CS 2336.004 Spring 2017

Assignment 1 – Class & Object Practice: Banking Program

Jan 30th, 2017
Due Date: Feb 12th, 2017 – 11:59pm

This banking system is to be created for this assignment. The program will consist of five(5) classes: MustangBanking (which will contain the main method), Account, CheckingAccount, SavingsAccount and Balance.

Both the CheckingAccount and SavingsAccount classes will inherit from class Account, and they will both have a data member of type Balance.

Within method main, which is within class MustangBanking, create an ArrayList of type Account. You may call this ArrayList accounts. Within a loop, display the messages "Welcome to your Mustang Bank!" and "Please choose from the following options:". These options will be:

1 – Create a new Checking Account
2 – Create a new Savings Account
3 – Delete an existing account
4 – View a specific account
5 – View all accounts
6 – Write a check through a specific Checking Account
7 – Withdraw funds from a specific account
8 – Deposit funds into an account
9 – Exit Program
Please enter your option below:

For option 1, call the method xCreateChecking, sending accounts. Upon returning, print out a statement that the Checking Account has been created.
For option 2, call the method xCreteSavings, sending accounts. Upon returning, print out a statement that the Savings Account has been created.
For option 3, call the method xDeleteAccount, sending accounts.
For option 4, call the method xViewSpecific, sending accounts.
For option 5, call the method xViewAll, sending accounts.
For option 6, call the method xWriteCheck, sending accounts.
For option 7, call the method xWithdraw, sending accounts.
For option 8, call the method xDeposit, sending accounts.
Option 9 should have the loop end and the program exit.

Classes:
The Account class will consist of:
iId (integer)
dInterestRate (double)
bBalance (Balance) – Balance is a separate class
With methods:
Constructor Account, which takes in the Id, Interest Rate and Balance amounts.

getId
setId
getInterestRate
setInterestRate
deposit - which takes in an amount and adds that to the current balance
withdraw – takes in an amount and subtracts it from the current balance
  NOTE: If funds do not exist for the withdraw, a message must be displayed that funds are not available.
displayAccount – Displays a message showing the account Id, Balance and Interest Rate

The CheckingAccount class extends, or inherits from class Account. In addition, It has:
Constructor CheckingAccount that takes in the ID, Balance and Interest Rate
  (recall, it must then call the super class with the command super() sending it the amounts as well)
It also has the methods:
xWriteCheck – which accepts the account ID
  It will first loop through the accounts to determin if the given account exists
  If so, will prompt the user for the check amount and will withdraw the check amount from the account
  If the account DOES NOT exist, a message must be displayed indicating the given ID does not exist.

The SavingsAccount class extends the Account class as well. In addition, it has:
Constructor SavingsAccount – Takes in the ID, Balance and Interest Rate
  (It too must then call the super class)
Withdraw – which takes in an amount to withdraw
  If funds are available, the received amount is subtracted from the Balance
  Otherwise, a message is displayed indicating funds are not available.

The Balance class will consist of:
dBalance (double)
with methods:
Constructor Balance – Have a default constructor which sets the dBalance to 0.0
  And a second constructor Balance which takes in an amount that Balance is set to.
getBalance
setBalance

PLEASE NOTE: During some of the method processing, you will want to be sure the account you are checking is either a savings or checking account. This can be done with the 'instanceof' operator.
For example:
If (accounts.get(i) instanceof SavingsAccount) { //-- this is a savings account }

In addition, please draw/write-out a UML diagram for this program.

NOTE: This specified requirements are written in accommodation of JAVA. If you prefer C++, please modify accordingly (e.g. use C++ array, no need for MustangBanking class).

Deliverables:
Turn in your code file(s) (.java or .cpp) and UML diagram file (e.g. pdf).