# Object-Oriented Programming Lab#5, Fall 23

## Today's Topics

- package
- Array of reference type
- toString()
- String concatenation/formatting
- Random number generation

**Sample Code to generate 3 digits random number: (2 different examples below)**

*The **num** variable in the examples below will store a 4-digit number in String format.*

***Example1**:*

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

***Example2**:*

```
Random rand = new Random();
String num = 100 + rand.nextInt(899) + "";
```

**ArrayLIst:** (Here T will be replaced with any Reference type such as BankAccount, Product, Student etc.)

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

# Problems/Assignments

## Problem#1

Create a Banking System, where a user can 1) **create** a new account, 2) **deposit** money to a **specific** account, 3) **withdraw** money from a **specific** account, 4) **check** the balance of a **specific** account, 5) view the details of a specific account, and 6) view the details of all accounts. 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.

**What you need to do:**

1) Create a **BankAccount** class under package ***bank*** and add the following inside the class.
   a. **3 instance variables**; ***name, accNum*** and ***balance***.
   b. Create a ***constructor*** and pass name and balance as the arguments and initialize the respective attributes. Also, generate a 6-digits random number and assign that number as String to the ***accNum*** variable.

   Add the **following 4 methods**;

   ***a. public void deposit(double depAmount)***
   - Inside the method the ***balance*** variable needs to be increased by the "***depAmount***" amount.

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

   ***c. public double getBalance()***
   - The method returns the ***balance***.

   ***d. public String toString()***
   - Inside the method generate a String in the format "Name:***name***; AccNum:***accNum***; Balance:***balance***" and return that string. Use String concatenation or ***String.format(..)*** method generate the formatted string.

2) Now create an **application class** (that has the main method) named "**Bank**" under package ***bank***. Declare and create a static **array/ArrayList** of **BankAccount** type. As BankAccount class is in a different package, you need to import the class first. If you choose array, set the array size to 10. Name the **array/ArrayList** variable as **accounts**. Add main method inside the class and provide the following **menu** on the console and take appropriate action.

   - '1' to create new account.
     - If user choose this menu, you need to create a new BankAccount object and add that to the ***accounts*** **array/ArrayList.** As you need 2 data name and balance to create a BankAccount type object, take input for these 2 fields (name, balance) from the user.

After taking the input, create a ***BankAccount*** object add the object to **accounts** array/ArrayList.

- '2' to **deposit** money.
  - For this option, you have to ask user for the account number of the account s/he wants to deposit and amount of money s/he wants to deposit. After taking the input do the following.

    - **Find** the account from the Array/ArrayList
    - If the account is available in the list**, call** the **deposit** method for that object with appropriate parameter.

- '3' to **withdraw** money.
  - For this option, you have to ask user for the account number of the account s/he wants to deposit and amount of money s/he wants to deposit. After taking the input do the following.

    - **Find** the account from the Array/ArrayList
    - If the account is available in the list**, call** the **withdraw** method for that object with appropriate parameter.

- '4' to **display** the **balance** of a specific account.
  - For this option, you have to ask user for the account number of the account s/he wants to check the balance. After taking the input do the following.

    - **Find** the account from the Array/ArrayList
    - If the account is available in the list**, call** the **getBalalnce()** method for that object and print the output.

- '5' to **display** the **details** of a specific account.
  - For this option, you have to ask user for the account number of the account s/he wants to view the details. After taking the input do the following.

    - **Find** the account from the Array/ArrayList
    - If the account is available in the list**,** pass the object to **print/println** method to show the details.

- '6' to **display** the **details** of all accounts.
  - For this option, do the following.

    - **Access** each of the account object from the Array/ArrayList
    - Print each object using the standard **print/println/printf** method.

- '0' to **exit** the system.

  - Come out of the loop if user chooses this option.

## **Problem#2:**

Update the **Online Store of Lab#3/4** to handle **multiple products**, where user can do the following

- Add new product to the system
- Update the price of a specific product
- View the discounted price of a specific product
- View the details of a specific product
- View the list of products with their details.