

Object-Oriented Programming Lab#7, Spring 2020

Today's Topics

- Inheritance
- encapsulation
- method override
- method overload
- subclass polymorphism
- abstract class

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, Fall 2019

What you need to do:

1. Create an abstract **BankAccount** class:

- a. Add 4 private instance variables; **memberName**, **accountNumber**, **accountBalance**, **minimumBalance**.
- b. 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 inside the class;

- c. **public void deposit(double depAmount)**
 - Inside the method the **accountBalance** need to be increased by the "**depAmount**" amount.
- d. **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**.
- e. Add **public getter** method for **accountNumber**, **accountBalance** attributes and getter/setter method for other attributes.
- f. **public void display()**
 - This method displays the attributes in the format "Name:[memberName]; Account Number:[accountNumber]; Balance:[accountBalance]".

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:

- 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**, **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 **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 **void display()**

- Call the **display()** method of parent class and then print "; interest:[accountBalance*interest]; maxWithdrawLimit:[maxWithLimit]".

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

3. Create a **CurrentAccount** class:

- Make this class as the subclass of the **BankAccount** class
- Add an instance variable **tradeLicenseNumber**.
- Implement constructor. . Note: You need to make sure **minimumBalance** is set to 5000.

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 **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. **void addAccount(String name, 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. **void addAccount(String name, 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. ***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. ***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. ***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.
- h. ***void display(String accountNum)***
 - Inside the method call ***findAccount(String)*** to find the account with matching **accountNum** and then call ***display()*** method of that object.
- i. ***void display()***
 - Loop through the list of the **BankAccount** (*accounts*) 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, 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 (***SavingsAccoun*** 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 check the balance of an account.
- Input ‘5’ to display the details of a specific account.
- Input ‘6’ to display the list of the accounts.
- Input ‘0’ to exit the system.