Object Oriented Programming-2 using Java

Assignment-2　　Class # 1205_10979　　MM – 20 Weightage – 10%

Note – This assignment requires creating UML and Implementing Program. Both parts are compulsory. All labs/assignments are individual assignments unless specified by the instructor and must be completed individually by all students; please refer to the Academic Honesty document located at;- https://caps.sheridancollege.ca/student-guide/academic-policies-and-procedures.aspx

Submission guidelines:-

a. Word Document - Copy/paste your UML Diagram and java code and output screenshots of execution of program from intellij terminal.
b. Generate html files using Javadoc utility.
c. Make a folder, add .java files and .html files to this folder and zip it.
d. Upload word document and zip folder to slate dropbox.
e. This assignment requires 3 classes i.e. a BankAccount, SavingAccount and AccountDemo class (to demonstrate the use and accurate implementation of BankAccount and SavingAccount classes).
f. UML Diagram and java code should match with each other.
g. Both submissions are compulsory

Rubrics Question-1:-

I. **UML diagram : 3 Marks**
II. **Coding Part : 17 Marks (marks distribution is given at each point)**

Question-1

Part-1;-

Create UML diagram for part-1 only i.e. for BankAccount and SavingsAccount class as per following specifications given by the client;-

Design a class named BankAccount with following instructions;-

1. BankAccount class should hold following information about account. Choose appropriate variable names and datatypes for storing following

information. (0.5 Marks for each correct variable declaration – total 2.5 marks)

A. Balance – double value
B. Number of deposits this month – int value
C. Number of withdrawals – int value
D. Annual Interest rate – double value
E. Monthly Service charges – double value

2. The class has following methods;- (each correct method will give you 1 mark, so 8*1=8 marks)

A. Default Constructor – Assigns 0 or 0.0 as the default value to each variable.
B. Parameterized constructor – This constructor should accept arguments for balance, monthly service charges and annual interest rate and then assigns it to respective variables.
C. Accessor methods for each variable.
D. Mutator methods for each variable.
E. Deposit method – This method should take amount of deposit as the argument. The method should add amount to balance. It should increment the variable holding number of deposits by 1.
F. Withdraw method - This method should take amount of withdraw as the argument. The method should subtract amount from balance. It should increment the variable holding number of withdrawals by 1. Person should not be able to withdraw money if account balance is less than amount entered by user. Make sure to validate the amount.
G. calcInterest – This method should add interest earned by the customer on amount they have in account. It should be done using following formulas;-

   Monthly Interest Rate=Annual Interest Rate/12

   Monthly Interest=Balance*Monthly Interest Rate

   Balance=Balance+Monthly Interest

H. monthlyProcess – This method should subtract Monthy Service Charges from balance, calls the calcInterest method and sets the number of withdrawals, number of deposits and monthly service charges to 0.

Design a SavingsAccount class that extends BankAccount class. (Please note accessor or mutator methods are not required in SavingAccount class)

1. SavingsAccount class should hold following information about account. Choose appropriate variable names and datatypes for storing following information. (0.5)
   A. Status – boolean value where true indicates status is active and false indicates inactive status. If account balance is less than 50 dollars then account would become inactive, otherwise account would be active.

2. The class has following methods;- (each correct method will give you 1 mark, so 3*1=3 marks)

   A. Parameterized constructor – This constructor should accept arguments for balance, monthly service charges and annual interest rate and then should pass these argument to BankAccount class constructor. It should update status of account using balance i.e. if balance<50 then inactive otherwise active.
   B. Deposit method – This method should take amount of deposit as the argument. The method should call deposit method defined in BankAccount class to update the balance. It should check balance, if account was inactive earlier and now after depositing if balance is above 50, then account status should be changed to active. (Hint: - you can use getBalance() method from BankAccount class to know the balance)
   C. Withdraw method - This method should take amount of withdraw as the argument. It should check status,
      a. if status is inactive then no withdrawals should be allowed.

b. If status is active then this method should call withdraw method from BankAccount class to update the balance. After updating the balance, again status should be updated.

Part-2;- (3 Marks)

Create class AccountDemo. Use Scanner class to take required input from user. Demonstrate the use of BankAccount and SavingsAccount classs in this program by following steps;-

