Marked lab exerises – Exercise 3

Lahcen Ouarbya

December 13, 2018

1 Objectives

The objectives of this assignment is to understand the concept of objects as well as inheritance in Java. You required to develop a simple bank application. you are given two two java classes ***BankACcount*** and ***SavingAccount***. There are two classes to implement: ***CheckingAccount*** and ***Bank*** classes. Both ***SavingAccount*** and ***CheckingAccount*** are subclasses of the ***BankACcount*** class.

CheckingAccount class

CheckingAccount **extends** Account. No interest is given by the bank to this type of account. Withdrawal and deposit via ATM are charged or electronic transactions are charged with a fee of £1 per transaction. The account balance can't be less than £0 using ATM or electronic transaction and the minimum amount that can be deposited is also £1. We can assume the deposited amount is always greater or equal to £1.
Withdrawal using checks is alway allowed. the first 3 checks use in a month are free of charge, however, any subsequent uses of check are charge a £2 fee for each withdrawal. the balance of a check withdrawal can't be less than -£10(£10 overdraft).
The **CheckingAccount** class contains the following:

1. *numberOfChecksUsed* stores the number of checks used every month. Starts at zero.

2. *numberOfTransactions* stores the number of checks used every month. Starts at zero.

3. ***public CheckingAccount(String id, double initialBalance)***: We assume that the initial balance won't be negative. You are required to use the keyword super.

4. ***withdraw(double amount)***: Implement the withdraw method via ATMs to take out the provided amount from the account balance. Incorporate the transaction fee for ATM/electronic withdrawals. A withdrawal that potentially lowers the balance below £0 is not allowed. In such cases the balance remains unchanged but the method returns false. The method returns true if the withdrawal is successful.

5. ***deposit(double amount)***: The provided amount is added to the account as a result of an ATM transaction. Applicable transaction fees are deducted from the balance.

6. ***resetChecksUsed()***. This method resets the numberOfChecksUsed to zero.

a. ***withdrawUsingCheck(double amount)***: This method allows one to use checks to withdraw cash. It updates the balance according to the rules described above. If the balance could fall below - £10 because of the amount and/or fees, the method doesn't change the balance, does not increment the number of checks used but just returns false. A successful withdrawal results in updating the balance and number of checks used, and the method returns true.

Bank class

The bank class keeps information on all its accounts in an ArrayList. The bank class can retrieve a particular account using a account ID and perform deposits and withdrawals. It can perform monthly maintenance tasks such as resetting the number of checks in CheckingAccounts and adding interest in SavingsAccounts. The bank class should contain the following

1. *name* of type string the store the name of the bank.
2. ArrayList *accounts:* to store the accounts
3. double *savingsInterestRate:* /stores eh rate of interest for SavingsAccount in percentage
4. ***public Bank(String name, double savingInterestRate)***: (constructor to initialise the extra instance variables.)
5. ***getAccount(String accountID)***: returns the first Account object corresponding to given accountID and returns null If the specified account does not exist.
6. ***deposit(String accountID, double amount)***: deposit provided amount into the account specified by accountID. If account is not found, then return false. Otherwise make the deposit and return true.
7. ***withdraw(String accountID, double amount)***: withdraw provided amount from the account specified by accountID. Returns true if successful, false otherwise. the list of accounts has to be updated.
8. ***transfer(String fromAccountID, String toAccountID, double amount)***: withdraws from provided fromAccountID and deposit to specified toAccountID. if the transaction is successful it returns true and otherwise). it returns true. The transaction fees corresponding to concrete implementations of withdraw and deposit will apply.
9. ***numberOfCheckingAccounts()***: iterates through all accounts and returns the number of checking accounts.
10. ***public void addInterest()***: iterates through all accounts and add interest to the eligible accounts (i.e., all SavingsAccounts) using the rate set in the class.
11. ***public void reset()***: iterates through all accounts and reset the number of checks for applicable accounts (i.e., all CheckingAccounts).
12. Add the main method to test your class. the body of this method should contain the code to test your code. You can put this method is a separate class if you wish.