**Project 2 Expectations**

Please review ***General Programming Expectations*** document located in Week 2 under **Week 2 Checklist**.

**Obtaining data for Assignment 2**

For Project 2, the assignment does not suggest a way of obtaining data for processing. There is a normal assumption that the programmer will generate several instances of the class and pass in a set of hard-coded values as arguments. There are two ways of obtaining the data for this program. The program can either (1) use a two-dimensional array with a set of initialized values or (2) a data file with the data being read into the program and then assigned into a two-dimensional array. Since there are various data types in the array row, you need to determine that best data type declaration for the array. Either way, a data file is supplied for this assignment to make it easier to test the programs among the members of this course and for the instructor to run the program for feedback and grading. The file to use with this program is ***PRJ2Accounts.txt***; this file was created for this specific course. It is expected that you will use this file. The following is the data in the file that can also be used to initialize the two-dimensional array:

7623, S, Jones, Phyllis, 2000.00, 0

7623, C, Jones, Phyllis, 1200.00, 2

8729, S, Smith, Cletus, 1000.00, 1

8729, C, Smith, Cletus, 1700.00, 0

7321, S, Booth, Betty, 4500.00, 2

3242, C, Seybright, Sam, 4612.00, 0

3242, S, Seybright, Sam, 100.00, 3

9823, S, Blu, Oscar, 234.00, 2

9823, C, Blu, Oscar, 2345.00, 0

2341, S, Grief, Sally, 12373.12, 1

2341, C, Grief, Sally, 5421.12, 2

8321, S, Matchel, Jacques, 2314.00, 0

8321, C, Matchel, Jacques, 56233.00, 0

The fields/columns in the file are

1. Bank Account Number
2. Type of Account [S = Savings; C = Checking]
3. Last Name of Account Holder
4. First Name of Account Holder
5. Initial Balance in the Account
6. Current Number of Transactions

If you chose to use a file to input the data, an additional method can be added to the code as either part of Account Class or an independent method inserted before the main () method. Reading a file and assignment the data into an array was covered in the prior assignment.

**One Consideration for the Initial Screen of the Program:**

**A screenshot of a cell phone

Description automatically generated**

**Example of the Balance for One Account:**

**A screenshot of a cell phone

Description automatically generated**

**Suggested Design for Assignment 2**

Here is one structure design of this program.

package cmis242.prj2smithx;

[imports here]

public class CMIS242 PRJ2SmithX {

public static Data Manipulation {

*[Declare array]*

*[METHOD: Coding to read in file and assign to array]*

*[MEHTOD: update the various rows of the array]*

*[METHOD: Display the output]*

} //end public static Data Manipulation

public static class ATM\_GUI {

*[Constructor (), define GUI Interface, listeners]*

} // end public static class ATM\_GUI

public static class Accounts {

*[Balance, deposit, withdrawal, transfer]*

} // end public static class Accounts

public static class Savings extends Accounts {

*[Constructor (), insufficient funds, fee, transfer, etc.]*

} // end public static class Savings extends Accounts

public static class Checking extends Accounts {

*[Constructor (), insufficient funds, multiples, fee, transfer, etc.]]*

} // end public static Checking extends Accounts

public static void main (String args[]) {

*[Declare instance of class]*

*[Call to read and assign file]*

*[Call to display output before any transactions]*

*[Call class to perform transactions]*

*[Call to display output after transactions completed]*

} // end public static void main (String args []) {

} // end public class CMIS242 PRJ2SmithX

The GUI Class

An example of what the ATM screen should look like is displayed at the beginning of the assignment in ***Project 2.pdf***. There are **two issues with the window [ATM screen]** as it appears in the pdf document. Because you have a set of accounts, there should be a TextField in the window to enter an account number. The current TextField shown should be considered the entry of the bank account number. Both TextFields should have a label explaining the purpose of the field. The bottom of the window should be like the following display:

Account Number: [TextField window]

Amount of Transaction: [TextField window]

There should be displays for all errors, the balance, and the various transactions. Each display should contain a reference to the account number and type. Use JOptionPane for all these display occurrences. A partial listing of displays using JOptionPane is listed below but may not be all the possibilities.

