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

**Date Given:** Oct. 16, 2018

**Date Due: Program** Oct 30 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

**Overview:**

This phase will concentrate on processing the transactions for the chequing account for a given statement period. You will develop 2 classes for this phase: The processing class (named **ProcessTrans**) and the client interface class (named **Client** or **ClientInterface**). In addition, it may be possible that you will be required to modify the other classes you have developed in Phase 1.

**The Processor Class:**

This is the main class to be developed in this phase. This class processes the transactions on the chequing account such as deposits and withdrawals as well as transactions not generated by the customer such as service charges and interest earned. In addition, it will also produce a statement of account at the end of the month. In order to perform these functions, the processor class works with an ArrayList of Transaction objects and a particular chequing account. The ArrayList will keep track of all the transactions on the chequing account that have occurred during the statement period.

The following provides a description of the processor class and its methods:

1. Instance variables

The processor class contains 4 instance variables: an ArrayList of Transaction objects, a Account object, the character Transaction type, and the starting date of the statement period.

1. The **run()** method

The run() method is the principal method of this class. It is the method called by the client to start the processing. Its main function is to load the data so that the transactions can be processed. There are two ways to load the data. The user will need to inform the system which one to use.

1. At the beginning

At the beginning of the month, the chequing account is initialized into its beginning state. The ArrayList will be empty and the account will be created using the four parameter Account constructor. The data will be inputted via terminal.

1. During the statement period

When the system is used sometime after the beginning of the month and before the end of the month, the account will have balances that have values as a result of previous transactions that have been processed. Thus, the account must be recreated as of the last access. The data for these balances are located in a file, which have been saved. (This will be discussed later). In addition, the transactions that have occurred since the beginning of the month should be loaded into the ArrayList. The transaction data is also located in a file. Thus, in this manner of loading, the Account data and the transactions data are to be loaded from files. The start date for the statement period is inputted using the terminal.

1. Customer Generated Transactions

There are three types of customer generated transactions. The following describe these transactions:

1. Making a Deposit

To process this type of a transaction, a method in the processor class is created that will accept an amount and a date that identifies the date of the transaction. The method then creates a deposit transaction type using ‘D’ as the transaction code. The account balances are updated using the appropriate method in the Account class.

Once these are done, a Transaction object is created using the transaction data and this object is added to the ArrayList of Transaction objects.

1. Making a Withdrawal

To process this type of a transaction, a method in the processor class is created that will accept an amount and a date that identifies the date of the transaction. It then checks to make sure that there is enough funds to withdraw the amount. If there are not enough funds, no transaction is created and the user is informed.

If there are sufficient funds, the method then creates a withdrawal transaction type using ‘W’ as the transaction code. The account balances are updated using the appropriate method in the Account class.

Once these are done, a Transaction object is created using the transaction data and this object is added to the ArrayList of Transaction objects.

1. Writing a Cheque

Writing a cheque is basically a withdrawal. For simplicity, we assume that there will be sufficient funds to cover the cheque (i.e. no bouncing cheque). However, after the funds are withdrawn, you should check first if the current balance is less than the required balance for service charge. If it is, then a transaction for the service charge, with type ‘S’, is created and added to the ArrayList .

1. Account Info Inquiry

From time to time, the user would be interested in the current information on the chequing account. This should print out the details of the account

1. Listing the Transactions

The user should be able to list all the transactions from the beginning of the statement period. This will allow the user to review the transactions. The format could be similar to:

2/6/2018 Deposit $ 1,200.00

3/6/2018 Withdraw $ 300.00

etc…

1. End of Statement Period Processes

At the end of the statement period, a number of processes are performed. The following describe these:

1. Calculation of Interest

At the end of the period, the interest is calculated using the method of the Account class. For this purpose, the date of the end of the period is entered via the keyboard.

Once the amount of the interest is determined, a transaction with type ‘I’ is created and added to the ArrayList.

1. End of Statement Period Report

At the end of the statement period, a report should be generated that lists the transactions and the running balance for the chequing account. It should look similar to the following:

Monthly Statement as of 30/6/2018Statement for John Smith, Account #123-456

01/06/2018 Opening Balance $ 900.0004/06/2018 Deposit $ 55.55 $ 955.5512/06/2018 Cheque $ 10.50 $ 945.0512/06/2018 Service Charge $ 3.50 $ 941.5515/06/2018 Withdrawal $ 70.00 $ 871.5530/06/2018 Interest $ 0.23 $ 871.78

To generate this report, you will need to use the ArrayList for the transaction and be able to determine the running balance.

1. Completing the process

After all the above is completed, the system should exit without saving any data into files.

1. Ending the session

At the end of any session, the data contained in the ArrayList as well as the account should be saved into files. Thus, methods should be created in the processor class to perform these.

Note that these files are going to be used whenever a new session is started during the statement period as mentioned in the section on loading the data.

If necessary, add methods to the Account class and Transaction class.

1. Use private helper methods to break up the large tasks and make the program more modular.

**The Client Class:**

You will receive the starting point for the Client Interface class. It contains the menu system shown below:

MAIN MENU:

1) Account information inquiry

2) Customer menu

3) Print transactions

4) End of Period Processing

5) Exit

CUSTOMER MENU:

1. Depositing to the account
2. Withdrawing funds from the account

3) Writing a cheque

4) Return to Main Menu

If Option 2 of the main menu is chosen, the Customer Menu is shown and the user is requested to enter his choice. The systems stays in this sub menu until Option 4 is chosen at which time, the system returns to the main menu.

You will need to complete the rest of this class. Note that this class only interacts with the Processor class and calls the Processor class methods. It may be necessary to add more methods to the Processor class than those mentioned above. It will also accept inputs from the user.

**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 order to save time, there is no need to document the Client Interface class - document only the Processor class. The Account class and Transaction class have been documented in phase 1.

**Testing**

As you develop the Processor class, you should test each method after you have completed it. Use your own test data to do this. It is imperative that the Processor class is thoroughly tested before the Client Interface class is developed. You may want to use a test client to do your testing and remove it in you final submission.

**Development Schedule**

This is a large, more complex program than you have done so far. Expect it to take longer that you might anticipate. Start this right away, and work on it over the entire time given you.

Develop your solution one bit at a time – never design, code and test a large program in its entirety. As you write each new method, test it, or by writing a test driver, or by calling one of the Processor class methods from the Client Interface.

A simple suggestion might be to delay the reading of the files to the end. Hardcode data for the account and use this to process transactions. Use your own test data. When all your other methods are working to your satisfaction, you can save the data into the file and read the same data from the file and load them to the appropriate instance variables. This way, you do not get hung up on working with files and sacrifice the rest of the program. Writing to the file should be the last process you should develop and test.

**Submission**

Submit the entire Java folder to the submit drive.