Assignment 3 - Bank Data Persisted to File

# 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 noticed that when the program stops and restarts, all of the data disappears. They require the ability to save all of the following details to a file and it should be possible to read this same information from a file:

1. MeritBank
   1. List of Account holders
   2. List of CD Offerings
   3. Next account number
2. CDOffering
   1. Description
   2. Term
   3. Interest Rate
3. AccountHolder
   1. Account holder details
   2. List of Checking Accounts
   3. List of Savings Accounts
   4. List of CD Accounts
4. CheckingAccount
   1. Account number
   2. Current Balance
   3. Interest Rate
   4. Date account opened
5. SavingsAccount
   1. Account number
   2. Current Balance
   3. Interest Rate
   4. Date account opened
6. CDAccount
   1. Account number
   2. Opening Balance
   3. Term
   4. Interest Rate
   5. Date account opened

Note: when reading from file, the data should overwrite the MeritBank data such that previous data no longer exists, only the data read from the file should exist.

Note: we will not be using Java Serialization since Merit America Bank wants other (possibly non-Java) applications to be able to parse this file.

Finally, after reading the file, Merit America Bank would like the list of account holders displayed in ascending order of their total balances.

## Details

Amend the following class:

* class BankAccount
  + BankAccount(double balance, double interestRate)
  + BankAccount(double balance, double interestRate, java.util.Date accountOpenedOn)
  + BankAccount(long accountNumber, double balance, double interestRate, java.util.Date accountOpenedOn)
  + java.util.Date getOpenedOn()
  + static BankAccount readFromString(String accountData) throws ParseException
    - Should throw a java.lang.NumberFormatException if String cannot be correctly parsed
  + String writeToString()

The following classes should now extend the BankAccount class:

* CheckingAccount
  + Create appropriate constructors
  + static CheckingAccount readFromString(String accountData) throws ParseException
    - Should throw a java.lang.NumberFormatException if String cannot be correctly parsed
* SavingsAccount
  + Create appropriate constructors
  + static SavingsAccount readFromString(String accountData) throws ParseException
    - Should throw a java.lang.NumberFormatException if String cannot be correctly parsed
* CDAccount
  + Create appropriate constructors
  + Override the deposit and withdraw methods to return false (CD Accounts cannot deposit new funds or withdraw funds until the term is reached)
  + static CDAccount readFromString(String accountData) throws ParseException
    - Should throw a java.lang.NumberFormatException if String cannot be correctly parsed
  + Override writeToString method to include term

Amend the following class:

* class AccountHolder implements Comparable<AccountHolder>
  + Implement the compareTo(AccountHolder otherAccountHolder) method such that account holders can be sorted by the combined balance of their accounts
  + String writeToString()
  + static AccountHolder readFromString(String accountHolderData) throws Exception
    - Should throw a java.lang.Exception if String cannot be correctly parsed

Add the following methods to the MeritBank class:

* static boolean readFromFile(String fileName)
* static boolean writeToFile(String fileName)
* static AccountHolder[] sortAccountHolders()
* static void setNextAccountNumber(long nextAccountNumber)

Add the following methods to the CDOffering class:

* static CDOffering readFromString(String cdOfferingDataString)
  + Should throw a java.lang.NumberFormatException if String cannot be correctly parsed
* String writeToString()

The following file describes Merit America Bank with:

1. Next bank account number: 11
2. 3 CD offerings
3. 2 account holders
   1. John Doe
      1. 1 checking account
      2. 2 savings accounts
   2. Jane S Doe
      1. 2 checking accounts
      2. 3 savings accounts
      3. 2 CD accounts

This is an example data file for the scenario described above:

