EECS1021, Winter 2020

Lab Test 2

Instructions

This is a 24-hour programming test. The deadline is **Saturday April 4th 2020 3pm** Before you begin, it is important that you read this entire document for instructions and advice.

If you are unsure how to interpret the programming task, write your assumptions in a comment.

You must submit your code through two systems: WebSubmit and Crowdmark via Moodle. We will use your WebSubmit submission to run tests on your actual .java files. We will use Crowdmark to give you feedback on your code in pdf format. If you only submit through one platform, your grading may be delayed or you might not be graded at all!

You may submit as many times as you want. See below for submission instructions. Your most recent submission will be the one recorded. Submit your work early and often. Work not submitted will not be marked. Work submitted late will not be marked. Submit .java files, NOT .class. **Re-download your files and check them before the deadline.**

You can use any API methods to solve these tasks.

We will use plagiarism detection software to identify any students who copied from another student/ let someone copy from them. If this happens, both of you will receive an automatic 0 and will be reported.

Write your name, student number and Passport York ID (first part of your York email address) as a comment in each file!

Download the zipped project file from Moodle and import it as with the in-lab exercises: Extract the zip file. Launch Eclipse and choose File > Import. Select General > Existing Projects into Workspace, click Next, click Browse..., select the directory called LT2 Starter, and then click "Finish".

Important: None of the methods that you define can contain any print statements.

Good luck and may you do your best!

Grading

If your submitted classes altogether compile with the given Tester class, then you already receive 10% of the marks for the test.

It is absolutely critical that you submit Java code that compiles (i.e., no red crosses shown on the Eclipse editor). **Your grade will be zero if your code contains any compilation errors** (i.e., syntax errors or type errors). In labtest 1 we gave partial credit for code that did not compile but you should be more experienced by now and you have 24 hours to fix any errors. If there is a part of your code that does not compile, don't include it in your submission.

To determine the remaining 90% of your marks, we will run test cases on your submitted classes. Say we run 10 test cases on your submitted code, and your submitted code compiles and passes 6 of them, then your final marks are: 10 + 90*6/10 = 64.

Your code needs to be general. That means it needs to work with any input numbers, not just with the given example numbers in the Tester class. Don't hardcode!

Programming Task

In this lab test you are required to code Java classes and methods. Only a tester class and its expected console output are given to you. The tester class illustrates how instances of classes are supposed to be created, and how methods may be invoked on their instances. **Don't modify the tester!** From this tester class, which does not compile to begin with, you are required to:

1. Identify the missing classes and methods (constructors, accessors, and mutators).

2. Create all missing classes, as well as add headers of all required methods (i.e., name, return type, input parameters, and return default values for accessors).

Completing the above two tasks should make everything compile.

3. Implement the required methods (with any extra attributes or helper methods which you consider necessary), such that executing the given tester produces the expected console output.

System Requirements

Percentages behind each paragraph signify the % of your grade determined by each part.

0) Code compiles (10%)

1) You need to develop a flight management system for an airline. Each flight is characterized by its flight number, origin airport, destination airport and distance. Airports are characterized by their three-letter airport code and country code. Airplanes have a type name, range in km and number of seats. Passengers have a name and integer passport number. (53.3%)

2) Each flight stores a list of passengers. (6.7%)

3) Passengers store a list of flights they are taking. When a passenger is added to a flight, this flight also needs to be added to the list of flights stored by the passenger. (3.3%)

4) Each flight has an airplane type. Before a plane is added, the method needs to check whether the plane's range is sufficiently large for the flight distance. If successful, the method needs to return true. If unsuccessful, it needs to return false. (6.7%)

5) A method is needed to determine whether the flight is overbooked i.e. it checks whether there are more passengers on the flight than seats on the plane. If the number of passengers is larger than the number of seats, the method needs to return by how many passengers the flight is overbooked. If not, the method needs to return zero. (6.7%)

6) Flights can be domestic (i.e. origin and destination have the same country code) or international (i.e. origin and destination have different country codes). We need a boolean method that determines whether a flight is international or domestic. (6.7%)

7) The airline needs a list of any passengers on a particular flight who have not yet entered their passport numbers, which is signified by a passport number of 0 in the system. (3.3%)

8) Passengers receive frequent flyer points for every flight they take. 1 point for domestic flights, 3 points for international flights. Based on a passenger's list of flights, the system needs to calculate their total number of points. (3.3%)

You are required to write, in valid Java syntax, classes, attributes, and methods to implement the above system requirements. Study the FlightTester class and its expected output carefully. It indicates the classes and headers of methods that you need to define. You are forbidden to define additional classes, whereas you are free to declare extra attributes or helper methods as you find necessary.

Submitting Your Work

You need to submit your code through **two different systems**: WebSubmit and Crowdmark.

WebSubmit

Open the Web Submit page in a new browser window or tab. If necessary, authenticate with Passport York. Once authenticated, ensure the correct year, term, and course are selected. Click on "Browse", navigate to your workspace directory, project directory, and `src` directory. Select your solution **.java** file(s) not including the Tester.

Your work is only submitted when you click on the "Submit Files" button. Do that to submit your work. Keep this browser window or tab open to submit your work again later if you make any changes that you want marked. **You will have to "browse" for the files before submitting them again.** Log-out when you are finished.

Make sure to redownload and check your files before the end of the labtest! This is very important to make sure nothing went wrong.

Crowdmark

Copy all of your classes into a text editor such as Microsoft Word and save them as one **.pdf** file not including the Tester. Start each class on a new page. Submit this pdf to Crowdmark using the Labtest 2 pdf submission link on Moodle.