—Assignment #4—

Make sure to review the submission requirements on pg. 4 of the Lab #1 document.

Follow the Style Guide as given in the Resources section of our conneX course page.

Learning Outcomes

When you have completed this assignment, you should understand:

- How to use procedural decomposition to break down a problem into methods.
- How to use `while` loops to process variable length input.
- How to use the `File` class to read from a file.
- How to use `Scanner` to process file input.
- How to use arrays to process data.
- How to develop your own strategy to test code for correctness.
- Gain insights into your own studying habits.

You will complete a program to process your study log and learn more about yourself as a student. Download the template file `StudyStats.java` from Assignment #4 on conneX. Write the missing code for the methods in this template and as usual, compile and test your code often as you write it up.

Make sure to back up your `study_log.txt` file while working on it, in case you accidentally erase the contents of the file. You can test your code on an a sample study log, `study_log_check.txt` also available for download from Assignment #4 on conneX.

A sample run for the completed program on the check file outputs the following:

```
jan 8 1300 1420 lecture uvic
jan 9 1330 1520 lab uvic
jan 11 1100 1245 selfcheck cschelpcentre
jan 11 1300 1420 lecture uvic
jan 11 1430 1530 help office
jan 12 1000 1100 selfcheck desk
jan 12 1400 1510 assignment1 desk
jan 13 1200 1720 assignment1 desk
jan 15 1040 1130 help cschelpcentre
jan 15 1300 1420 lecture uvic
jan 15 1440 1500 help office
jan 15 1700 2100 assignment1 desk
jan 16 1530 1720 lab uvic
jan 18 0900 1100 assignment1 desk
jan 18 1300 1420 lecture uvic
jan 20 1000 1200 selfcheck desk
jan 23 1040 1300 selfcheck library
```

```
jan 23 1900 2200 assignment2 desk
jan 24 1000 1110 selfcheck library
jan 24 1400 1530 selfcheck library
jan 24 1600 1700 assignment2 library
jan 25 1300 1420 lecture uvic
jan 25 1600 2100 assignment2 desk
jan 26 1300 1830 assignment2 computer
jan 27 1600 1800 assignment2 computer
jan 28 1700 1900 assignment2 desk
jan 29 1300 1420 lecture uvic
jan 30 1330 1520 lab uvic
jan 31 1900 2100 selfcheck library

Your average study session in minutes is: 120.86206896551724
Your maximum time study session was:
jan 26 1300 1830 assignment2 computer
```

I recommend starting with simply understanding the code in the template as it is written before you begin to work, and become more familiar with the `start` and `stop` arrays. Walk through the code and convince yourself that it is storing the file entry data into the arrays. You can see the output printed as the arrays are filled with the helper method `printEntry`.

Then start with the `timeInMins` method, and make sure it calculates the number of minutes from midnight to the input parameter `time` correctly.

Then use `timeInMins` to help you write the next method `entryMins` (you can actually write this method in one statement). This method should return the total number of minutes in an entry, from its `start` to its `stop` time. You should not have any entries in your log with start time before midnight, and stop time after midnight on the morning of the next day. If you do, simply split the entry into two entries, one for each day.

Then use `entryMins` to help you write `aveStudyMins` (make sure to cast the total number of minutes as type `double` in your final calculation, before you divide). This method should calculate the average number of minutes across entries in your log. You can then see for yourself how much time are you spending on average when you decide to study.

You can also use `entryMins` to help you write `maxStudyTimeEntry` (see min/max loops on pg. 256 in your textbook). Note that you can also save the index of the maximum entry you find with an integer variable, not just the maximum value itself. You can then see what activity made you spend the most time in one study session.

There are many more things you could program yourself to find out about your study habits.

If you decide to write a method to test parts of your code, you could leave that extra code in your submission, but make sure to get rid of any calls to it before you submit. If you have an adequate testing method, then it can earn you up to 5 buffer marks (doesn't contribute to a mark above 30).

Grading

Marks will be allocated out of 30 points:

- (0 points) You do not submit any program file.
- (5 points) Your program does not compile. The marker cannot fix the complier error easily, or if they can, the output is severely mismatched with the assignment requirements.
- (10 points) Your program does not compile, but the marker can fix the error easily.
- (15 points) Your program compiles, but only outputs a few of the assignment requirements.
- (20 points) Your program compiles, outputs most of the assignment requirements, and is missing documentation comments or has formatting issues.
- (25 points) Your program compiles, outputs everything required, but is missing documentation comments or has formatting issues.
- (30 points) Your program compiles, outputs everything required, and has proper documentation and formatting.