JP2 2019: Lab exam (Version D)

This lab exam is intended to be completed during the 9-11am Tuesday timeslot (Lab sections 7 and 8). If you are taking the exam at a different time, you should locate the correct specification instead of this one.

Overview

You will model a simplified airline "frequent-flier" programme. A member of the frequent flier has a number of "miles", which they can earn and use in the following ways:

- Every time they travel on a flight, they **earn** miles depending on the details of the flight
- Every time they **purchase** a flight, they **spend** miles depending on the cost of the flight

In addition, the programme has four possible levels: Basic, Bronze, Silver, and Gold. A member can also choose to use their miles to **upgrade** to the next level, as follows:

- All members start at Basic
- Going from Basic to Bronze costs 100 miles
- Going from Bronze to Silver costs 500 miles
- Going from Silver to Gold costs 1000 miles

Here is an example. A member at rank Basic has a balance of 50 miles, and has two upcoming flights that they have not yet taken: Flight 1 will earn 70 miles, and Flight 2 will earn 100 miles.

- After the member takes Flight 1, their balance is 120 miles and Flight 1 is removed from the upcoming flights list
- After the member also takes Flight 2, their balance will be 220 miles and their list of upcoming flights is empty
- The member then chooses to rank up – their rank goes up to Bronze, and their miles balance goes to 120
- They then spend 80 miles on a new flight (Flight 3, which will earn 200 miles) – their balance will then be 40 miles, and Flight 3 is added to the list.
- Next, they take Flight 3 – their balance will then go up to 240 miles and their pending flight list will again be empty

Task 1: Rank (2 marks)

Note about implementation: all classes created in this exam should be put in the ***flights*** *package.*

You must create an enumerated type **Rank** with the following values:

BASIC, BRONZE, SILVER, GOLD

Task 2: Flight (6 marks)

You must create a class **Flight** representing a flight that can be purchased or taken. A Flight has the following properties; you should create a field in the class for each.

- origin (a String)
- destination (a String)
- price in miles (an integer)
- miles earned (an integer)

The **Flight** class should have a constructor with the following signature to initialise its fields:

public Flight(String origin, String destination, int priceInMiles, int milesEarned)

The **Flight** class should also include the following:

- A complete set of **get** methods, and no **set** methods
- Appropriate implementations of **equals()**, **hashCode()**, and **toString()**

Task 3: Member (3 marks)

Create a class **Member** representing a member of the frequent-flier programme. The **Member** class should have the following properties:

- The member's name (a string)
- The member's current miles balance (an integer)
- The member's current rank (a Rank)
- A list of the Flights that they have booked but not yet taken (a list of **Flight** objects)

The **Member** class should also have a constructor that sets all of its fields. **You do not need to override equals(), hashCode(), or toString() for this class.**

Task 4: Member methods (8 marks)

Define methods in the **Member** class to implement the behaviour described in the overview. The methods should have the following signatures and behaviours.

- **public void purchaseFlight(Flight flight)**
  - If the member has enough miles to purchase the flight, the miles are deducted from the total and the flight is added to the end of the booked flights list
  - Otherwise, this method should throw an **IllegalArgumentException**
- **public void takeNextFlight()**
  - If there is at least one booked flight in the list, it is removed from the list and the miles for that flight are added to the member's miles balance
  - If there is no upcoming flight, this method should do nothing
- **public void rankUp()**
  - If the user can rank up – that is, their level is not Gold and they have enough miles to move up to the next level – then their rank is updated to the next level and the necessary miles are deducted from their total.
  - If the user cannot rank up, this method should do nothing.

What to submit

On Moodle, go to **Lab Exam Submission – Version D** and upload the following three files:

- Rank.java
- Flight.java
- Member.java

Be sure to submit to the correct assignment link, and be sure to submit before the deadline.