

Due:

Friday, 24-December-2021 by 23:59

Deliverables:

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

- ATM_{StudentNumber}.java

Specifications:

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 102 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 acctNumTo was not found in the acctNums array
 - iii. 2 if the acctNumFrom was not found in the acctNums array
 - iv. 3 if the amount cannot be withdrawn from the from account
 - d. If the transfer cannot be completed (i.e. does not return a code of 0), none of the balances should be changed
- 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.

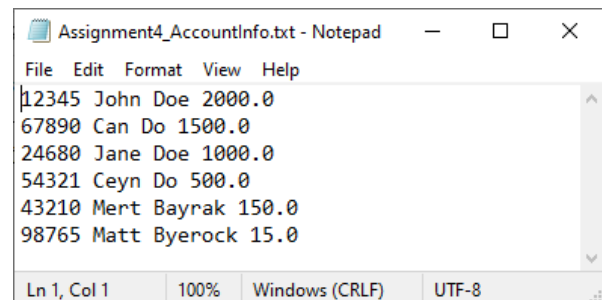
```
\Assignment4>java ATM_123456789 Assignment4
\Assignment4>
```

The account information will be contained in a file called

{basefilename}_AccountInfo.txt

It will have the format described above.

Example shown:



Assignment4_AccountInfo.txt - Notepad

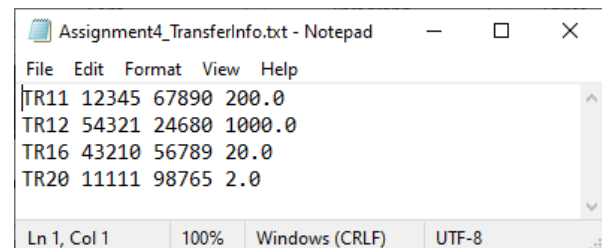
| File | Edit | Format | View | Help |
|-------|------|---------|--------|------|
| 12345 | John | Doe | 2000.0 | |
| 67890 | Can | Do | 1500.0 | |
| 24680 | Jane | Doe | 1000.0 | |
| 54321 | Ceyn | Do | 500.0 | |
| 43210 | Mert | Bayrak | 150.0 | |
| 98765 | Matt | Byerock | 15.0 | |

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

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. Example shown:



Assignment4_TransferInfo.txt - Notepad

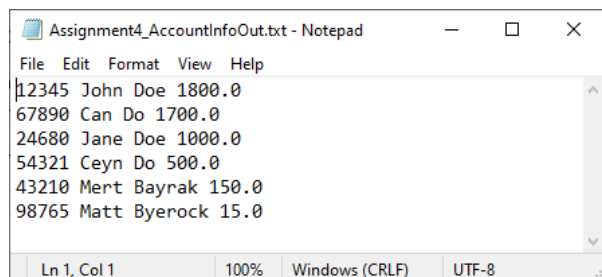
| File | Edit | Format | View | Help |
|------|-------|--------|--------|------|
| TR11 | 12345 | 67890 | 200.0 | |
| TR12 | 54321 | 24680 | 1000.0 | |
| TR16 | 43210 | 56789 | 20.0 | |
| TR20 | 11111 | 98765 | 2.0 | |

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

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.



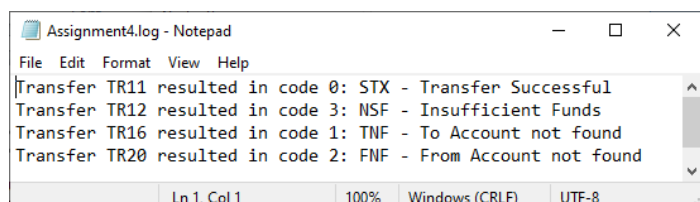
Assignment4_AccountInfoOut.txt - Notepad

| File | Edit | Format | View | Help |
|-------|------|---------|--------|------|
| 12345 | John | Doe | 1800.0 | |
| 67890 | Can | Do | 1700.0 | |
| 24680 | Jane | Doe | 1000.0 | |
| 54321 | Ceyn | Do | 500.0 | |
| 43210 | Mert | Bayrak | 150.0 | |
| 98765 | Matt | Byerock | 15.0 | |

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

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:



Assignment4.log - Notepad

| File | Edit | Format | View | Help |
|----------|------|--|------|------|
| Transfer | TR11 | resulted in code 0: STX - Transfer Successful | | |
| Transfer | TR12 | resulted in code 3: NSF - Insufficient Funds | | |
| Transfer | TR16 | resulted in code 1: TNF - To Account not found | | |
| Transfer | TR20 | resulted in code 2: FNF - From Account not found | | |

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

CSE 101 Programming Assignment 4

Page 4 of 5

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:

| Assignment4 | | | |
|------------------------------|--------------------|---------------|------|
| Name | Date modified | Type | Size |
| Assignment4_AccountInfo.txt | 12/10/2021 4:03 PM | Text Document | 1 KB |
| Assignment4_TransferInfo.txt | 12/10/2021 4:16 PM | Text Document | 1 KB |
| ATM_123456789.class | 12/10/2021 4:50 PM | CLASS File | 5 KB |
| ATM_123456789.java | 12/10/2021 4:50 PM | JAVA File | 8 KB |

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

| Assignment4 | | | |
|--------------------------------|--------------------|---------------|------|
| Name | Date modified | Type | Size |
| Assignment4.log | 12/10/2021 5:08 PM | Text Document | 1 KB |
| Assignment4_AccountInfo.txt | 12/10/2021 4:03 PM | Text Document | 1 KB |
| Assignment4_AccountInfoOut.txt | 12/10/2021 5:08 PM | Text Document | 1 KB |
| Assignment4_TransferInfo.txt | 12/10/2021 4:16 PM | Text Document | 1 KB |
| ATM_123456789.class | 12/10/2021 4:50 PM | CLASS File | 5 KB |
| ATM_123456789.java | 12/10/2021 4:50 PM | JAVA File | 8 KB |

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

Grading:

MS Teams 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.

Quiz: There will not be a quiz based on this assignment. The material contained in this assignment will be tested on the exam and will be expanded in CSE102.

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]);

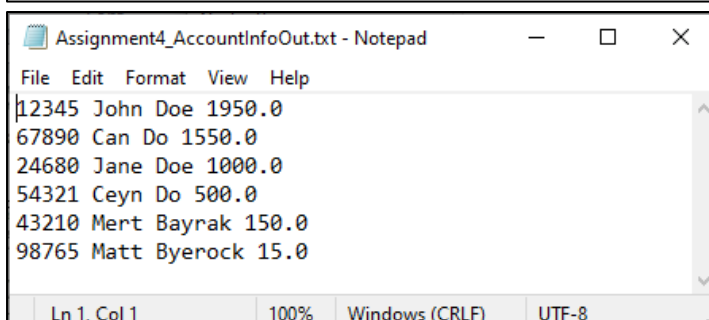
        System.out.println("The return value of transferring 150 from the first account to the second is : " +
            ATM_123456789.transfer(acctNums, acctBalances, 12345, 67890, 150));
        System.out.println("The new balance for the first account is " + acctBalances[0]);
        System.out.println("The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



```
Assignment4_AccountInfoOut.txt - Notepad
File Edit Format View Help
12345 John Doe 1950.0
67890 Can Do 1550.0
24680 Jane Doe 1000.0
54321 Ceyn Do 500.0
43210 Mert Bayrak 150.0
98765 Matt Byerock 15.0
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```