Consider an **Account** class that has:

* private instance variables:
  + int accountID
  + String name
  + double balance
* An appropriate constructor
* The following public methods:
  + public double getBalance( )
  + public int getAccountID( )
  + public String getName( )
  + public void deposit(double amount) throws IllegalArgumentException
  + public void withdraw(double amount) throws IllegalArgumentException
  + public void transfer(double amount, Account destinationAccount) throws IllegalArgumentException
  + public String toString( )
  + public boolean equals(Object obj) - comparison based on accountID only

Implement a well-structured Java program that uses the **Account** class and **AccountDriver** class to enable a bank to maintain customer accounts. The account information is kept in a text-file of the form:

|  |
| --- |
| 1007 Ahmad Said 12503.50  1004 Fatima Hassan 58000.00  1003 Suleiman Wasim 15705.50  1002 Majed Sameer 22000.00  1006 Muhammad Adel 250600.50  1005 Muhsim Zuheir 5000.50  1001 Aisha Abdullatif 27800.50 |

where each line of the text-file contains a **unique** account ID, the customer first and second names, and the current account balance in Saudi Riyals (SAR).

Your **AccountDriver** program must have the following main menu:

**1. Display Account Info for all Accounts**

**2. Display Account Info for a particular Account**

**3. Withdraw**

**4. Deposit**

**5. Transfer**

**6. Add New Account**

**7. Delete Account**

**8. Exit**

**Please select your choice:**

Your program must loop as long as option 8 has not been selected. It must display an appropriate error message if an invalid choice is entered. After executing each of the options 1 to 7, your program must pause and display the message: **“Press Enter key to continue . . .”** . Your program must display the main menu after pressing the Enter key. Each of the options 1 to 7 must be implemented in a separate static method. The code for **“Press Enter key to continue . . .”** must also be implemented is a separate static method.

The other options must have the following behaviors:

**Option 1: Display Account Info for all Accounts**

It displays the account information of all accounts. The option must be implemented by reading directly from accounts text-file. It then waits for the Enter key to be pressed before returning control to the main menu:

Please select your choice: **1**

AccountID Account Owner Account Balance (SAR)

1007 Ahmad Said 12503.50

1004 Fatima Hassan 58000.00

1003 Suleiman Wasim 15705.50

1002 Majed Sameer 22000.00

1006 Muhammad Adel 250600.50

1005 Muhsim Zuheir 5000.50

1001 Aisha Abdullatif 27800.50

Press Enter key to continue . . .

**Option 2: Display Account Info for a particular Account**

It prompts for and reads an accountID. It then searches for this accountID in the accounts text-file. If the account is not found an appropriate error message is displayed, otherwise; the account information is displayed. In both cases, the option waits for the Enter key to be pressed before returning control to the main menu.

|  |
| --- |
| Please select your choice: **2**  Enter accountID: **1002**  AccountID : 1002 Account Owner : Majed Sameer Account Balance: 22,000.00 SAR  Press Enter key to continue . . . |
| Please select your choice: **2**  Enter accountID: **1552**  Error: Invalid account number  Press Enter key to continue . . . |

**Option 3: Withdraw**

It prompts for and reads an accountID and the amount to withdraw. If the withdraw amount is zero or negative an appropriate error message is displayed, otherwise it searches for this accountID in the accounts text-file. If the accountID is not found or if the balance is not sufficient an appropriate error message is displayed, otherwise; the contents of the text-file are copied into **an array of Account objects** (count the number of lines in the text-file, and use this number as the array size). The account is updated, and the accounts text-file is then updated and the new balance is displayed:

|  |
| --- |
| Please select your choice: **3**  Please enter accountID: **1006**  Please enter withdraw amount (SAR): **2500**  Balance before withdrawal : 250,600.50 Saudi Riyals  Amount withdrawn from Account: 2,500 Saudi Riyals  New Balance: 248,100.50 Saudi Riyals  Accounts file has been updated . . .  Press Enter key to continue . . . |
| Please select your choice: **3**  Please enter accountID: **1006**  Please enter withdraw amount (SAR): **-2500**  Error: Invalid withdraw amount  Press Enter key to continue . . . |
| Please select your choice: **3**  Please enter accountID: **1006**  Please enter withdraw amount (SAR): **500250**  Error: Insufficient balance. Balance available is only 250,600.50 SAR  Press Enter key to continue . . . |

Whether the update operation was done or not, control is returned to the main menu after pressing the Enter key.

