EECS 2030 Winter 2020: Lab 3

Lassonde School of Engineering, York University

INTRODUCTION TO LAB-3

Students in the same lab section are allowed to work in groups of 2 or 3 students. This lab will be graded for style and correctness. This lab is due on **21st Feb 2020**.

STYLE RULES FOR CODING

The style rules are not overly restrictive in EECS2030.

1) Your programs should use the normal Java conventions (class names begin with an uppercase letter, variable names begin with a lowercase letter, public static final constants should be in all caps, etc.).
2) In general, use short but descriptive variable names. There are exceptions to this rule; for example, traditional loop variables are often called i, j, k, etc. Avoid very long names; they are hard to read, take up too much screen space, and are easy to mistype.
3) Use a consistent indentation size. Beware of the TAB vs SPACE problem: Tabs have no fixed size; one editor might interpret a tab to be 4 spaces and another might use 8 spaces. If you mix tabs and spaces, you will have indenting errors when your code is viewed in different editors.
4) Use a consistent and aligned brace style.
5) Insert a space around operators (except the period ".")
6) Avoid using "magic numbers". A magic number is a number that appears in a program in place of a named constant.
7) A good IDE (integrated development environment) such as eclipse will correct many style errors for you. In eclipse, you can select the code that you want to format, right click to bring up a context menu, and choose Source $\rightarrow$ Format to automatically format your code.

LAB EXERCISE

Complete the implementations of various classes in a Flight Management System. Read the API for FlightManagementSystem carefully, the API of this lab is available under doc folder in the Java project folder (doc$\rightarrow$index.html). Some example tests are given to you in the Tester class. You are required to write (at least three) additional tests to further ensure the correctness of your Flight Management implementations.

We consider a database that stores information about passengers and flights. Each passenger record (e.g., String name, double TicketAmount, int flightID) can be uniquely identified by a String ID (e.g., "e1"), whereas each flight record (e.g., String airline and String flight

for flight name and airline name, respectively) can be uniquely identified by an integer id (e.g., 2000).

You must implement all methods in the **FlightManagementSystem** by manipulating the two attributes **passengers** and **flights**, each of which declared as a HashMap.

```
 class FlightManagementSystem {
HashMap<Integer, FlightInfo> flights;
HashMap<String, PassengerInfo > passengers;
/* All methods must be implemented via 'flights' and 'passengers'
*/
}
```

Override the **compareTo** and **equal** methods in the PassengerInfo class. Override the **equals** method in the FlightInfo class.For simplicity, you can assume that a passenger cannot be added in multiple flights. For this question, you are required to submit Java files such as **FlightInfo, PassengerInfo, FlightManagementSystem, Tester** in your Eclipse project. Note that the exception classes are available in the project folder.

LAB EXERCISE

Complete the implementations of various classes in a bookStore. Read the API for bookStore carefully, the API of this lab is available under doc folder in the Java project folder (doc→index.html). Some example tests are given to you in the **Testerv3** class.

We consider a database that stores information about books and owners. Each book has (e.g., String title, int price, int yearPublished), whereas BookStoreOwner has (String name and long id). You must implement all methods in the **BookStore**.

For this question, you are required to submit Java files such as **Book, BookStore, BookStoreOwner, SortBooksbyYear** in your Eclipse project.

SUBMIT INSTRUCTIONS FOR STUDENTS NOT WORKING IN A GROUP

Submit your solutions using the submit command. Refer to lab0 (manual) webpage for instructions. Remember that you first need to find your workspace directory, then you need to find your project directory. In your project directory, your files will be located under different package directories. For example, your **FlightManagementSystem.java** should be in the directory **EECS2030_LAB3W20/src/fms**.

Once you are in the directory, issue **submit 2030 lab3 FlightInfo.java, Passenger-Info.java, FlightManagementSystem.java, Tester.java**

Repeat the above steps for each of the other program files. At any time and in any directory, you can view the files that you have submitted by issuing **submit -l 2030 lab3**

SUBMIT INSTRUCTIONS FOR STUDENTS WORKING IN A GROUP

If you are working in a group, in addition to submitting program files as shown above, you also need to submit a text file containing EECS login names of your group members.

Create a plain text file named **group.txt**. You can do this in eclipse using the menu File $\rightarrow$ New $\rightarrow$ File, or, use any other text editors. Type your login names into the file with each login name on its own line. For example, if the students with EECS login names rey, finn, and dameronp, worked in a group the contents of **group.txt** would be:

rey
finn
dameronp

Submit your group information using submit command. In the directory where the text file is saved, issue **submit 2030 lab3 group.txt**

SUBMIT INSTRUCTIONS FOR STUDENTS OUTSIDE PRISM LAB

It is possible to submit work from outside the Prism lab, but the process is not trivial; do not attempt to do so at the last minute if the process is new to you. The process for submitting from outside of the Prism lab involves the following steps:

1) transfer the files from your computer to the undergraduate EECS server red.eecs.yorku.ca
2) remotely log in to your EECS account
3) submit your newly transferred files in your remote login session using the instructions for submitting from within the lab
4) repeat Steps 1 and 3 as required

Windows users will likely need to install additional software first. Mac users have all of the required software as part of MacOS.