CSE 101 Programming Assignment 4

Due:

Friday, 13-December-2024 by 23:59

Deliverables:

The following Java file should be submitted to Moodle by the due date and time specified above. Submissions received after the deadline will be subject to the late policy described in the syllabus.

- Assignment4.java

Specifications:

1. Login to Moodle at https://lms.akdeniz.edu.tr/
2. Login with your Akdeniz student email account
3. My Courses > CSE101T Mühendislik > Assignment 4
4. **Your filename must be Assignment4.java, otherwise you cannot upload it.**
5. (Optional) You can download empty Assignment4 file from download link.
6. (Optional) You can provide your implementation on the **Edit** tab
7. From the **Submission** tab you can upload your file. **Read step 4 again.**
8. Go to the **Edit** tab and press save button (the one next to +) or use command 'Ctrl + s' to save the document
9. Go to the **Submission view** tab and check if you can see your document
10. Done.

Overview: You will continue your program this week to maintain the account balances for bank customers. Do not forget your headers with @author and @version information. This is the last program this term; however, you should fully understand the concepts covered in all these assignments prior to beginning the next semester CSE 102T course.

Requirements: Write a program that will perform the following tasks:

1. Get accounts information
    a. This will involve reading a file of account information from a file.
2. Make transfers between accounts
    a. A series of transfers will be requested (also through a file)
3. Create output file of accounts
    a. After all transfers are completed, write to an output file the new account information in the same format as the source file
4. Create a log file
    a. A log should note the requested transfer and whether it was successful

To facilitate the execution of this program, you will add and modify (at minimum) the following methods:

1. countAccounts(filename)
    a. A method to determine how many accounts are in a file using a filename
    b. Takes a String as a parameter representing the filename
    c. Displays nothing

   d. Returns an integer that represents the number of accounts in the file (Protip: this will be the number of lines in the file)
2. readAccountInfo(acctNums, names, surnames, balances, filename)
   a. A method to read the contents of the file into the various necessary arrays
   b. It will take five parameters
      i. Integer array for account numbers
      ii. String array for account owner names
      iii. String array for account owner surnames
      iv. Decimal array for account balances
      v. String for the filename
      vi. Each array passed should be the size of the number of accounts in the file
   c. It will read the file to get the information and place it into the arrays
      i. The file format will be one account per line
      ii. Each line will consist of an account number, space, Name (no spaces), space, Surname (no spaces), space, the current balance
      iii. Example below:
   d. Protip: you can use the return value from the countAccounts() method to set the size of the arrays before calling this method
   e. Returns None
3. deposit(balances, index, amount)
   a. A method to make a deposit to the balances array at the given index
   b. Returns True if the deposit is successful, false otherwise
   c. If the deposit amount is valid, the balances array value at the index given should be increased by the amount passed
   d. If the deposit amount is not valid, the array should not be changed
   e. Note: this is not the same as the isDepositValid() method, but can make use of that method
4. withdrawal(balances, index, amount)
   a. A method to make a withdrawal from the balances array at the index
   b. Returns True if the withdrawal is successful, false otherwise
   c. If the withdrawal amount is valid, the balances array value at the index given should be decreased by the amount passed
   d. If the withdrawal amount is not valid, the array should not be changed
   e. Note: this is not the same as the isWithdrawalValid() method, but can make use of that method
5. transfer(acctNums, balances, acctNumFrom, acctNumTo, amount)
   a. A method to make a transfer from one account to another
   b. Will update the balances array passed by removing funds from the acctNumFrom account and adding the funds to the acctNumTo account
   c. Returns an integer with a code to indicate if a transfer was successful
      i. 0 for successful transfer
      ii. 1 if the amount cannot be withdrawn from the from account
      iii. 2 if the acctNumFrom was not found in the acctNums array

iv. 3 if the acctNumTo was not found in the acctNums array

d. If the transfer cannot be completed (i.e. does not return a code of 0), none of the balances should be changed
e. Except for the final code (3), bill pay codes should be the same code/message type

6. writeAccountInfo(acctNums, names, surnames, balances, filename)
    a. A method to write the contents of the arrays to an output file
    b. It will take five parameters
        i. Same as readAccountInfo()
    c. It will write a new file using the information contained in the arrays with the same format as the source file for readAccountInfo()
    d. Returns None
7. findAcct(acctNums, acctNum)
    a. No change from previous assignment
8. isDepositValid(amount)
    a. No change from previous assignment
9. isWithdrawalValid(balance, amount)
    a. No change from previous assignment
10. Any other methods you feel helpful can be implemented, however, these will be the only methods tested.