* Accounts 7623 over 4 transactions: $1.50 added to transaction
* Withdrawal from 7623 must be in $20.00 multiples
* Insufficient funds for withdrawal 7623 Checking; Balance $18.50
* Insufficient funds in 7623 Savings; Balance $0.98
* The balance for 7623 Savings is $2,000.00
* The balance for 7623 Savings after $60.00 withdrawal is $1,940.00
* The balance for 7623 Checking after $100.00 deposit is $1,300.00
* The balance for 7623 Checking after transfer of $100 to Savings is $1,200.00 and balance for 7623 Savings after transfer of $100.00 from Checking is $2,140.00

When the user wishes to end the program, there should be a button to Exit/End the program. Add an Exit/End button to the GUI window.

**There are some considerations in designing the program**:

1. When the GUI window first displays only the Balance button should be available for use. Only after an account number is entered and verified as existing should the Deposit, Withdrawal, and Transfer buttons be available for use.
2. If the account number does not exist, an error message should be displayed. In normal circumstance, the program would ask if a new account should be created. For this assignment this process will not be implemented.
3. The single TextField is ambiguous. It is assumed that it exists for the number used to change the balance of the various accounts, but then why is there a Balance button. With the set of data provided, there would need to be a way of obtaining an account number to retrieve a balance. Then another field for the transaction needed for the other buttons. The following is the preferred display of elements on the window/panel already reference above:
   1. The TextField currently visible on the window/panel is used to retrieve the account number.
   2. A second TextField should be created to accept a transaction amount.
4. To the left of the TextField(s) should be a label as suggested above.
5. There is the possibility of the account holder having only a savings or checking account, therefore when the Transfer function is invoked there could be a problem requiring much coding. For this assignment assume that there are two accounts – a savings and a checking – for a bank customer.
6. To use the checkbox for processing, check the appropriate type field related to the account. [The Savings radio button is the default button.]
7. The checking used for Withdrawal should also apply to Deposit.
8. For the Insufficient Funds check, assume there is at least one multiple of $20 in the checking account and more than $1.00 in the savings account. Only one multiple is necessary to determine sufficient funds.
9. When a value is placed in a TextField it is normally a String and needs to be parsed into an integer or double, therefore, the statement “ensure . . . the text field is numeric.” is confusing. The text field for the transaction amount needs to be a double, so it should be parsed into a double.
10. If the array is a String data type, then the account number can be checked without parsing.
11. How to proceed with the accumulation of four withdrawals to impose a fee is not clear. The suggested rules for this requirement of the assignment use the following conditions for coding:
    1. Check the number of withdrawals between the two bank accounts.
    2. Sum the numbers in column 5 of the array for the two rows that relate to the accounts of one bank customer.
    3. If the total withdrawals are less than 4, then process the withdrawal and add one to the appropriate account in column 5.
    4. If the total withdrawals are 4, then add one to the appropriate account in column 5 and deduct a $1.50 from the balance of the appropriate account after the transaction is completed.
    5. To streamline the check of more than four withdrawals, change both numbers in column 5 to a 9 once four is reached and use the 9 for adding the $1.50 rather than add the two accounts. [This is optional.]
    6. The program can have a process that resets the number of withdrawals for each account to zero. [This is optional.]

**Expectations When the Program Runs:**

1. When the program begins the process of modifying the accounts, display in the **Output Window of the IDE** all the records in the array in horizontal order.

7623, S, Jones, Phyllis, 2000.00, 0

7623, C, Jones, Phyllis, 1200.00, 2

8729, S, Smith, Cletus, 1000.00, 1

8729, C, Smith, Cletus, 1700.00, 0

7321, S, Booth, Betty, 4500.00, 2

3242, C, Seybright, Sam, 4612.00, 0

3242, S, Seybright, Sam, 100.00, 3

9823, S, Blu, Oscar, 234.00, 2

9823, C, Blu, Oscar, 2345.00, 0

2341, S, Grief, Sally, 12373.12, 1

2341, C, Grief, Sally, 5421.12, 2

8321, S, Matchel, Jacques, 2314.00, 0

8321, C, Matchel, Jacques, 56233.00, 0

1. Follow the Test Plan scenarios presented in the matrix to process through the program.
2. When the user clicks the Exit/End button on the window, a new set of records is displayed in the **Output Window of the IDE** based on the values currently in the array. [All the changes made during the processing of data through the program run should be reflected in the new display.]