Object-Oriented Programming Lab#8, Fall 2021

Today's Topics

- Inheritance
- encapsulation
- method override
- method overload
- subclass polymorphism
- Add project reference

A Banking System

Create a **Banking System**, where a user can **create new account**, **deposit** money, **withdraw** money, **check** the balance, and view the transaction records. The application will be used by both the bank employee and customer/account holder.

There are different types of **BankAccount** a user can create. See below for the requirements of different types of account.

- **Savings account:** A savings account allows user to accumulate ***interest*** on funds he has saved for future needs. Savings account required a ***minimum balance***. For our purpose let's assume the **minimum balance** is 2000 Tk and **interest rate** is 5%. From savings account, user is only **allowed to withdraw a maximum amount** of money which will be set up during the account creation.

- **Current account:** Current account offers easy access to your money for your daily transactional needs and help keep your cash secure. You need a **trading license** to open a Current account. There is no restriction on how much money you can withdraw from Current account but you need a ***minimum balance*** of 5000 TK in your account.

The system will have the following functionalities.

a. Add new BankAccount into in the system.
b. Search for a specific account.
c. Deposit money to an account on behalf of the customer.
d. Withdraw money from a specific account on behalf of customer.

e. View the summary of an account (name, account number, type, and balance)

f. View the list of specific type of Bank Accounts (Savings or Current)

g. View the list of all accounts.

What you need to do: (Note: Do not use default package)

1. **Create a Project name BankingLibrary (and do the following)**
2. Create **BankAccount** class under a **package**:
    a. Add 5 **private** instance variables; ***memberName***, ***memberNID***, ***accountNumber***, ***accountBalance***, and ***minimumBalance***
    b. Implement constructor. You need to pass ***memberName***, ***memberNID***, ***accountBalance*** & ***minimumBalance*** as parameter.
        - *You need to auto-generate a 5 digit **accountNumber** inside the constructor. So, you do not need to pass the **accountNumber** as a parameter in the constructor.* (See the example below for how to generate 5 digit random number)

    Add the following methods inside the class:

    a. **public void deposit(double depAmount)**

        - Inside the method the ***accountBalance*** need to be increased by the "***depAmount***" amount.

    b. **public void withdraw(double withAmount)**
        - The ***accountBalance*** is decreased by "***withAmount***" amount. We have to make sure the balance does not become less than ***minimumBalance***.

    c. Add **public getter** method for ***accountNumber***, ***accountBalance*** attributes and getter/setter method for other attributes.

    d. **Override toString()** method

        - From the method, return a String in the format "membeName-accountNumber-accountBalance-minimumBalance".

    Code to generate 5 digit random number: (3 different examples below)

    *The **num** variable in the examples below will store a 5 digit number in String format.*

    ***Example1**:*

```
Random rand = new Random();
String num ="" + rand.nextInt(10) + rand.nextInt(10)+ rand.nextInt(10)+
rand.nextInt(10)+ rand.nextInt(10);
```

    ***Example2**:*

```
Random rand = new Random();
String num = 10000 + rand.nextInt(89999) + "";
```

***Example3**:*

```
String num = 10000 + (int)(Math.random()*89999) + "";
```

3. Create a **SavingsAccount** class under the same package:
    a. Make this class a **subclass** of **BankAccount** class.
    b. Add **two additional private** instance variables.
        - One is "***interest***", initialize it to 5% [0.05].
        - Another variable for ***maximum withdraw*** amount limit, name it as ***maxWithLimit***.
    c. Implement constructor.

       You need to pass ***memberName***, ***memberNID, accountBalance***, and ***maxWithLimit*** as parameter. Inside the constructor, call parent class's constructor. Note: You need to make sure ***minimumBalance*** is set to 2000.
    d. Add a **private** method ***double calculateInterest()***

       Inside the method calculate the total interest (accountBalance*interest) and return the total interest.
    e. Add ***public double getNetBalance***() method.

       This method will calculate the total interest by calling ***calculateInterest()*** method and return (*accountBalance* + total interest) but **it won't** change the ***accountBalance*** value.
    f. Override ***withdraw(double)*** method.