Design: To execute this program, the **main** method will be used by passing the base filename as a command line argument. Example text files are included in the class materials folder.

java Assignment4 Assignment4

The account information will be contained in a file called {basefilename}_AccountInfo.txt It will have the format described above. The transfer information will be contained in a file called {basefilename}_TransferInfo.txt It will have the format of transfer number (String), space, acctNumFrom, space, acctNumTo, space, amount. The Bill Pay information will be contained in a file called {basefilename}_BillPay.txt
It will have the format of Bill pay number (String), space, acctNumFrom, space, Bill type (String with no spaces), space, amount. The output account information will be written to a file called {basefilename}_AccountInfoOut.txt It will have the same format as the source account information. The log file will be written to a file called {basefilename}.log It will have the format of “Transfer “, transfer number, “ resulted in code “, result code, “: “, description. Example shown:

Code: Your code should take the command line argument for the filename and perform the tasks described in the requirements. Prior to running your program, the folder containing your java file should look like the following:

Name	Size	Type	Date Modified
Assignment4_TransferInfo.txt	66 bytes	plain text document	Today
Assignment4_BillPay.txt	53 bytes	plain text document	Today
Assignment4_AccountInfo.txt	136 bytes	plain text document	Today
Assignment4.java	8.6 KiB	Java source code	Today
Assignment4.class	5.1 KiB	Java class	Today

Once your program is executed, the folder should then look like the following:

Name	Size	Type	Date Modified
Assignment4_TransferInfo.txt	66 bytes	plain text document	Today
Assignment4_BillPay.txt	53 bytes	plain text document	Today
Assignment4_AccountInfoOut.txt	132 bytes	plain text document	Today
Assignment4_AccountInfo.txt	136 bytes	plain text document	Today
Assignment4.java	8.6 KiB	Java source code	Today
Assignment4.class	5.1 KiB	Java class	Today
Assignment4.log	303 bytes	application log	Today

Test: You are responsible for testing your program. It is important to not rely solely on the examples presented in this Assignment description.

Grading:
Moodle Submission: You can submit multiple times, however, we will only grade the last version that you submitted.

NOTE: If you use System.exit() in your code, you will automatically **receive 0 points** for this assignment.

Examples

Running the code below (with the declaration statements replaced with blanks) resulted in the following output and text file.

```
public class ATMExamples {
   public static void main(String[] args) throws Exception {
      ________ acctNums = ________;
      ________ acctNames = ________;
      ________ acctSurnames = ________;
      ________ acctBalances = ________;
      ________ numOfAccounts;

      numOfAccounts = ATM_123456789.countAccounts("Assignment4_AccountInfo.txt");
      System.out.println("The number of accounts is " + numOfAccounts);

      ATM_123456789.readAccountInfo(acctNums, acctNames, acctSurnames, acctBalances, "Assignment4_AccountInfo.txt");
      System.out.println("The information in the file is:");
      System.out.println("Number\tName\tBalance");
      for(int i = 0; i < acctNums.length; i++)
         System.out.println(acctNums[i] + "\t" + acctNames[i] + "\t" + acctSurnames[i] + "\t" + acctBalances[i]);

      System.out.println("The deposit is successful: " + ATM_123456789.deposit(acctBalances, 0, 100));
      System.out.println("The new balance for the first account is " + acctBalances[0]);

      System.out.println("The withdrawal is successful: " + ATM_123456789.withdrawal(acctBalances, 1, 100));
      System.out.println("The new balance for the second account is " + acctBalances[1]);

      System.out.println("The return value of transferring 150 from the first account to the second is : " +
                         ATM_123456789.transfer(acctNums, acctBalances, 12345, 67890, 150));
      System.out.println("The new balance for the first account is " + acctBalances[0]);
      System.out.println("The new balance for the second account is " + acctBalances[1]);

      System.out.println("The return value of transferring 150 from the last account to the second is : " +
                         ATM_123456789.transfer(acctNums, acctBalances, 98765, 67890, 150));
      System.out.println("The new balance for the last account is " + acctBalances[5]);
      System.out.println("The new balance for the second account is " + acctBalances[1]);

      ATM_123456789.writeAccountInfo(acctNums, acctNames, acctSurnames, acctBalances, "Assignment4_AccountInfoOut.txt");

   }
}
```

```
 ----jGRASP exec: java ATMExamples
The number of accounts is 6
The information in the file is:
Number   Name  Balance
12345 John  Doe   2000.0
67890 Can   Do 1500.0
24680 Jane  Doe   1000.0
54321 Ceyn  Do 500.0
43210 Mert  Bayrak   150.0
98765 Matt  Byerock  15.0
The deposit is successful: true
The new balance for the first account is 2100.0
The withdrawal is successful: true
The new balance for the second account is 1400.0
The return value of transferring 150 from the first account to the second is : 0
The new balance for the first account is 1950.0
The new balance for the second account is 1550.0
The return value of transferring 150 from the last account to the second is : 3
The new balance for the last account is 15.0
The new balance for the second account is 1550.0

 ----jGRASP: operation complete.
```