CSCI 2540 Assignment 3

(100 points)

Due date: Thursday, Feb. 10 (by 11:59pm)

For this assignment, you will be writing three classes. One of them is the **StudentAccount** class that represents student account. The account can be used for deposit, charge, and transfer. The other class is for a special type of StudentsAccount which can receive rewards under certain conditions. This class is the **RewardsAccount** which is a subclass of StudentAccount. You also need to write **StudentAccountDemo** class to test the two classes.

For StudentAccount class, you will need to have variables to represent the account number (acctNo – using long integer type) and balance (using double type) for each account. **The variables should be private.** The account number will be generated automatically based on a counter. For example, the first account created will have the account number 1, and the one created next will have the account number 2. Therefore, you will also need to have a variable to represent the counter and update this counter accordingly. The counter is not for individual objects; instead it will be shared by the entire class.

For RewardsAccount class, you will also have a double variable (rewards) to represent the rewards balance.

For both classes, you will need to decide which variables and methods should be static and which ones should be non-static.

1. StudentAccount class

You will need to write the following methods for the StudentAccount class:

Constructors

All constructors should be defined as "public".

- A default constructor should be included. The default balance is set to zero and the account number is automatically generated based on the value of the counter.
- A constructor should be included that receives a single double value as a parameter. The parameter will provide the initial balance for the account. Account number will be automatically generated.

Public Methods

- **getAcctNo** - This is a "get" method used to retrieve the account number.
- **getBalance** - This is a "get" method used to retrieve the account balance.
- **getCounter** – this is a “get” method used to retrieve the number of accounts created, which also serves as the counter for automatically generating the account number.

- **setBalance** – This is a "set" method used to set the account balance.

  Since the account number and counter should not be modified manually, so there should be no set method for those two variables.

- **deposit** - This method should receive a double value as a parameter. The parameter indicates how much will be deposited into the account. The amount should be positive (otherwise it does nothing and the program should print a message). The account balance should be updated accordingly.

- **charge** - This method should receive a double value as parameter. The parameter indicates how much will be charged to the account and the balance should be decreased accordingly. The charge amount should be positive (otherwise it does nothing and the program should print a message). The new balance is allowed to be negative.

- **transferIn** – this method allows transferring money from another StudentAccount. There are two parameters, an object of StudentAccount and the amount to be transferred in. The amount should be positive (otherwise it does nothing). The balance of both accounts should be updated accordingly. The new balance is allowed to be negative.

- **transferOut** – this method allows a transfer from the current account to another StudentAccount. There are two parameters, an object of the other StudentAccount and the amount to be transferred out. The amount should be positive (otherwise it does nothing). The balance of both accounts should be updated accordingly. The new balance is allowed to be negative.

- **printInfo** – this method takes no parameter. It displays the account information in the following format:
  Account number:
  Current balance:

Overridden Methods

The following method is inherited from the Object class, and you will need to overwrite it.

- **toString** - This method takes no parameters and returns a String object. It returns a String object with the account information in the following format:
  Account number:
  Current balance:

Other methods

Your Account class should include the following method from the Comparable interface.

- **compareTo** - This method should have one parameter of StudentAccount type and return an integer (1, 0, or -1). The method is to test the order of two StudentAccount objects (the "this" object and the parameter) based on their account balance. If the balance of this object is higher, it returns 1; if the balances are the same, it returns 0, otherwise it returns -1. Note that in order to have this method, **your StudentAccount class should implement the Comparable<StudentAccount> interface**.

2. RewardsAccount class

This class is a subclass of StudentAccount class. It has a new variable (rewards – using double type) that records the rewards balance.

Constructors

All constructors should be defined as "public". Keep in mind that a subclass's constructor should first call a constructor in the super class.

- A default constructor should be included. The default rewards amount is zero.
- A constructor should be included that receives a single double value as a parameter. The parameter will provide the initial account balance for the account. If the initial account balance is great than or equal to $100, then $5.00 will be received for the rewards.

Public Methods

- **getRewards** - This is a "get" method used to retrieve the rewards balance.

- **redeemRewards** – this method receives no parameter. If the current rewards balance is at least $25, rewards can be redeemed and credited to the account balance. After redemption, the reward balance should be reset to zero. If the rewards balance is less than $25, rewards cannot be redeemed and a message should be printed informing the user that there is insufficient amount for redeeming the rewards. No additional rewards can be given when redeeming rewards.

Overridden Methods

The following methods are inherited either from the Object class or the StudentAccount class, and you will need to override them.

- **toString** - This method takes no parameters and returns a String object. It returns a String object in the format of:
  Account number:
  Current balance:
  Rewards balance:

- **deposit** – the only difference is that if the deposit amount is at least $100, then $5 will be added to the rewards balance.

- **printInfo** – this method takes no parameter. It prints the account information in the following format:
  Account number:
  Current balance:
  Rewards balance:

3. StudentAccountDemo class

For the **StudentAccountDemo** class, you need to include a main method to test all the methods in the StudentAccount and RewardsAccount class. You need to create multiple objects to be able to test all the methods in both classes. You also need to test multiple cases for different possibilities.

Technical Notes

- At the top of EVERY Java file you create, you must include a comment which states your name.
- YOU MUST PROVIDE COMMENTS for EVERY method that is included in your class. It does not matter how simple the method is, you MUST still comment it. **Follow the standard "javadoc" commenting style for formatting your comments for each public method.**
- You do not need to finish writing the entire class before you can start testing it.
- Programs that do not compile will receive an automatic grade of "F".

Submission instructions:

You need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 3, create a new package and **name your package as assg3_yourPirateId** (use lower case for your pirate id), such as assg3_smithj19. You also need to include a statement such as "package assg3_smithj19;" at the beginning of each of your .java file. **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. (You can use 7-zip software to zip files/folder).