Homework 3-Weightage 5%: (Contains 3 questions)
Announcement Date: Wednesday, March 04, 2020
Due Date: March 08, 2020 at 4pm
Please turn in .java files and not .doc/docx files and the screenshots of outputs.

Question 1: This Question requires you to model a bank system. (weightage 2%)
For this question, data should be immutable and only be accessible through the appropriate methods.

A-Create a class BankAccount:

1- with variables for storing the account number and balance.
2- One parameter constructor to set the account number. The same constructor will set the balance to zero.
3- Two methods for depositing and withdrawing amounts (that will be passed as parameters). The methods should check that the amount to be deposited or withdrawn is not negative.
4- Getters methods for getting the account number and balance.
5- The class should override the toString method.

B- Using the BankAccount class as a base class, write two derived classes called `SavingsAccount` and `CurrentAccount`. A `SavingsAccount` object, in addition to the attributes of a BankAccounnt object, should have an interest variable and a method which adds interest to the account. A `CurrentAccount` object, in addition to the attributes of a BankAccounnt object, should have an overdraft limit variable. Ensure that you have overridden methods of the BankAccount class as necessary in both derived classes.

C- Now create a `Bank` class, an object of which contains an array of BankAccount objects. Accounts in the array could be instances of the BankAccount class, the `SavingsAccount` class, or the `CurrentAccount` class. Create some test accounts (2 of each type).

D- Write an update method in the bank class. It iterates through each account, updating it in the following ways: Savings accounts get interest added (via the method you already wrote); CurrentAccounts get a letter sent if they are in overdraft.

E- The `Bank` class requires methods for opening and closing accounts, and for paying a dividend into each account.

Hints:

- Note that the balance of an account may only be modified through the `deposit(double)` and `withdraw(double)` methods.
- Be sure to test what you have done after each step.

You need to submit:

1- BankAccount.java file
2- SavingsAccount.java file
3- CurrentAccout.java file
4- Bank.java file
5- Test.java file with main method for creating and printing the objects of each class.

Question 2 (weightage 1.5%)

Create a class Player that holds information about an athlete: name, team, and uniform number. Create another class PlayerCompare that uses the Player class to read in information about two baseball players and determine whether or not they are the same player.

1. The class PlayerCompare reads in two players and prints "Same player" if they are the same, "Different players" if they are different. Use the equals method, which Player inherits from the Object class, to determine whether two players are the same. Are the results what you expect?

2. The problem above is that as defined in the Object class, equals does an address comparison. It says that two objects are the same if they live at the same memory location, that is, if the variables that hold references to them are aliases. The two Player objects in this program are not aliases, so even if they contain exactly the same information they will be "not equal." To make equals compare the actual information in the object, you can override it with a definition specific to the class. It might make sense to say that two players are "equal" (the same player) if they are on the same team and have the same uniform number. Use this strategy to define an equals method for the Player class. Your method should take a Player object and return true if it is equal to the current object, false otherwise. Test your PlayerCompare program using your modified Player class. It should give the results you would expect.

You are required to submit Player.java and PlayerCompare.java and a testplayer.java that creates the objects and tests.

Question 3 (weightage 1.5%)

A-Define an enumeration type COLORS for colors BLACK, RED, BLUE, GREEN, and YELLOW.

B-We can add the concrete instance methods in enum classes. Define a concrete method printColor which displays and returns the colors based on which field of the enum class is calling it.

C- Define an abstract method colorCombo which prints if it is a primary colour, if not then how the particular color is formed.

RED: Primary color
BLUE: Primary color
Yellow: Primary color
Green: Formed by combination of yellow and blue
Black: Formed by combination of red yellow and blue

Remember, we must **implement the abstract method at each enum field**, individually. You are required to add abstract methods for each enum field. So, there will be five definitions of colorCombo. Create a class EnumDemo to create objects for each of the five colors. Then call the printColor method and finally call the colorCombo method. Please submit all java files.

****

Happy Java-ing!!