**Option 4. Deposit**

It prompts for and reads an accountID and the amount to deposit. If the deposit amount is zero or negative an appropriate error message is displayed; otherwise it searches for this accountID in the accounts text-file. If the accountID is not found an appropriate error message is displayed, otherwise; the contents of the text-file are copied into **an array of Account objects** (count the number of lines in the text-file, and use this number as the array size). The account is updated, and the accounts text-file is then updated and the new balance is displayed:

|  |
| --- |
| Please select your choice: **4**  Please enter accountID: **1001**  Please enter deposit amount (SAR): **3500**  Balance before deposit : 27,800.50 Saudi Riyals  Amount deposited to Account: 3,500 Saudi Riyals  New Balance: 31,300.50 Saudi Riyals  Accounts file has been updated . . .  Press Enter key to continue . . . |
| Please select your choice: **4**  Please enter accountID: **2225**  Please enter deposit amount (SAR): **4500**  Error: Invalid account number  Press Enter key to continue . . . |

Control is returned to the main menu after pressing the Enter key.

**Option 5: Transfer**

It prompts for and reads a source accountID, a destination accountID and the amount to be transferred. If the transfer amount is zero or negative an appropriate error message is displayed; otherwise the accounts text-file is searched for these two account IDs. If one or both accountIDs do not exist, or if the transfer amount is greater than the source accountBalance an appropriate error message is displayed; otherwise the contents of the text-file are copied into **an array of Account objects** (count the number of lines in the text-file, and use this number as the array size). The two accounts are updated, and the accounts text-file is then updated and the new source balance is displayed:

|  |
| --- |
| Please select your choice: **5**  Please enter source accountID: **1004**  Please enter destination accounID: **1005**  Please enter transfer amount (SAR): **6000**  Source balance before transfer : 58,000.00 Saudi Riyals  Amount transferred to account 1005: 6,000 Saudi Riyals  New source balance: 52,000.00 Saudi Riyals  Accounts file has been updated . . .  Press Enter key to continue . . . |
| Please select your choice:  **5**  Please enter source accountID: **1004**  Please enter destination accounID: **20076**  Please enter transfer amount (SAR): **6000**  Error: Invalid destination account ID  Press Enter key to continue . . . |

**Options 6** . Add New Account

To implement Option 6, search the text-file for the accountID of the account to be added. If the accountID exists, display an error; otherwise append the new account information to the end of the text-file by an output statement that first generates a new line; but that does not generate an extra new line at the end of the text-file.

**Options 7. Delete Account**

To implement option 7, search the text-file for the accountID of the account to be deleted, keeping track of its line number in the file. If the accountID does not exist, display an error; otherwise read the account information from the text-file into an array of **Account** objects, then overwrite the text-file with the contents of the array, but skipping the account you want to delete. Make sure you write the last line without generating an extra blank line at the end of the text-file.

**Note:**

* Your project must not use parallel arrays. IT MUST USE an array of **Account** objects.
* Your project must not use 2D-arrays.
* THE CODE FOR PROCESSING THE OPTIONS MUST NOT BE PLACED IN Acounts.class ; IT MUST BE IN AccountDriver.class
* You may use private static methods in your program, like countLines, searchForID, updateTextFile, loadToArrayOfObjects, etc.
* Use the clause **throws FileNotFoundException** or **throws IOException** for each method that performs File I/O.
* You must not share code with another project group. Doing so will result in a zero grade for all groups involved.
* Project demo slots of 15 to 20 minutes per group will be annouced later. All group members are required to attend the project demo together. A grade of zero will be given for any student not attending the Project Demo.

**Hints:**

* When you update the accounts text-file, make sure you do not insert a blank line at the end of the file, because this will generate reading errors. Write the last file line without generating **‘\n’**
* Whenever you use **nextLine( )** that follows **next( )**, **nextInt( )** or **nextDouble( )**, make sure you use a dummy **nextLine( )** to remove **‘\n** from the input buffer.
* To return a formatted string from a method use:

**String.format(FormatString, ExpressionList)**

Examples:

1. return **String.format**(“ID = %d, Balance = **%,.2f** Saudi Riyals”, id, balance);

2. public void withdraw(int accountID, double amount) throws IllegalArgumentException

{

if(amount > balance)

throw new

IllegalArgumentException(String.format(“Error: Available balance is **%,.2f**”, balance));

// . . .

}