Assignment 2 - Bank Foundation Enhancements

Introduction

Merit America Bank is pleased with the current progress with the software system and has some additional requirements they would like you to implement. The additional requirements might require new code or amending previously written code.

In this assignment, you will be working in a pair with a fellow cohort colleague using Pair Programming.

Requirements

Merit America Bank would like to be able to keep track of its account holders and would also like to be able to offer a new type of account called a Certificate of Deposit, or CD. They would also like to increase the ability for account holders to have unlimited checking and savings accounts, with some conditions. Finally, all accounts should have a unique account number.

Merit America Bank will offer various CDs:

1. 1 year term at 1.8%
2. 2 year term at 1.9%
3. 3 year term at 2.0%
4. 5 year term at 2.5%
5. 10 year term at 2.2%

Merit America Bank wants to be able to change their CD offerings at any time. They also want to be able to easily determine the CD offering that provides the highest future value and the second highest from the available current offerings.

Merit America Bank requires an Object Oriented Design for this software system. As a result, they are expecting a BankAccount superclass with CheckingAccount, SavingsAccount, and CDAccount as subclasses of BankAccount.

Merit America Bank will allow account holders to open new checking and savings accounts as long as the combined balances are under $250,000 (CD Accounts are not included in this calculation). Once an account holder has combined balances exceeding $250,000, it should no

longer be possible to open new checking or savings accounts until the combined balances are below $250,000.

At this time, Merit America Bank requires you to use arrays for any lists required.

Hint: Although Java arrays do have a maximum capacity, a new array with a larger capacity could be created and the contents of the current array can be copied into it in order to allow for storage of additional elements.

Create the following classes with the following methods:

1. MeritBank
    a. static void addAccountHolder(AccountHolder accountHolder)
    b. static AccountHolder[] getAccountHolders()
    c. static CDOffering[] getCDOfferings()
    d. static CDOffering getBestCDOffering(double depositAmount)
    e. static CDOffering getSecondBestCDOffering(double depositAmount)
    f. static void clearCDOfferings()
    g. static void setCDOfferings(CDOffering[] offerings)
    h. static long getNextAccountNumber()
    i. static double totalBalances()
    j. static double futureValue(double presentValue, double interestRate, int term)
2. CDOffering
    a. CDOffering(int term, double interestRate)
    b. int getTerm()
    c. double getInterestRate()
3. BankAccount
    a. BankAccount(double balance, double interestRate)
    b. BankAccount(long accountNumber, double balance, double interestRate)
    c. long getAccountNumber()
    d. double getBalance()
    e. double getInterestRate()
    f. boolean withdraw(double amount)
    g. boolean deposit (double amount)
    h. double futureValue(int years)
4. CDAccount
    a. CDAccount(CDOffering offering, double balance)
    b. double getBalance()
    c. double getInterestRate()
    d. int getTerm()
    e. java.util.Date getStartDate()
    f. long getAccountNumber()
    g. double futureValue()

Amend the following classes to provide the following methods:

1. AccountHolder
    a. AccountHolder(String firstName, String middleName, String lastName, String ssn)
    b. String getFirstName()
    c. void setFirstName()
    d. String getMiddleName()
    e. void setMiddleName()
    f. String getLastName()
    g. void setLastName()
    h. String getSSN()
    i. void setSSN()
    j. CheckingAccount addCheckingAccount(double openingBalance)
    k. CheckingAccount addCheckingAccount(CheckingAccount checkingAccount)
    l. CheckingAccount[] getCheckingAccounts()
    m. int getNumberOfCheckingAccounts()
    n. double getCheckingBalance()
    o. SavingsAccount addSavingsAccount(double openingBalance)
    p. SavingsAccount addSavingsAccount(SavingsAccount savingsAccount)
    q. SavingsAccount[] getSavingsAccounts()
    r. int getNumberOfSavingsAccounts()
    s. double getSavingsBalance()
    t. CDAccount addCDAccount(CDOffering offering, double openingBalance)
    u. CDAccount addCDAccount(CDAccount cdAccount)
    v. CDAccount[] getCDAccounts()
    w. int getNumberOfCDAccounts()
    x. double getCDBalance()
    y. double getCombinedBalance()
2. CheckingAccount
    a. Extend from BankAccount
3. SavingsAccount
    a. Extend from BankAccount
4. MeritAmericaBankApp
    a. public static void main(String[] args)
        i. Add 5 CDOfferings to MeritBank
        ii. Instantiate a new AccountHolder (ah1)
        iii. Add a checking account with an opening balance of $1,000 as well as a savings account with an opening balance of $10,000 to ah1
        iv. Add a checking account with an opening balance of $5,000 as well as a savings account with an opening balance of $50,000 to ah1
        v. Add a checking account with an opening balance of $50,000 as well as a savings account with an opening balance of $500,000 to ah1

vi. Add a checking account with an opening balance of $5,000 as well as a savings account with an opening balance of $50,000 to ah1
vii. Confirm last checking and savings accounts were not created
viii. Add the best CD offering as a CD account on ah1
ix. Add ah1 to Merit Bank's list of account holders
x. Instantiate a new AccountHolder (ah2)
xi. Add a checking account with an opening balance of $1,000 as well as a savings account with an opening balance of $10,000 to ah2
xii. Add the second best CD offering as a CD account on ah2
xiii. Add ah2 to Merit Bank's list of account holders
xiv. Clear the CDs being offered by MeritBank
xv. Instantiate a new AccountHolder (ah3)
xvi. Add the second best CD offering as a CD account on ah3
xvii. Confirm a CD account was not created on ah3
xviii. Add a checking account with an opening balance of $1,000 as well as a savings account with an opening balance of $10,000 to ah3
xix. Add ah3 to Merit Bank's list of account holders
xx. Get the total balance of all accounts held by Merit Bank's account holders

Instructions

1. Visit this GitHub repository and follow the instructions to work with the provided starter code:
    a. Fork the repository
    b. Clone your fork (consider adding your partner as a collaborator to your forked repository)
    c. Import the project into Eclipse
    d. Run the application in Eclipse
    e. Run the test cases in Eclipse
2. Complete the assignment requirements such that all test cases are passing.
    a. Note: you may copy your existing files from Assignment 1 as a starting point.
3. Upload your zipped 'assignment2' folder to HackerRank for submission.

Expectations

1. Code should be readable
    a. For example: use meaningful variable names, use proper naming conventions, properly indent code, comment your code
2. Use the "this" keyword to reference instance variables/methods
3. Use of java.util.List interface and implementing classes is not permitted
4. Use of java.util.Arrays class is not permitted