CS 1713 Fall 2011 Project 3: Banking System

This is the third major project that you are required to complete in this class. You MUST design and code this project on your own. You can ask for debugging help, particularly from the tutors, TAs, or instructor, but this project is to be your own work. If you have a question about this project other than simple debugging, the only one who can help you is your instructor. The project relies on knowledge and skills you are developing on Strings, array, ArrayList, class hierarchy, and file I/O.

This project is due on December 6.

Overview
In this project, you will develop a simple Banking System to manage customers’ account information associated with a bank. The Banking System shall support regular banking operations including withdrawal money, deposit check or cash, and transaction tracking. Additionally, the Banking System should facilitate the addition and deletion of Customer accounts, and provide customers with activity history for various accounts, such as checking account and saving account.

The purpose of this project is to give you experience designing and implementing a software system using multiple classes, ArrayList and arrays of objects, and search and sorting algorithms.

Part 1 (6 points)
In this part you will create four classes, the BankAccount class, the Customer class, the Transaction class, and the BankTester class.

The BankAccount class is an abstract class, storing the information about Customer(s), the account number, the date that the account is opened, the interest rate, and the account balance. The Customer class stores customer’s information including his or her name, address, social security number, and phone numbers. You will implement constructors for the BankAccount and Customer classes to initialize their attributes. You will implement the accessor methods for the respective attributes in both classes as well. Additionally, you will implement the toString() methods for both classes to display the objects appropriately.

The Customer class has additional methods called setAddress and setPhone that update the customer address and phone numbers respectively.

The BankAccount class has a method called deposit that creates a credit transaction, and updates the account balances with the amount of money the customer deposits. The BankAccount has a method called setInterest that sets the interest rate. The BankAccount class has a method to add a customer; at most five customers can be associated with one bank account. Additionally, the BankAccount class has an abstract method called calculateInterest that computes the interest.

The Transaction class stores the information about a BankAccount’s transaction, including the date, the type (can be either credit or debit), and the amount. You will implement a constructor to initialize all the attributes. You will implement the accessor methods for the respective attributes. You will implement the toString() method to display a Transaction object appropriately.
The BankTester class has a main method to test the other classes. The program should start by printing a line in the form:

BankTester Written by YourName

This line should include your full name. Add code to test the methods of the three classes you implemented thoroughly.

Part 2 (4 points)
In this part, you will create two classes, the CheckingAccount and the SavingAccount class. Both the CheckingAccount and SavingAccount Classes are derived from the BankAccount class. They both implement the calculateInterest method. Each class must have a constructor and the toString method.

The CheckingAccount has a method called withdraw that creates a debit transaction and updates the account balances with the amount of money the customer withdraws.

The SavingAccount has a method called transfer that allows a customer to transfer money from his or her SavingAccount to his or her CheckingAccount. The transfer method will create two transactions.

Add code to test the methods of the two classes you implemented thoroughly.

Part 3 (6 points)
In this part, you will create a static class called Bank that stores the information of all the BankAccount objects the bank manages and all the Transaction objects. (The Bank class is static in the sense that all of its attributes and methods are static.) For simplicity, you can assume that the BankAccount objects are held in an array of a fixed size of 100. The Bank class has a method to create an account for a customer. The Bank class has a method to find a specific bank account using the account number. Additionally, the Bank class has a method to delete a customer’s account. After the deletion, you have to perform a maintenance operation to ensure all the accounts in the array are consecutive.

The Bank class uses an ArrayList to hold all the transactions that have been created so far. Construct a method to sort the transaction objects in the ArrayList by the account number, breaking the tie using the date on which a transaction happened. Using binary search to find all the transactions happened on a specific date.

Add code to the BankTester class to test all the methods of the Bank class. In particular, create several instances of BankAccounts and put them in an array. Also, create some ArrayList objects, storing instances of the Transaction class, including one of size 0, one of size 10, and one of size 100.

Part 4 (4 points)
In this part, you will create a text file called transactionList that stores a list of Transactions have been created. When the methods deposit, withdraw, and transfer create transactions, the file will be opened for output, one transaction per line. A transaction is formatted as a String, containing the account number, the date, the type, and the amount.
Add a method called `printActivity` to the `Bank` class that retrieves a specific bank account’s activities during the current month and prints all of them in the ascending order by date. Add code to the `BankTester` class to test this part thoroughly.

**Hand-In Requirements (Deliverables):**

This project must be done using eclipse. The project name should be your login name followed by the digit 1, all lower case. For example, if your name is John Doe your project name might be `jDoe1`. Put all of your classes in a package called `pr03`. Implement and test your `Classes` in stages. As you implement a feature, add code to the `BankTester` to test the feature just added. Keep all of your test code.

You will submit a zip file of your project directory electronically through Blackboard.