**Langara College**

**Department of Computing Science & Information Systems**

**CPSC1181 – Object-oriented Computing**

###### **Lab3: Objects and Classes**

**Objectives:**

1. Create classes to model objects
2. Perform JUnit tests

**Instructions:**

1. Open Eclipse and make **LabProjects** as the workspace.
2. Inside Eclipse, create a project named **Lab3** to store all the files for this lab. Make sure uncheck **Create module-info.java file**.
3. Add Javadoc comments and regular comments to all your code

**Problems [60 marks]**

**Problem 1: [20 marks] The Account class**

1. Inside Eclips, right-click project Lab3->New->Class to add a new class
2. Enter **lab3** as the package name
3. Enter **Account** as the class name
4. Leave the modifiers as public and none
5. Do not check **public static void main(String[] args)**
6. Check **Generate comments**
7. Click Finish
8. [**15 marks**] Create a class named **Account** that contains:
9. A private int data field named **accountNumber** for the account.
10. A private double data field named **balance** for the account.
11. A private double data field named **annualInterestRate** that stores the current interest rate.
12. A private **static** int data field named **totalAccounts** to keep track how many accounts have been created by the bank. Its initial value is set to 1000000. When a new account is created, totalAccounts is incremented by 1 and this new value is assigned to the account number of the new account.
13. A **no-arg constructor** that creates a default account with balance 0 and annualInterestRate 0.
14. Another **constructor** that creates an account with a given balance and a given annualInteresRate.
15. The accessor and mutator methods for balance and annualInterestRate, and the accessor for accountNumber. No mutator should be used for the accountNumber because the accountNumber should not be changed after the account is created.
16. A method named getMonthlyInterest() that returns the monthly interest, which is balance multiplied by monthly interest rate.
17. A method named withdraw(amount) that withdraws a specified amount from the account.
18. A method named deposit(amount) that deposits a specified amount to the account.
19. Create a toString method that returns a string summary of the Account object.
20. [**5 marks**] Create the JUnit test class called **TestAccount** to ensure that all of the methods of your class work correctly. The toString method does not need to be tested.

**Problem2: [40 marks] The Train class**

1. Add the new class named **Train** to the package **lab3**
2. Leave the modifiers as public and none
3. Do not check **public static void main(String[] args)**
4. Check **Generate comments**
5. Click Finish
6. [**30 marks**] Create a class called **Train** that simulates a freight train.
   1. Each Train object should have a name, a number representing the power of its locomotive(s), and an array of freight cars. Each freight car is represented by an integer representing how many tons that it weighs. There are no locomotives in the cars array.
   2. Create the constructor so that it accepts 2 parameters, the train’s name and the total power of the locomotive(s). The train initially has no cars.
   3. Create the getters and setters necessary for the instance data of this class. Only create a getter for the array of freight cars, no setter. Freight cars can be added/removed with other methods. Create a toString method that returns a string summary of the Train object.
   4. Create the method getTotalWeightOfCars() that returns the sum of the weights of all of the cars and getNumberOfCars() that returns how many cars the train has (does not include locomotives).
   5. Create a method to compute the maximum speed of the train. Subtract the total weight of the train from the pulling power of the locomotive(s). This gives the maximum speed in km/h. The highest possible speed is 150 km/h. So, if a train has locomotive power 160 and [20, 30, 20] as its cars, its maximum speed would be (160 – (20 + 30 + 20)) = 90. If we assume that the locomotive power is 1000 and the train has no cars, the top speed would be 150 as this is the maximum speed possible.
   6. Create a method void removeAllCars() that will remove all the cars from the train.
   7. Create a method void addCars(int... weights) that will add cars to the end of the train. Add the passed in weights to the collection of cars.
   8. Create a method void mergeTrains(Train other) that will add all of the locomotive power and cars from the other train to the current train. So after calling train1.mergeTrains(train2), train2 will have no cars and 0 locomotive power when done. They will all have been added to train1.
7. [**10 marks**] Create the JUnit test class called **TestTrain** to ensure that all of the methods of your class work correctly. Use assertArrayEquals when testing the methods that return arrays. The toString method does not need to be tested.

**What to hand in**

1. Click Project->Generate Javadoc… to geneate the Javadoc documents for all the java files inside Lab3->src->lab3. Leave all the other selections unchanged.
2. Zip the folder **Lab3** and upload it to D2L.

**When to hand in**

By 11:59pm, Wednesday, January 25, 2023.