Overview

To complete the week 1 practical, I had to write a Java program which could process data from a file. For this practical, the data was stored in comma-separated- values (CSV) format. My program needed to be able to read in data from a given file, processes the data, create a new file containing the holder number, spectrometer, research group, solvent, and experiment name for each sample. The program also needed to generate some summary statics. I opted to complete all the extensions too.

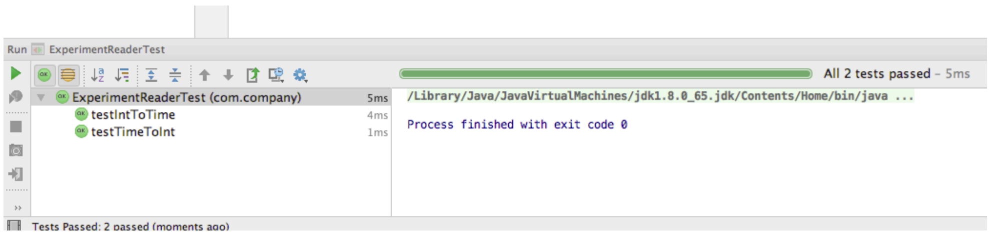
Design

I have used multiple classes and methods to create the solution to the given problem. The first class contains the main method, in this method a new instance of the class "experimentReader" is created and handed in the constant "INPUT_FILE_PATH" which is used when creating a new buffer reader. Next the print writer is created which makes a new text file, where the data is output to. To ensure that all the lines of the CSV file are read I have created a while loop which uses a method from the experimentReader object to check for new lines. A subclass of the experimentReader class, experiment is then created. The experiment class only has attributes in it which are used to store the data for each sample. Other methods are then used to write the correct data to the output file. So that the program can out a HTML document too, I have added appropriate parameters to the methods creating the file so that the correct HTML tags can be written to the correct file. The experimentReader class is used to primary to handle the input file and withdraw the correct data. The first method is used checks if the current experiment object is equal to null, which is should be until the program reaches the end of the file. The program then checks to see if there is a next line in the CSV, if there is the values are stored in an array and the then the attributes of the experiment object are set to the correct values in the array. The program will also skip the first line of the CSV file because the values stored in the first line are the column names and not actual data. There are also multiple methods which allow the program to calculate statistical data which is saved at the end of the output file. To ensure the program can deal with the various exceptions that could occur during execution, I have added numerous try and catch blocks which will allow the program to attempt to execute code but in the event of the code throwing an exception, the program can still continue. The over all solution to the problem is relatively simple but efficient. Due to the time constraints I was unable to fully utilise and understand the JavaFX library. However, I have attempted the 3rd extension, which was to have the program generate some sort of graphical representation of the data. To do this I had to re-write a different program and change the way in which data is both read in and dealt with, these alterations have resulted in the program being some what slower than the original program.

Testing

To test my program and make sure it works as expected I will create create and ExperimentReader object and numerous Experiment objects. I continue by making sure each method from every class is called and works correctly. Because there are not many instances where the user needs to input data into the program, I will not have to do to many verification and validation tests, so for example entering a number instead of a string for a user name. Moreover, I have also incorporated many try and catch error statements which allow my program to catch many other errors. However, I will, where possible, enter invalid data in an attempt to break the program and therefore preform both extreme and erroneous tests. If the program works as expected, the program will be able to deal with the tests without crashing or returning inaccurate data. Testing the mathematical and logical operands in the program is also very important to ensure the data is being correctly manipulated. Finally, although this wont effect the program itself, I will make sure the text output is formatted in way which is suitable for the user.

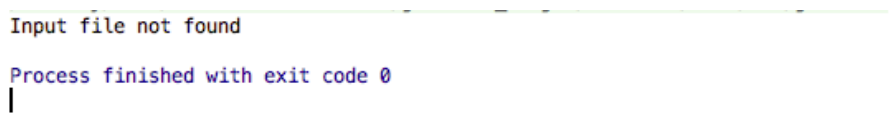
Examples



```
Run  ExperimentReaderTest
All 2 tests passed - 5ms
ExperimentReaderTest (com.company)   5ms
    testIntToTime                    4ms
    testTimeToInt                    1ms

/Library/Java/JavaVirtualMachines/jdk1.8.0_65.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

Tests Passed: 2 passed (moments ago)
```

I have used Junit to test some of the universal functions in my program. I have also compared the data that is output to the text and html file with that in the CSV. By using Microsoft Excel I was able to successfully find the shortest and longest recorded durations, the number of samples that were run on each spectrometer, the percentage of samples that were tested using the experiment "proton.a.and" and also the total duration of all tests. The results match those that are calculated when the program is run.

Next I changed the program to try and read in a file that did not exist, the program correctly recognised the error and displayed the right error message.



```
Input file not found

Process finished with exit code 0
```

Evaluation

My program successfully meets all the initial requirements for the program and moreover I have also added extensions in. The core functionality of my program successfully works and is also able to with erroneous and extreme tests without crashing but instead recognising the errors. The program which displays the graph is not as efficient as I would like it to be, as it reads in the data file twice in order to display the graph. However, this is only a minor problem and does not effect the functionality of the program, but instead increase the execution time. When using small data sets this is unnoticeable.

Conclusion

In conclusion I'm happy with end program(s). They complete the task set and have a verity of added functionality which make the program more flexible and user friendly. The extension however was difficult to program and given more time I would like to have improved the graph functionality. The use of object orientation well as splitting the code into numerous functions which are able to provide multiple functionality, makes my code easy to read and follow and moreover there is no repeated code.