CS1811 Assignment 1: Track and Trace

- Submission Date: Monday 9 November 2020, 16:00
- Feedback Date: Monday 30 November 2020

Learning Outcomes Assessed

This assignment tests your ability to write a complete Java program, using the control flow constructs described in the lectures. That is,

- sequencing,
- branching,
- loops,
- method calls.

You will write a program that interacts with the user via a Scanner and performs some simple calculations.

Goal

You will write a program `TrackAndTrace` to help restaurants, cafes, and pubs to control the spread of the coronavirus.

This is a good time to remind you to install the NHS Track and Trace app if you have not done so already!

Our application will not be as clever as the NHS app, but will implement the basic functionality. The user will first enter a time period where an infected person was known to be on the premises. A time period will be represented by two integers – a start and an end time. For example a start of 5 and end of 7 means the person was on the premises from 5 o' clock to 7 o' clock. We will only count whole hours for this exercise.

After entering this time period, the user then enters the names of all customers and the time they were on site. The program will print out if each customer needs testing, and summarise at the end how many tests are needed.

An example run is shown below, with an infectious person present from 5 to 7, and two customers. The first will need to have a test, the second will not. A total of 1 test is needed.

```
$ java TrackAndTrace
Enter the start time:
5
Enter the end time:
7
Enter the number of customers:
2
Enter the customer's name:
```

```
Sulaimaan
Enter the arrival time:
4
Enter the departure time:
8
Sulaimaan needs a test.
Enter the customer's name:
Dmani
Enter the arrival time:
8
Enter the departure time:
9
Dmani does not need a test.
There are 1 tests needed.
```

Instructions

Complete your work on NoMachine. Create a file `TrackAndTrace.java`. In this file you will write a Java program.

Submit your file `TestAndTrace.java` using the submission button below.

You can test your submission using the CS1811 Assignment 1 Testing Tool. Please download the tester file and save it to the same directory where you are working on your assignment. To use the tester, first compile your code, and then run the command below. This [walkthrough video][tester-download] may help if you are having difficulties.

```
$ javac TrackAndTrace.java
$ java -jar cs1811Assignment1Tester.jar
```

This will run a series of tests you can use to verify the correctness of you solution. If a test does not pass, an explanation will be provided. An example output is shown below.

```
Running testQ0_MainClass

  Test passed

Running testQ1_01OverlapMethod

  After calling

      boolean b0 = TrackAndTrace.overlap(1, 2, 3, 4);

  expected the result of TrackAndTrace.overlap(1, 2, 3, 4) to be
  false because the time periods don't overlap. Instead the call
  returned true.
```

```
Tests failed
```

In the above run, the first test passed, but the second test showed that there was a problem with the method `overlap` in the `TrackAndTrace` class. In particular, it returns the wrong answer with the arguments 1, 2, 3, 4. You can write a test class with a main method that runs the code shown to debug the problem.

You program will take input from the keyboard and print output. The tester will test this too and provide information about failing tests. For example,

```
Running testQ2_01GetOverlapsMethod

  Test failed!

  Failing interaction:
  > Enter the number of customers:
  < 1
  > Enter the customer's name:
  < Danny
  > Enter the arrival time:
  < 8
  > Enter the departure time:
  < 10
  > Danny does not need a test.

  Expected 'Danny needs a test.
```

In the above run, input is marked with `<` and output with `>`. Your program does not need to output these symbols. The test failed because the output was wrong. The program said "Danny does not need a test." when it should have said "Danny does need a test.".

The tester source code is included in the JAR file, which can be unzipped and inspected.

Marking Criteria

The marks awarded per question are shown below. When awarding marks, we take into account the functionality and your programming style. That is

- the code should produce the right output, and
- the code should follow good programming style.

Good programming style means formatting your code so that it is readable (make good use of indentation), using good variable names, and avoiding overly complex or incomplete solutions.

For example, to print the numbers from 1 to 10, the following code is badly formatted and repetitive.

```
System.out.println("1"); System.out.println("2");
        System.out.println(
        "3");
    System.out.println("4");
System.out.println("5"   );System.out.println( "6"
                        ); System.out.println("7");
System.out.println("8"); System.out.println("9");
    System.out.println("10");
```

A much cleaner solution is shorter and nicely indented.

```
for (int i = 1; i <= 10; i++)
    System.out.println(i);
```

All the work you submit should be solely your own work. Coursework submissions are routinely checked for this.

Questions

Question 0: Your Program Class [5 Marks]

Create a file `TrackAndTrace.java` and declare inside it a class `TrackAndTrace`.

After completing this question, compile your code with

```
$ javac TrackAndTrace.java
```

and then run the tester

```
$ java -jar cs1811Assignment1Tester.jar
```

You should see that `testQ1_00MainClass` is passed:

```
Running testQ0_MainClass

  Test passed
```

If this is not the case, make sure that you have not defined a `package` at the top of your TrackAndTrace.java file.

