**Assignment 2 Phase 1 – Bank Account (Part 1)**

**Date Given:** Oct. 2, 2018

**Date Due:** Oct. 15, 2018 at midnight in the submit drive

**Objectives:**

* to develop classses using the Best Software Practices guidelines including the use of previously deveoped classes
* to implement and test the design

**Description:**

Most of us have a chequing account. As owners of chequing accounts, we need to keep track of the different activities that are involved with the chequing account to insure that it will continue to be beneficial to us. To assist us in doing this, the bank performs the bookkeeping and periodically presents us with a report on the activities on the chequing account. This assignment looks at how the bank handles our chequing account.

**Overview**

For this phase, you will write two classes that are needed by the bank: Account (which handles all aspect of the chequing account) and Transaction (what activities are done to the chequing account, such as deposit, withdrawal, etc.). In addition, because it is necessary to keep track of the dates when these transactions occur.

**Requirements**

**The Account Class**

The Account class is used to keep track of your chequing account. This will contain variables and methods that will be used by the bank to process the transactions. The following provides a description of the Account class.

1. Data Fields

**Static Variables**

Static variables are used whenever these variables belong to the entire class rather than the object. Note that if an object changes the value of a static variable, it is changed for all objects of that class. For this class, declare 4 static double variables:

1. Interest Rate, expressed as a % - e.g. 0.5

This is the interest rate that is used to calculate the interest earned by the account during the statement period, if applicable.

1. Required balance for interest to be paid.

This is the balance required before interest is paid. As long at the minimum balance of the account for the month is higher than this required balance, then interest is paid on the minimum balance using the daily interest rate.

1. Required balance for Service Charge

This is the balance required before a service charge is applied to the account whenever a cheque is written. If the balance of the account is not less than this required balance when the cheque is cashed, no service charge is applied.

1. Service Charge

This is the amount of the service charge that is applied whenever the cheque is cashed under the conditions described above.

For this assignment, assume that the starting values of these variables are:

Daily Interest Rate: 0.25

Required Balance for interest to be paid: $ 700.00

Required Balance for Service Charge: $ 1,000.00

Service Charge (when applicable): $ 4.50

**Instance Variables**

The following are the regular instance variables of the class:

1. A String variable for the Account Number (e.g. 123-456789)
2. A String variable for the Customer Name (e. g. “John Doe”)
3. A double variable for the current balance
4. A double variable for the balance at the start of this statement period
5. A double variable for the minimum balance for this statement period
6. A String to store the date of the last statement issued
7. Constructor

There are two parameterized constructors.

1. The first takes in 4 parametes: a String for the name, a String for the account number, a double for the starting balance, and a String to set the date of the last statement. The constructor actually constructs the starting point of the chequing account for the current statement period based on the last statement date. This has implications on how the balances are set up. .
2. The second takes 6 parameters: a String for the name, a String for the account number, a double for the current balance, a double for the starting balance, a double for the lowest balance, and a String to set the date of the last statement. This constructor is used when creating the account at the middle of the statement period.

1. Accessors and Mutators

There should be 10 accessors, one for each of the data fields**.**

However, for purposes of this assignment, because the account balances are changed only by transactions, there is only one mutator, that to change the customer name. Although technically, there should be mutators for the static variables, for the purposes of this assignment, we will not deal with this complication and assume that these do not change for the short term.

1. Processing Methods

There should be methods that will process transactions. At the very least there should methods to perform the following:

1. Making a Deposit

The method is passed a double value for the amount of deposit and the appropriate data field should be updated.

1. Withdrawing from the account

The method is passed a double value for the amount of withdrawal. The appropriate data fields, including the lowest balance if appropriate, should be updated.

1. Processing a Service Charge

Writing a cheque is treated as a withdrawal. However, it is possible that this will incur a service charge. Thus, it is necessary to create a method that will process the service charge.

1. Calculate and process the interest

Interest is earned if the minimum balance is greater than the required balance during the statement period. The interest for the statement period if applicable is the product of the interest rate and the minimum balance for the statement period. Note that the daily interest rate is entered as a percent

The amount of the interest is then added to the balance of the account.

1. Helper Method

For now, there should be a method that will print out the details of the account. It should look like the following (properly formatted):

Account Number: 123-456

Customer Name: John Doe

Current Balance: $ 1,200.00

Opening Balance: $ 600.00

Lowest Balance: $ 400.00

Last Statement Date: 31/10/2018

**The Transaction Class**

The Transaction class will keep track of the transactions that are made on the particular chequing account. Although this class is included in Phase 1, it will really not be used until Phase 2. You have the option of delaying the development of this class until Phase 2. However, there will be a much heavier development in Phase 2 and thus postponing the development of the Transaction class would impact on your work in Phase 2.

The following provides a description of the Transaction class:

1. Data Fields

There are 3 data fields (regular instance variables) for this class:

1. A character variable to store the code for the transaction type (e.g. ‘D’ for deposit)
2. A double variable to store the amount of the transaction (e.g. 30.05)
3. A String to store the date that the transaction had taken place.
4. Constructor
5. Default Constructor

The default constructor initializes the type to blank and the other fields to their default value.

1. Parametrized Constructor

There should be a parametrized constructor containing 3 parameters: a character for the type, a String for the date of the transaction, and a double for the amount of the transaction.

1. Accessors

There should be accessors for each of the instance variables

1. Mutator

To simplify the development, there will be only one mutator (a method called **set**) that takes 3 parameters (as described in the parametrized constructor) and sets the appropriate data fields.

1. Helper Method

There should be a method (called **transDetails**) that will return a String that contains the details of the transaction as described below:

D 12/6/2018 30.05

This will be used in phase 2 to save the transactions of the current statement period to a file.

**Programming / Documentation Requirements**

Ensure that proper style as specified in the standards document are followed, including proper indentation and blank lines to enhance readability. In addition, both classes must be documented to create proper JavaDoc documentation.

**Class Design**

As part of the development process, it is strongly suggested that you develop the class design for both classes based on the description above. This design should be handwritten and should be similar to the tutorial on class design. In addition, you should also include any helper methods you feel is necessary.

Although you do not need to submit these, you should create class designs for both classes before you even start coding the classes. This will help quite a bit in your coding.

**Development**

1. The Account class should be developed first. Ensure that the class is thoroughly tested t to make sure that it functions as required. You should test each method as it is developed. A test client is highly recommended to test every method for this class.
2. Develop the transaction class next and test this class in the same way that you did the Account class.

Note that at this time, the two classes would seem independent of each other. However, these two classes will interact in Phase 2 and hence it is imperative that the two classes are thoroughly tested. If they are not tested properly, this would add more work for Phase 2 that may prevent you from fully developing the solution for Phase 2.

**Submission**

Your assignment should be submitted on the due date. Submit the entire Java folder into the submit drive.