Homework 7 – ISM 3230 Summer 2017

Due to Canvas: Monday, July 24th 11:59pm

Bank application

Write a program that allows the user to create and process bank account information for a customer. The user can create a new account, add money to the account, calculate interest, deduct money from the account, and print the balance information. In addition, the user can also update most information for a given bank account.

You **must** use two classes in this program – one class containing your main method and another class to store and process the account information. Please refer to the UML diagram and description on the next page for specifics regarding the BankAccount class. All other logic should belong in the class with your `main` method.

Menu to be displayed for the user:

```
ACCOUNT PROCESSING MENU
1. Create new account - empty account
2. Create new account - information available
3. Make deposit
4. Make withdrawal
5. Calculate monthly interest
6. View account balance
7. Next available account number
8. Update monthly interest rate
9. Print account information
10. Exit
```

Note: To indicate the menu option, the user will enter the **number** of the menu option, NOT the menu action. For example, if the user wants to create a new account, he/she would type 1 (not "Create new account"). Please refer to the Sample Output file for full details of what the program should look like when it runs. Additional details follow regarding the different methods of BankAccount and features of the main program.

Be sure to implement the following menu items as follows for full credit:

- When creating a new empty account (option #1), you MUST use the default constructor of BankAccount to create the new account object. Set the value of the account number (nAccountNumber) using the next account number (nNextAccountNumber) in this constructor. Be sure to increment the value of nNextAccountNumber after you set the value for the current account. For example, if nAccountNumber = 0 and nNextAccountNumber = 123 when the account is created, after the constructor runs nAccountNumber = 123 and nNextAccountNumber = 124.
- When creating a new account with information (option #2), you MUST use the overloaded constructor of BankAccount to create the new account object. Set the value of the account number (nAccountNumber) using the next account number (nNextAccountNumber) in this constructor. Be sure to increment the value of nNextAccountNumber after you set the value for the current account. For example, if nAccountNumber = 0 and nNextAccountNumber = 123 when the account is created, after the constructor runs nAccountNumber = 123 and nNextAccountNumber = 124.

UML Diagram for the BankAccount class

BankAccount
-nAccountNumber: int
-nNextAccountNumber: int
-dBalance: double
-dMonthlyInterestRate: double
+BankAccount()
+BankAccount(dStartBalance: double, dIntRate: double)
+getAccountNumber(): int
+getBalance(): double
+getMonthlyInterestRate(): double
+setMonthlyInterestRate(dRate: double): void
+getNextAccountNumber(): int
+printBankInfo(): void
+makeDeposit(dDepositAmt: double): void
+makeWithdrawal(dWithdrawalAmt: double): Boolean
+monthlyInterest(): double

Additional details for the BankAccount class:
The following methods are the accessors and mutators for the BankAccount class:

- *getAccountNumber*
- *getBalance*
- *getMonthlyInterestRate*
- *setMonthlyInterestRate*
- *getNextAccountNumber*

The following methods are not accessors or mutators. They will manipulate the attributes of the BankAccount class or print the information stored in BankAccount:

- *printBankInfo* – should print the following information (refer to the Sample Output file for formatting details)
    - Account number
    - Balance
    - Monthly interest (call the monthlyInterest method to access this information – do NOT perform the calculation in this method)
- *makeDeposit* – should add the amount deposited to the account balance
- *makeWithdrawal* – should deduct the amount deposited from the account balance and return **true** if there are sufficient funds; if there are insufficient funds, the balance should not change and **false** should be returned
- *monthlyInterest* – should calculate the interest (total dollars) that would be earned each month on the given account balance

This assignment MUST be created individually. You must turn in your OWN source code and Java bytecode executable files. You MAY NOT share files!

Instructions

- Compile and execute your program to ensure that it works correctly.
- Be sure to run your final program using the sample data included in the Sample Output file to ensure your program works properly.
- Make sure your output labels match those in the Sample Output file exactly.

Notes

- You MUST store any calculations in variables (i.e. do not calculate the length conversions directly in your `System.out.println` statements)
- You do NOT need to worry about formatting the decimal places on values you calculate
- You must include the two different classes specified on the previous page (BankAccount and a class with your `main` method). You may include additional methods in the class with your `main` method if you want, but BankAccount should include ALL methods listed in the UML diagram.
- You must use/call the available methods in BankAccount to manipulate the bank account information. Your main method should not change attribute values of BankAccount directly.
- You may perform the tasks in any order as long as your output follows the order of the output in the example scenarios contained in the Sample Output file

To receive full credit:

- Submit the following files in a single **zip file** to Canvas
  - Files with your Java source code (.java files) [you will have two for this assignment]
  - Files with your Java bytecode executable files (.class files) [you will have two for this assignment]
  - **Refer to the instructions for creating the zip file on Canvas – if you do not create the file with the correct folders and structure, you will lose points**
    - *Modules → NetBeans Information -> NetBeans -> where are my files for my hw?*
- You must follow the appropriate Coding Standards listed in the Coding Standards document under Supplemental Materials on Canvas.
  - 40% of your grade on the source code will be based on how well you follow these standards and how well you comment your source code
- Submit your zip file to Canvas using the Assignment submission feature by 11:59pm 7/24/2017. Instructions for submission are available on Canvas where you downloaded this file (Assignments --> Homework 7).