Question 1: Overlapping Time Periods [10 Marks]

Add a (static) method `overlap` to your `TrackAndTrace` class. It should

- take four `int` arguments representing two time periods:
  - the first argument is the start of period 1,
  - the second argument is the end of period 1,
  - the third argument is the start of period 2, and
  - the fourth argument is the end of period 2;
- return a boolean that is true if the time periods overlap.

A number give the hour in 24-hour format (0 is midnight, 7 is in the morning, and 19 is in the evening). For example, a time period from 17 to 19 overlaps with a period from 18 to 21. Conversely, the time periods 14 to 16 and 16 to 18 do *not* overlap, even though they meet.

You can assume that times do not cross midnight. That is the start time is always a smaller number than the end time. For example, the time period 19 to 21 is valid, but the period 23 to 1 is not.

After completing this question, compile your code with

```
$ javac TrackAndTrace.java
```

and then run the tester

```
$ java -jar cs1811Assignment1Tester.jar
```

You should see that `testQ0_MainClass` and hopefully all tests starting with `testQ1_` now pass.

Question 2: Reading the Customer List [40 Marks]

Add a (static) method `getOverlaps` to your `TrackAndTrace` class. It should do the following.

- Take two `int` arguments representing the start and end of an infectious time period.
- Ask the user `Enter the number of customers:` (use `println` for this) and then read the number of customers from the keyboard.
- For each customer
  - ask the user to `Enter the customer's name:` and read it from the keyboard,
  - ask the user to `Enter the arrival time:` and read it from the keyboard,
  - ask the user to `Enter the departure time:` and read it from the keyboard,
  - print `____ needs a test.` if they overlap with the infection time period (where _____ is the customer's name), or print `____ does not need a test.` if they do not.

Don't forget to compile your code and run the tester. All tests starting `testQ2_` should now pass.

Question 3: Your Main Method [20 Marks]

Add a `main` method to your `TrackAndTrace` class. This should do the following.

- Ask the user to `Enter the start time:` and read the arrival time of the infected person from the keyboard.
- Ask the user to `Enter the end time:` and read the departure time of the infected person from the keyboard.

- Use `getOverlaps` to read the customer list and print out who needs testing.
- At the end, print out `There are n tests needed.` where `n` is the number of people who need a test.

For the last bullet point, you will need to change your `getOverlaps` method to return how many overlaps were found.

Don't forget to compile your code and run the tester. All tests starting `testQ3_` should now pass.

Note: this method needs a `Scanner` over `System.in` but cannot take one as an argument. See this tutorial for a method to do this without arguments.

Question 4: After Midnight [10 Marks]

In the previous exercises, we assume that time periods do not cross midnight. Update your `overlap` method so that time periods might cross midnight. For example, 22 to 1 would mean the customer arrived at 10pm and left and 1am. This time period should overlap with 21 to 2 but not 2 to 21.

You can assume that all venues open on or after 7, and close before or at 6. In this case the time period 2 to 4 would overlap with 23 to 3.

Don't forget to compile your code and run the tester. All tests starting `testQ4_` should now pass.

Question 5: Reading Files [15 Marks]

The final marks of an assignment should be designed to test the ability of students to show independence and creative thought. This question will go beyond the aspects of the module taught so far, and look ahead. In particular, you may wish to view the videos on array (slides) and advanced arrays (slides) (Week 7 (9 Nov)) that covers command line arguments and reading files (slides)(Week 8 (16 Nov)). You may also look at Section 5.3 (arrays) and Section 6.3 (files) of the module textbook.

If there are a large number of customers, it will take a long time to enter each visitor individually each time a case is reported. It would be more convenient to record the customers in files and read the data from the files when needed.

Watch the videos above, and then extend your program so that the user can supply a list of files on the command line. If no files are specified, the program should ask the user to enter each customer individually. Otherwise, your program should iterate through the list of files, and read the customer data from them. An example file might be formatted as follows:

```
Yusif 18 19
Amber 17 22
Marco 14 17
```

An example run might be

```
$ java TrackAndTrace customer-log-1.txt customer-log-2.txt
Enter the start time:
16
Enter the end time:
18
Amber needs a test.
Marco needs a test.
There are 2 tests needed.
```

If a file is not found, your program should print "WARNING: <filename> not found." and continue to the next file, where "<filename>" is the name of the file that was not found. An example interaction is shown below.

```
$ java TrackAndTrace notafile.txt customer-log-1.txt
Enter the start time:
16
Enter the end time:
18
WARNING: notafile.txt not found.
Amber needs a test.
Marco needs a test.
There are 2 tests needed.
```