayed desired chart, user can open another file and display its chart without ending the debug process (for formatted files only). This explains the iteration symbol shown in the UML activity diagram. This worked perfectly also.
c. To save chart image as a .jpeg file for later reference, if user desires. The result shows that it works best for single chart windows, because it was able to save only one of the four charts.

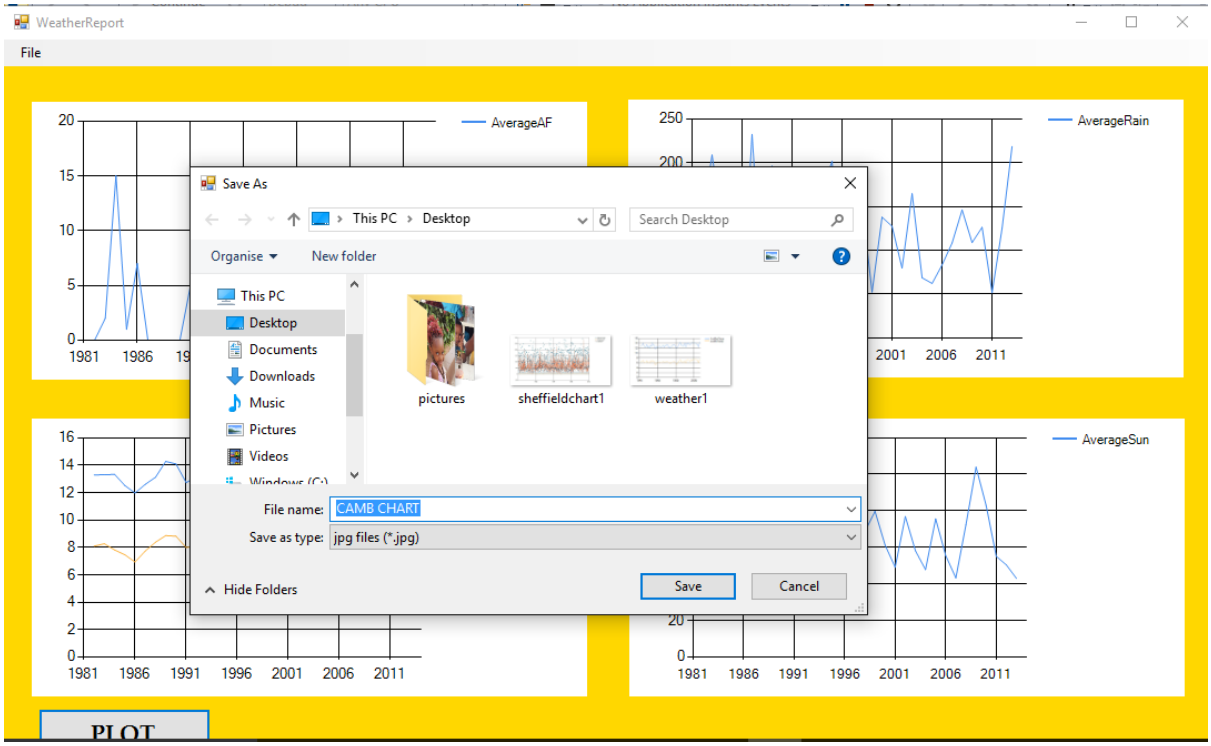
Result images are shown below.



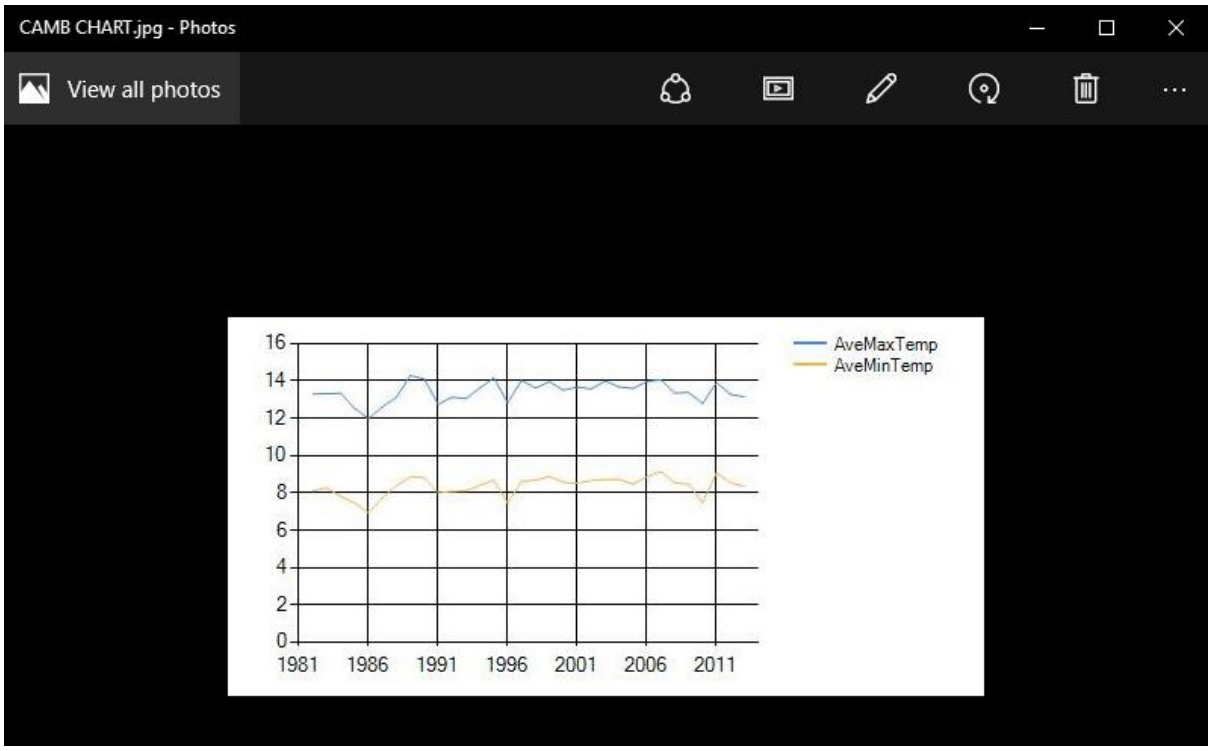

Picture 8a: Showing result for case 5a




Picture 8b: Showing result for case 5b




Picture 8c: Showing result for case 5c




Picture 8d: Showing saved image for case 5c

## CONCLUSION

Both programs worked satisfactorily; analysing each weather data provided and displaying it for user.