CSCI 2540 Assignment 3

(100 points)

Due date: Thursday, Sept. 22, by 11:59pm

For this assignment, you will be writing four classes, including **Account**, **CheckingAccount**, **SavingsAccount**, and **AccountTest** class.

The **Account** class represents a bank account. An account can be used for deposit, withdraw, and transfer. Each account has two variables, acctNo and balance. Use String type for acctNo and double type for balance. The variables should be private.

CheckingAccount and **SavingsAccount** are subclasses of Account. CheckingAccount has an additional variable that represents the overdraft amount. It also has a static variable that represent the overdraft fee. SavingsAccount has an additional variable that represents the interest rate.

AccountTest class is the class with the main method. It tests all the methods (including constructors) of the three other classes.

1. Account class

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

Constructors

All constructors should be defined as "public".

- A one-parameter constructor should be included. The parameter is a given account number (of String type). The default balance is zero.
- A two-parameter constructor should be included. The two parameters are a given account number and an initial balance. The initial balance should be non-negative.

Public Methods

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

  Since the account number should not be modified manually, so there should be no set method for acctNo.
- **deposit** - This method should receive a double value as a parameter. The parameter indicates the amount to be deposited into the account. The amount should be positive (otherwise it doesn't do anything and the program should print a message). The account balance should be updated accordingly.

- **withdraw** - This method should receive a double value as parameter. The parameter indicates the amount to be withdrawn to the account and the balance should be decreased accordingly. The withdrawal amount should be positive (otherwise it doesn't do anything and the program should print a message). If there is not enough money in the account to be withdrawn, a message should be printed and no withdrawal is done.
- **transfer** – This method allows transferring money from this account to another account. There are two parameters, an object of Account and the amount to be transferred to. The amount should be positive (otherwise it doesn't do anything and the program should print a message). The balance of both accounts should be updated accordingly. If there is not enough money in this account, a message should be printed and no transfer is done. For example, to transfer $100 from account a1 to a2, it should call

  ```
  a1.transfer(a2, 100.0)
  ```

- **displayInfo** – 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 parameter and returns a String object. It returns a String object with the account information in the following format:
  Account number:
  Current balance:

- **equals** - This method has one parameter of **Object** type and returns a boolean value. If the object that is passed as a parameter is not any type of account, it should return false. Two accounts are equal if they have the same acctNo.

2. CheckingAccount class

CheckingAccount is a subclass of Account class. It has a new variable called overdraft (using double type) that represents the overdraft amount (such as -200.0). A checking account can have negative balance as long as it is not lower than overdraft amount. But every time an overdraft is done, there is a fixed fee. Use a **static** variable called fee to represent the fee. This overdraft fee is the same for all the checking accounts, so it should be static.

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 one-parameter constructor should be included. The parameter is a given account number. The default balance and overdraft amount are zero.

- A three-parameter constructor should be included. The three parameters are a given account number, an initial balance, and a given overdraft amount. The initial balance should be non-negative.

Public Methods

- **getOverdraft -** This is a "get" method used to get the overdraft amount.
- **setOverdraft -** This is a "set" method used to modify the overdraft amount.
- **getFee -** This is a "get" method used to get the overdraft fee.
- **setFee -** This is a "set" method used to modify the overdraft fee.

Overridden Methods

The following methods are inherited either from the Object class or the Account class, and you will need to overwrite 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:
  Overdraft amount:
- **withdraw** – This method is similar to the withdraw method in Account class except that even if there is not enough money in the checking account, a withdraw can be done as long as it is within overdraft limit, but overdraft fee will be charged. The new balance after the fee charge cannot be lower than the overdraft limit.
- **transfer** – This method is similar to the transfer method in Account class except that even if there is not enough money in the checking account, a transfer can be done as long as it is within overdraft limit, but overdraft fee will be charged. The new balance after the fee charge cannot be lower than the overdraft limit.
- **displayInfo** – this method takes no parameter. It prints the checking account information in the following format:

  Account number:
  Current balance:
  Overdraft amount:

3. SavingsAccount class

This class is also a subclass of Account class. It has a new variable called interestRate (using double type) that records the interest rate (such as 0.03). Saving accounts do not allow overdraft, i.e., the balance cannot be negative.

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 one-parameter constructor should be included. The parameter is a given account number. The default balance and interest rate are zero.
- A three-parameter constructor should be included. The three parameters are a given account number, an initial balance, and a given interest rate. The initial balance should be non-negative.

Public Methods

- **getInterestRate** - This is a "get" method used to get the interest rate.
- **setInterestRate -** This is a "set" method used to modify the interest rate.
- **addInterest –** Add the interest earned to the balance. New balance should be equal to (1+interestRate)*balance.

Overridden Methods

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

- **toString** - This method takes no parameter and returns a String object. It returns a String object in the format of:
  Account number:
  Current balance:
  Interest rate:
- **displayInfo** – this method takes no parameter. It prints the savings account information in the following format:
  Account number:
  Current balance:
  Interest rate:

4. AccountTest class

For the **AccountTest** class, you need to include a main method to test all the methods in Account, CheckingAccount, and SavingsAccount class. You need to create multiple objects to be able to test all the methods (including all the constructors and overridden methods) in all three classes. You also need to test multiple cases for different possibilities when applicable. Please also include a test case that transfers from a checking account to a savings account and vice versa. Please also print sufficient messages about the output.

Technical Notes

- At the top of EVERY Java file you create, you must include a comment which states your name and what the program does. Follow the standard "Javadoc" commenting style (/** ……. */) for formatting your comments.
- YOU MUST PROVIDE COMMENTS for EVERY method that is included in your class. Follow the standard "Javadoc" commenting style (/** ……. */) for formatting your comments and include @param for each parameter and @return if there is a return type that is not void. You should also include Javadoc comments before each constructor.
- 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_smithj21. You also need to include a statement such as "package assg3_smithj21;" at the beginning of each of your .java file (it will be automatically generated in Eclipse if you create the class inside the given package). **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). **Once your zip file is unzipped, it will generate a folder that matches your package name.**