**Lab Assignment 1:**

**Banking System**

|  |
| --- |
| **Due Date**  **(a two-week LA)** |
| **1/27/16 11:59pm – For Wednesday Lab Sections**  **1/28/16 11:59pm – For Thursday Lab Sections** |

**Objectives**

* Review the topics covered in CS1110
* File input
* String Split
* Classes

**Problem Specification**

Write a Java application of a Banking System prototype. The program loads all the account data from a text file. The user interacts with the program via the terminal. The program can process three types of transactions: deposit, withdraw and transfer. The updated balance will be temporarily stored in the memory until the user wants to exit the program. Before the program terminates, the updated account balances will be written back to the text file.

**Input File**

The input file will be provided. Alternatively, you can create your own file, but it should retain the format of the given file. The input file contains several lines. Each line contains two numbers: an integer and a double separated by a single space. The integer is the account number while the double is the current balance of the corresponding account number (on the same line). Here is an example.

*Example*

100001 100.01

100002 1000.00

100003 245.65

100004 24565.32

100005 2345.78

100006 23245.62

100007 4565.74

100008 2435.85

100009 324216.33

100010 0.00

**User Interaction**

User interacts with the program through the menu options. Here is an example of a user interaction.

*1.Deposit*

*2.Withdraw*

*3.Transfer*

*0.Exit*

*->1*

*Please input the account number->100001*

*Please input the amount->200*

*1.Deposit*

*2.Withdraw*

*3.Transfer*

*0.Exit*

*->2*

*Please input the account number->100002*

*Please input the amount->300*

*1.Deposit*

*2.Withdraw*

*3.Transfer*

*0.Exit*

*->3*

*Please input the account number to transfer from->100003*

*Please input the account number to transfer to->100004*

*Please input the amount->200*

*1.Deposit*

*2.Withdraw*

*3.Transfer*

*0.Exit*

*->0*

Note that a “transfer” transaction is a withdrawal from one account followed by a deposit of the same amount to another account.

**Update Account Database (input file)**

After the program terminates, the original input file would have been updated with the new balances for the affected accounts. In the example below, you will see that the balances of the first four accounts have changed.

100001 500.01

100002 200.00

100003 45.65

100004 24765.32

100005 2345.78

100006 23245.62

100007 4565.74

100008 2435.85

100009 324216.33

100010 0.00

**Design** **Requirements**

***You are required to finish the design report before you leave the lab.***

Your lab report should contain the following two parts.

1. Basic Structure

List all the methods you plan to have for this project. Give each method a one-sentence description explaining what this method does.

The required methods are listed below. You can copy them to your report.

Your program should have an **Account** class. The attribute of the account class are:

* **private int accountNumber;**
* **private double accountBalance;**

The constructor for the Account class should receive the values to initialize the attributes and contain the code to assign these values to the attributes.

**public** Account(**int** accountNum, **double** accountBal);

The Account class should also have a get method for each of the attributes and methods for each of the two main transactions (deposit, withdraw). There should be no set method for the account number as this should not change.

// Return account number

**public** **int** getAccountNumber();

// Return account balance

**public** **double** getAccountBalance();

// Deposit specified amount in this account

**public** **void** deposit(**double** depositAmount);

// Withdraw specified amount from this account

**public** **void** withdraw(**double** withdrawAmount);

The following methods should be in your main method. The transfer transaction should be implemented in the main class (not in the Account class) because it involves two different accounts.

// Print menu showing available options (1, 2, 3, 0) to the screen

**public** **static** **void** printMenu();

// Read in the user’s desired option / choice

**public** **static** **int** getChoice(Scanner keyboard)

// Get relevant account number from user for a deposit or withdraw transaction

**public** **static** **int** getAccountNumber(Scanner keyboard)

// Get relevant account numbers from user for a transfer transaction

**public** **static** **int**[] getTransferAccountNumbers(Scanner keyboard)

// Get amount from user for the current transaction

**public** **static** **double** getAmount(Scanner keyboard)

// Update input file with the new account balances based on user’s transactions

**public** **static** **void** updateAccountDatabase(Account[] account) **throws** IOException;

1. Pseudocode

Write pseudocode for all the required methods.

**Additional Requirements**

**Coding Standards**

You must adhere to all conventions in the CS 1120 Java coding standards. This includes the use of white space for readability and the use of comments to explain the meaning of various methods and attributes. Be sure to follow the conventions for naming classes, variables, method parameters and methods.

**Assignment Submission**

* Generate a .zip file that contains all your files, including:
  + Program Files
  + Any input or output files
* Submit the .zip file to the appropriate folder in elearning.

**NOTE**: The elearning folder will be inaccessible after the due date/time.