11

3

1,0.018

3,0.019

5,0.02

2

Doe,,John,1234567890

1

1,1000,0.0001,01/01/2020

2

2,10000,0.01,01/02/2020

3,150000,0.01,01/02/2020

0

Doe,S,Jane,0987654321

2

4,100,0.0001,12/01/2019

5,200,0.0001,12/15/2019

3

6,15000,0.01,12/01/2019

7,2100,0.01,12/15/2019

8,9000,0.01,01/01/2020

2

9,5000,0.02,01/01/2020,5

10,5000,0.025,01/01/2020,10

Explanation:

|  |  |
| --- | --- |
| 11 | Next account number is 11 |
| 3 | 3 CD offerings |
| 1,0.018 | 1st CD offering: 1 year term at 1.8% |
| 3,0.019 | 2nd CD offering: 3 year term at 1.9% |
| 5,0.02 | 3rd CD offering: 5 year term at 2.0% |
| 2 | 2 account holders |
| Doe,,John,1234567890 | 1st account holder: John Doe (no middle name) with SSN: 1234567890 |
| 1 | 1st account holder has 1 checking account |
| 1,1000,0.0001,01/01/2020 | Checking account #1 has $1,000 balance, 0.01% interest rate, and was opened on 1/1/2020 |
| 2 | 1st account holder has 2 savings accounts |
| 2,10000,0.01,01/02/2020 | 1st savings account #2, has $10,000 balance, 1% interest rate, and was opened on 1/2/2020 |
| 3,150000,0.01,01/02/2020 | 2nd savings account #3, has $150,000 balance, 1% interest rate, and was opened on 1/2/2020 |
| 0 | 1st account holder does not have any CD accounts |
| Doe,S,Jane,0987654321 | 2nd account holder: Jane S Doe with SSN 0987654321 |
| 2 | 2nd account holder has 2 checking accounts |
| 4,100,0.0001,12/01/2019 | 1st checking account #4, current balance of $100 with 0.01% interest rate opened on 12/1/2019 |
| 5,200,0.0001,12/15/2019 | 2nd checking account #5, current balance of $200 with 0.01% interest rate opened on 12/15/2019 |
| 3 | 2nd account holder has 3 savings accounts |
| 6,15000,0.01,12/01/2019 | 1st savings account #6, current balance of $15,000 with 1% interest rate opened on 12/1/2019 |
| 7,2100,0.01,12/15/2019 | 2nd savings account #67, current balance of $2,100 with 1% interest rate opened on 12/15/2019 |
| 8,9000,0.01,01/01/2020 | 3rd savings account #8, current balance of $9,000 with 1% interest rate opened on 1/1/2020 |
| 2 | 2nd account holder has 2 CD accounts |
| 9,5000,0.02,01/01/2020,5 | 1st CD account #9, opening balance of $5,000 with 2% interest rate and a 5 year term, opened on 1/1/2020 |
| 10,5000,0.025,01/01/2020,10 | 2nd CD account #10, opening balance of $5,000 with 2.5% interest rate and a 10 year term, opened on 1/1/2020 |

# Instructions

1. Visit this [GitHub repository](https://github.com/MeritAmerica/assignment3) and follow the instructions to work with the provided starter code:
   1. Fork the repository
   2. Clone your fork (consider adding your partner as a collaborator to your forked repository)
   3. Import the project into Eclipse
   4. Run the application in Eclipse
   5. Run the test cases in Eclipse
2. Complete the assignment requirements such that all test cases are passing.
   1. Note: you may copy your existing files from Assignment 2 as a starting point.
3. Upload your zipped ‘assignment3’ folder to [HackerRank](https://www.hackerrank.com/tests/9gijn2ci62c/42c9d24a2b1f9053efe23674fe242bd1?try_test=true) for submission.

# Expectations

1. Functionality from prior assignments should still work except where amended for this assignment’s requirements
2. Write your own unit tests for all of the requirements in this assignment (think about edge cases)
3. Code should be readable
   1. For example: use meaningful variable names, use proper naming conventions, properly indent code, comment your code
4. Use the “this” keyword to reference instance variables/methods
5. Utilize exception handling (try/catch/throws) to properly handle errors