       This method will allow to withdraw money if the withdraw amount is less than the maximum withdraw limit and doesn't set the ***accountBalance*** less than ***minimumBalance*** after withdraw. So, you need to **call the parent class's withdraw method**.

    g. Override ***the toString()***

    - Call the ***toString()*** method of parent class and then concatenate "-maxWithLimit".

    h. Add getter/setter method for the additional attributes.

4. Create a **CurrentAccount** class under the same package:
    a. Make this class as the subclass of the **BankAccount** class
    b. Add an instance variable ***tradeLicenseNumber***.
    c. Implement constructor. You need to pass ***memberName***, ***memberNID, accountBalance***, and ***tradeLicenseNumber*** as parameter. Note: You need to make sure ***minimumBalance*** is set to 5000.

5. Add a class name "**Bank**" under a package which will mimic a real Bank that holds a list of **BankAccount**. You can use an Array or **ArrayList** to hold the list of **BankAccount**. So, the class will have only one attribute ***ArrayList<BankAccount> accounts***. Add the following methods to the class.

   a. ***private void addAccount(BankAccount acc)***

   - Inside the method, add the ***acc*** object to the ***accounts*** list.

   b. ***public void addAccount(String name, String mNid, double balance, double maxWithimit)***

   - Inside the method, create a ***SavingAccount*** object using the parameter provided and add the account to the list using ***addAccount(BankAccount)*** method.

   c. ***public void addAccount(String name, String mNid, double balance, String tradeLicense)***

   - Inside the method, create a ***CurrentAccount*** object using the parameter provided and add the account to the list using ***addAccount(BankAccount)*** method.

   d. ***private BankAccount findAccount(String accountNum)***

   - This method will loop through the list of the **BankAccount** (***accounts***) and find the account that has matching ***accountNumber*** as the parameter. If the matching **BankAccount** is available return the object otherwise return null.

   e. ***public void deposit(String accountNum, double amt)***

   - Inside the method call ***findAccount(String)*** to find the account with matching ***accountNum*** and then call ***deposit(double)*** method of that object.

   f. ***public* void withdraw(String *accountNum*, double amt)**

   - Inside the method call ***findAccount(String)*** to find the account with matching ***accountNum*** and then call ***withdraw(double)*** method of that object.

   g. ***public* void transfer(String fromAccNum, String toAccNum, double amt)**

   - This method will transfer money from one account to another. So, inside the method, call ***findAccount(..)*** for both ***fromAccNum*** and ***toAccNum***. If both accounts are available in the list, call ***withdraw(…)*** for the ***fromAccNum*** and ***deposit(…)*** for the ***toAccNum***.

   h. ***public* double getBalance(String *accountNum*)**

   - Inside the method call ***findAccount(String)*** to find the account with matching ***accountNum***. If the account is a **CurrentAccount**, call ***getBalance()*** method; otherwise call ***getNetBalance()*** method using the object.

   i. ***public* String getAccountSummary(String *accountNum*)**

   - Inside the method call ***findAccount(String)*** to find the account with matching ***accountNum*** and then call **toString()** method of that object and return the value.

   -

   j. ***public* String getListOfAccounts()**

- Loop through the list of the **BankAccount** (***accounts***) and call ***toString***() for each account and concatenate output of the method in a separate line.

6. Create an **application class** (that has the main method) named "**BankApp**" which will have the **main** method.
    - In the main method, ask if the user is an employee or a customer/account holder.
    - If user is an employee display the following menu to user and take necessary action.
        - Input '1' to add a new Account.

          You need to provide use a submenu to create different types of account. So, you have to ask for **user name**, **what type of account** he wants to open and what would be the **initial balance**. The system will create the account (***SavingsAccount*** or ***CurrentAccount*** object) with a randomly generated 5 digit account number.
        - Input '2' to deposit to an existing account
        - Input '3' to withdraw from an account.
        - **Input '4' to transfer money from one account to another account.**
        - Input '5' to check the balance of an account.
        - Input '6' to display the summary of a specific account. Summary will include the membeName, accountNumber, accountBalance, and minimumBalance.
        - Input '0' to exit the system.