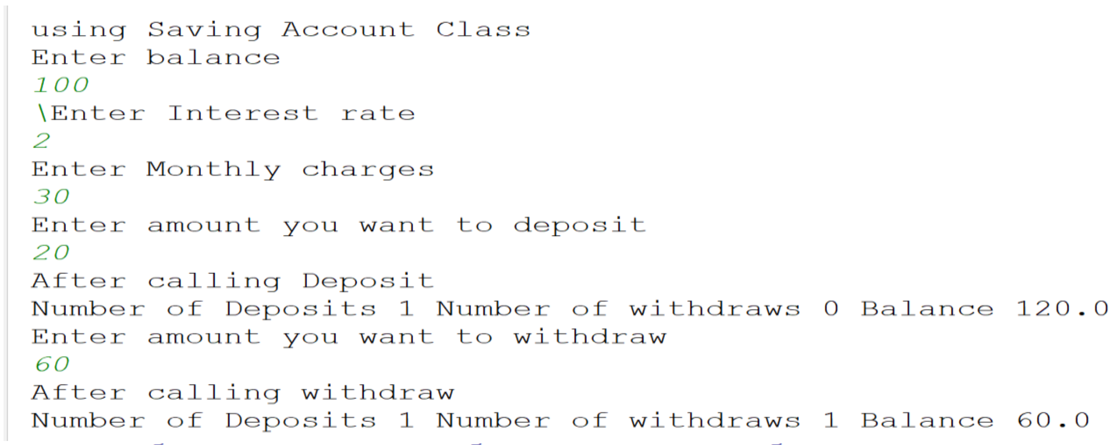
Create first object for BankAccount class using parameterized constructor and then call the deposit method and withdraw using objects you just created and print the number of deposits, number of withdrawals and updated balance after each call.

Create 2nd object for SavingsAccount class using parameterized constructor, and then call the deposit method and withdraw using object you created for SavingAccount class and print the number of deposits, number of withdrawals and updated balance after each call.

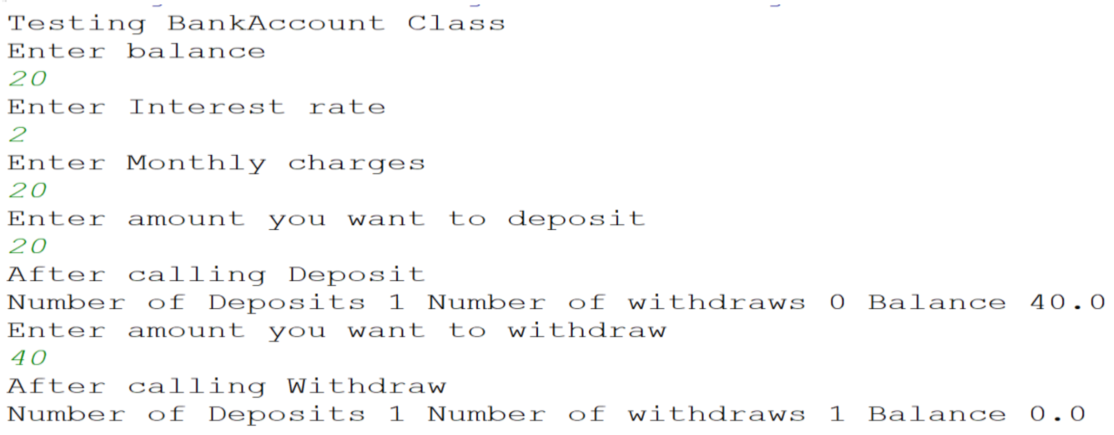
Output Screenshots are provided for reference:-

I executed this program twice, both sample run screenshots are provided here. First one showing normal execution, second one validating data and showing error messages. Here is the sample output:-

Sample Run -1



```
using Saving Account Class
Enter balance
100
\Enter Interest rate
2
Enter Monthly charges
30
Enter amount you want to deposit
20
After calling Deposit
Number of Deposits 1 Number of withdraws 0 Balance 120.0
Enter amount you want to withdraw
60
After calling withdraw
Number of Deposits 1 Number of withdraws 1 Balance 60.0
```



```
Testing BankAccount Class
Enter balance
20
Enter Interest rate
2
Enter Monthly charges
20
Enter amount you want to deposit
20
After calling Deposit
Number of Deposits 1 Number of withdraws 0 Balance 40.0
Enter amount you want to withdraw
40
After calling Withdraw
Number of Deposits 1 Number of withdraws 1 Balance 0.0
```

Sample Run -2

```
Testing BankAccount Class
Enter balance
100
Enter Interest rate
2
Enter Monthly charges
20
Enter amount you want to deposit
20
After calling Deposit
Number of Deposits 1 Number of withdraws 0 Balance 120.0
Enter amount you want to withdraw
200
Insufficient balance <- error
After calling Withdraw
Number of Deposits 1 Number of withdraws 0 Balance 120.0
```

```
 using Saving Account Class
 Enter balance
 10
 Enter Interest rate
 2
 Enter Monthly charges
 20
 Enter amount you want to deposit                    <50
 20
 After calling Deposit
 Number of Deposits 1 Number of withdraws 0 Balance 30.0
 Enter amount you want to withdraw
 20
 Account is deactivated
 After calling withdraw
 Number of Deposits 1 Number of withdraws 0 Balance 30.0
```