# Object-Oriented Programming Lab#7

## Today's Topics

- Class and Object
- Constructor - Initialization of fields.
- Method Overloading
- Access Modifier and Encapsulation
- User Input - JOptionPane
- ArrayList

**ArrayLIst:**

| **Action** | **Code** |
|---|---|
| Creating an ArrayList | ArrayList<T> list = new ArrayList<>(); |
| Adding element to arraylist | list.add(new T()); |
| Accessing an element | T t = list.get(int index) |
| Size of arraylist | int l = list.size(); |

## Problems/Assignments – Banking System

Create a Banking System, where a user can **create** a new account, **deposit** money, **withdraw** money and **check** the balance. Each Account is identified by its **account number, balance, and the name** of the account holder. The system should be able to handle multiple accounts.

# Object-Oriented Programming Lab#7

**What you need to do:**

1) Create a **BankAccount** class that has **3 private instance variables**; ***memeberName***, ***accNumber***, and ***accountBalance***.
   - Implement a **constructor**. You need to pass ***memberName*** *and* ***accountBalance*** as parameters.
     - *You need to auto-generate 5 digits* ***accNumber*** *inside the constructor. So, you do not need to pass the* ***acctNumber*** *as a parameter in the constructor.* (See the example below for how to generate 5 digits random number)

   Add the following methods inside the class;

   **a. public void deposit(double depAmount)**
   - Inside the method, the ***accountBalance*** needs 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 balanced does not become negative.

   **c. Getter method for all attributes.**
   - **i. public String getMemberName()**
     The method returns the ***memberName***.
   - **ii. public String getAccNum()**
     The method returns the ***accNumber***.
   - **iii. public double getBalance()**
     The method returns the ***accountBalance***.

   **d. public void display()**
   - This method displays the attributes in the format "Name:[membeName]; Account Number:[accountNumber]; Balance:[accountBalance]".

   **Code to generate 5 digits 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 another **class** named "**Bank**" which will have 2 attributes; ***bankName*** and an **ArrayList** (named ***accounts)*** of ***BankAccount*** type to store the list of **BankAccounts**. Now do the following.
   a. Create a **constructor** and pass bank name as parameter.
      - Inside the constructor, **initialize** the ***bankName*** and **instantiate** the ***accounts*** object.

   Create the **following methods** inside the class;

   ***a. private void addAccount(BankAccount acc)***
      - Inside this method, write code to add the ***acc*** to the list ***accounts***.
   ***b.*** **public *void addAccount(String name, double balance)***
      - This method will create a ***BankAccount*** object using the parameter provided and add the account to the list using ***addAccount(BankAccount)*** method.
   **c. private BankAccount findAccount(String accNum)**
      - Inside the method, search for the **BankAccount** with ***accNumber==accNum*** in ***accounts*** list. If the **BankAccount** exists in the list return the **BankAccount**, otherwise return **null**.
   **d. public void deposit(String accNum, double depAmount)**
      - Call ***findAccount(…)*** with the ***accNum*** as the parameter. If ***findAccount(…)*** returns a **BankAccount**, call **deposit(…)** method using that account.
   **e. public void withdraw(String accNum, double depAmount)**
      - Call ***findAccount(…)*** with the ***accNum*** as the parameter. If ***findAccount(…)*** returns a **BankAccount**, call ***withdraw(..)*** method using that account.
   f. **public *void display(*String accNum*)***
      - Call ***findAccount(…)*** with the ***accNum*** as the parameter. If ***findAccount(…)*** returns a **BankAccount**, call ***display(..)*** method using that account.
   g. **public *void display()***
      - Inside the method, display info of all **BankAccount**s in "**accounts**" ArrayList. Use ***BankAccount*** class's ***display***() method to display each BankAccount's info.

3) Now create an **application class** named "**BankApp**" which will have the **main** method.
   a. Inside the main method, create an object [name it ***bank***] of **Bank** class and then provide the following **menu** on the console. Once the user enters his/her option, you need to read the value and take appropriate action (See below) using the **bank** object.

      - Input '1' to create a new account.
        If the user chooses this option, you have to ask the user for the member name and initial balance. After getting the value call **addAccount(…)** method using **bank** object.

      - Input '2' to deposit money.
        If the user chooses this option, you have to ask the user for the account number and amount of money he wants to deposit. Call ***deposit***(…) method using **bank** object.

- Input ‘3’ to withdraw money.

  If the user chooses this option, you have to ask the user for the account number and the amount of money he wants to withdraw. Call ***withdraw***(…) method using **bank** object.

- Input ‘4’ to display the account info of a **particular** account

  If the user chooses this option, ask for the ***accNum*** from the user and call ***display(String)*** using the b**ank** object

- Input ‘5’ to display **all** accounts info

  If the user chooses this option, call ***display()*** using the **bank** object.

- Input ‘0’ to exit the system.