COE528 (Winter 2024) Lab 1: Review Java programming and JUnit

General Lab Rule
Each lab must have its own directory (folder). For the first lab, the directory is lab1. For the second lab, the directory is lab2, etc.

Objectives

- Review Java, JUnit and Netbeans.
- Write subclasses.
- Use composition.
- Use instance and static variables.
- **Duration: two weeks.**

Flight Booking System

Implement a straightforward model of a flight booking system. In our model, we will have Flight objects (representing flights), and passengers (either a frequent flyer **member** or a **non-member**) who want to book a flight for travel. If a seat is available on a given flight, then a ticket will be issued at a price depending on whether the ticket is for a member or a non-member. To develop this system, we will need six classes.

I. Flight class:

(a) Implementing the Flight class:

Each object of this class represents a flight. The Flight class consists of instance variables, a constructor and instance methods.

This class has seven instance variables:

- flightNumber of type int
- origin of type String
- destination of type String
- departureTime of type String
- capacity of type int
- numberOfSeatsLeft of type int
- originalPrice of type double

This class should have:

- a constructor that initializes the instance variables. It takes six parameters. If the specified origin and destination are equal, an IllegalArgumentException should be thrown to abort the construction. The numberOfSeatsLeft variable is initialized depending on the specified capacity.
- getters and setters for each instance variable.
- a bookASeat() method. If the numberOfSeatsLeft is greater than 0, this method decrements the numberOfSeatsLeft variable and returns true. Otherwise, it returns false.

This class should override the toString method that returns a String representation of a Flight object. The String should include the flight number, the origin, the destination, the departure time and the original price. e.g., "Flight 1030, Toronto to Kolkata, 03/02/99 7:50 pm, original price: 1000$"

(b) Testing the Flight class (use JUnit 4):

Once the Flight class has no compilation errors:

- Write the code for the JUnit test method `testConstructor`() that tests the `constructor` by providing it with valid arguments.

- Write a JUnit test method `testInvalidConstructor()` that tests the `constructor` with invalid arguments. The test should pass only if an IllegalArgumentException is thrown.

- Test the public methods that include all the getters, setters, the bookASeat() method and the toString() method, using Junit.

II. Ticket class:

This class represents a flight ticket and has four instance variables:

  - passenger of type Passenger
  - flight of type Flight
  - price of type double
  - number of type int. Each Ticket has a unique ticket number (the first Ticket has ticket number 1, the second has ticket number 2, etc. (Hint: implementing this will require adding a static variable.)

This class should

- have a constructor that takes three parameters: Passenger p, Flight flight, and double price. It should initialize all the instance variables accordingly.

- getters and setters for each instance variable.

- override the toString() method that returns a String representation of a Ticket object. The String should include the passenger's name, information about the flight (e.g. the flight number, the origin, the destination, the departure, and the original price) and the actual ticket price. e.g., "Julia Chow, Flight 1030, Toronto to Kolkata, 03/02/99 7:50 pm, original price: 1000$, ticket price: $600.00"

III. Passenger class:

This is an abstract class. The subclasses of this class are Member and NonMember. The class has attributes common to all kinds of passengers. It contains two instance variables:

- name of type String
- age of type int.

This class should have a constructor for initializing the instance variables, getters and setters for each instance variable. This class also has an *abstract method* `double applyDiscount(double p)`. This method should be overridden by both the subclasses. In the subclasses, this `applyDiscount` method should return a price after applying the appropriate discount.

IV. Member class:

A Member object represents a frequent flyer member. This class has one instance variable:

- yearsOfMembership of type int

This class extends the Passenger class and overrides the applyDiscount(double p) method as follows. If the member has a membership for more than 5 years, then a 50% discount will be applied. If the member has a membership for more than 1 year but less than or equal to 5 years, then a 10% discount will be applied. Otherwise, there is no discount.

V. NonMember class:

A NonMember object represents a passenger who is not a frequent flyer member. This class has no instance variables.
This class extends the Passenger class and overrides the applyDiscount(double p) method as follows. If the age of the person is more than 65, a 10% discount will be applied. Otherwise no discount.

VI. Manager class:

A Manager object manages the flights of a specific airline. It maintains an array of flights and an array of issued tickets. It should have the following methods:

- public void createFlights(): this method should populate the array of flights.
- public void displayAvailableFlights(String origin, String destination): this method should display all the available flights from origin to destination. It should display only those flights that are not yet fully booked.
- public Flight getFlight(int flightNumber): this method should return the Flight object for the specified flight number.
- public void bookSeat(int flightNumber, Passenger p): This method first tries to find a flight for the given flight number. If such a flight exists, then it tries to book a seat on that flight. If the booking is successful, then apply the appropriate discount on the price depending on whether the passenger is a member or a non-member (Polymorphism works here). Finally, it issues a ticket with the appropriate price.
- public static void main(String[] args): This method should call all the other methods of the Manager class to see if they work. You are responsible for developing a convincing plan for the main method, i.e., for convincing the TA that your methods work properly.

Step 1: *Create a Netbeans project and Person class*

1. Create a Netbeans project called `FlightBooking,` which should be placed in a folder called `lab1` (all lowercase and no spaces). The `lab1` folder should itself be in your `coe528` folder.
2. Create a Java file (class library type) called `Flight;` set the package to `coe528.lab1;`
3. Implement the Flight class as specified in I(a).
4. Write JUnit tests to test the methods of the Flight class as specified in I(b).
5. Create a Java file (class library type) for each remaining class and set the package to `coe528.lab1;` (All Java files in this lab should have the package declaration).
6. Implement the classes as specified in II, III, IV, V, and VI.
7. Generate the javadocs, compile and run the project.
8. It should compile correctly and produce output.

Step 2: Submit your lab

Due date: 11:59 pm, the day before your scheduled lab 2 session.

You must submit your lab electronically on D2L. Please zip up your NetBeans project containing all source files and submit it to the respective assignment folder on D2L.