Object-Oriented Programming Lab#7, Spring 2019

Today's Topics

- Inheritance
- method override
- subclass polymorphism

A Banking System

Create a **Banking System**, where a user can **create new account**, **deposit** money, **withdraw** money and **check** the balance. 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.

Object-Oriented Programming Lab#7, Spring 2019

What you need to do:

1. Create the **BankAccount** class:
    - Add 4 instance variables; ***memberName***, ***accountNumber***, ***accountBalance***, ***minimumBalance.***
    - Implement constructor. You need to pass ***memberName***, ***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
        - **void *deposit*(double amt).**
        - **void *withdraw*(double amt)**
            - You should only allow withdrawing if the ***accountBalance*** is equal or more than ***minimumBalance*** after withdrawing.
        - **double *getBalance*()** which will return the ***accountBalance*** attribute.
        - **void display() -** Print the value of all attributes.

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) + "";
```

2. Create a **SavingsAccount** class:
    - This class is a **subclass** of **BankAccount** class.
    - This will have **two additional** instance variables
        - One is "***interest***" and initialized to 5%.
        - Another variable for *maximum withdraw* amount limit, name it as ***maxWithLimit***.
    - Implement constructor.

      You need to pass ***memberName***, ***accountBalance*** , and ***maxWithLimit*** as parameter. Inside the constructor, call parent class's constructor.

- Override ***getBalance***() method.

  This method will calculate the total interest of the ***accountBalance*** value and return (*accountBalance* + total interest) but it won't change the ***accountBalance*** value.
- Override ***withdraw(double amount)*** method.

  This method will allow to withdraw money if the withdraw ***amount*** is less than the ***maxWithLimit*** and doesn't set the ***accountBalance*** less than ***minimumBalance*** after withdraw. So, you need to check the ***maxWithLimit*** condition and then call the **withdraw()** method of **BankAccount** class.

3. Create a **CurrentAccount** class:
   - Should **extend** the **BankAccount** class
   - Add an instance variable ***tradeLicenseNumber***.
   - Implement constructor.

     You need to pass ***memberName***, ***accountBalance*** , and ***tradeLicenseNumber*** as parameter. Inside the constructor, call parent class's constructor.

4. Now create a class name "**Bank**" 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 ***BankAccount[] accounts***. Add the following methods to the class.
   - ***void addAccount(BankAccount)***
   - This method will add a new ***BankAccount*** object to the list ***accounts***. Use the parameters to create the BankAccount object.
   - ***void addAccount(String name, double balance, double minimumBal, double maxWithimit)***
   - This method will create a ***SavingsAccount*** object ***acc*** using the parameter provided and add the account to the list using ***addAccount(BankAccount acc)*** method.
   - ***void addAccount(String name, double balance, String tradeLicense)***
   - This method will create a ***CurrentAccount*** object using the parameter provided and add the account to the list using ***addAccount(BankAccount acc)*** method.
   - ***BankAccount findAccount(String aacountNum)***
   - 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.
   - ***void deposit(String aacountNum, double amt)***
   - Inside the method call ***findAccount(String aacountNum)*** to find the ***BankAccount*** with matching ***accountNum*** and then call ***deposit(double amt)*** method of that object.
   - **void withdraw(String *aacountNum*, double amt)**
   - Inside the method call ***findAccount(String aacountNum)*** to find the ***BankAccount*** with matching ***accountNum*** and then call ***withdraw(double)*** method of that object.

- **void display()**
- Loop through the list of the **BankAccounts** (***accounts*** *variable*) and call ***display ()*** method of **BankAccount** class.

5. Create an **application class** (that has the main method) named "**BankApp**" which will have the **main** method.
    - In the main method, declare a variable '***bank'*** of **Bank** type. Display the following menu to user and take necessary action(call appropriate method for ***bank*** variable).
        - Input '1' to add a new Account.

          You need to provide a submenu to create different types of account. You have to ask for **what type of BankAccount** he/she wants to open. Depending on the user response you need to ask for the remaining inputs and then create the account (***SavingsAccount*** or ***CurrentAccount*** object) and assign it to ***bank*** variable.
        - Input '2' to deposit to an existing account
        - Input '3' to withdraw from an account.
        - Input '4' to display the list of the accounts.
        - Input '0' to exit the system.