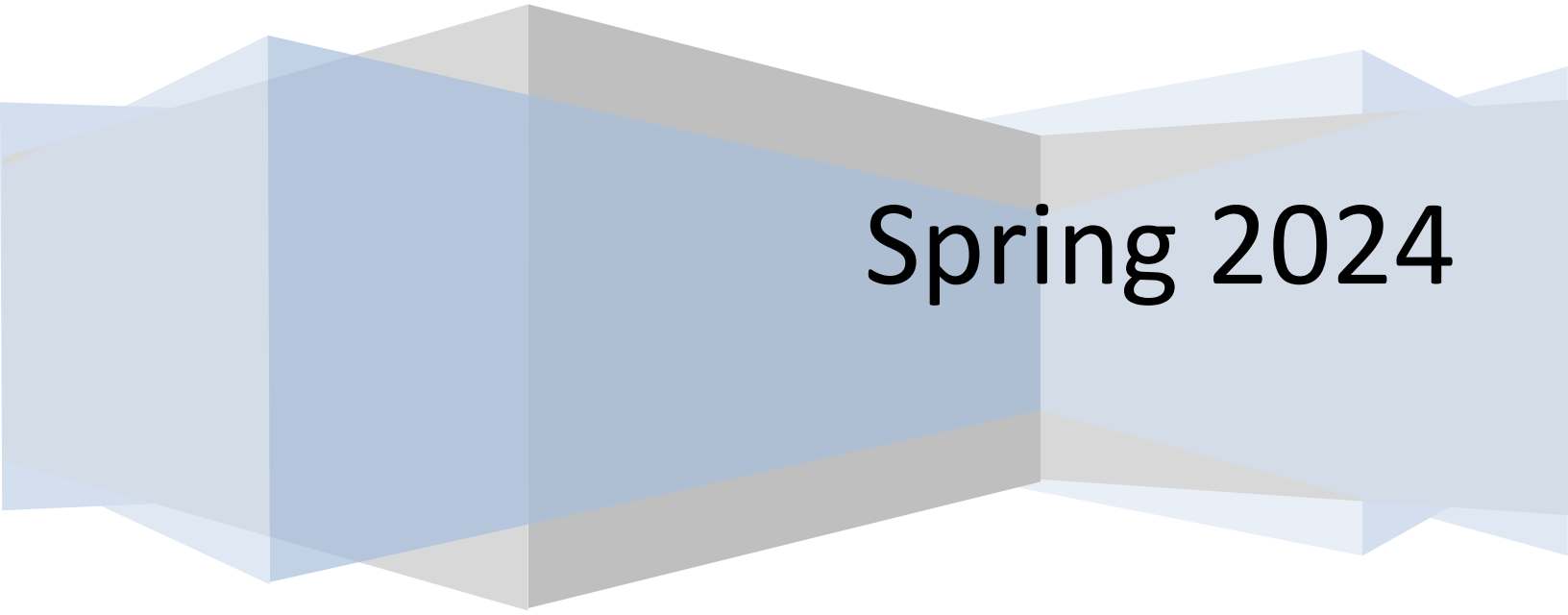
Northern University, Nowshera

Data members, Methods & access Modifiers/Specifiers

Week # 06 - Lecture 11- 12



Spring 2024

Learning Objectives:

1. **Instance variable**
2. **new keyword**
3. **Member Functions/Methods**
4. **Ways to initialize objects**
5. **Creating multiple objects by one type only**
6. **Access Modifiers**

1. Instance variables

A variable which is created inside the class but outside the method is known as an instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

1. Instance data members are those whose memory space is created each and every time whenever an object is created.

2. Instance data members are always meant for storing specific values.

3. Programmatically instance data members declaration should not be preceded by a keyword static.

Syntax: -

Data type v1, v2, v3 ……. Vn;

4. Each and every instance data member must be accessed with respective **object name**.

Ex: -

Objname.instance data member name

5. Instance data members are also known as **Object Level Data Members** because they depend on object name and independent from class name.

Ex: -

```
int pin;
int stno;
string sname;
```

Example:

```
public class Bicycle {

    int gear;
    int speed;

 // Method goes here

}
```

Instance Variables

2. new keyword

The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

2.1 Object and Class Example: main within the class

In this example, we have created a Student class which has two data members id and name. We are creating the object of the Student class by new keyword and printing the object's value.

Here, we are creating a main() method inside the class.

File: Student.java

```
1. class Student{
2. int id;
3. String name;
4.  public static void main(String args[]){
5.   Student s1=new Student();
6.   System.out.println(s1.id);
```

```
7.     System.out.println(s1.name);
8.    }
9.   }
```

Output:

```
0
null
```

2.2 Object and Class Example: main outside the class

In real time development, we create classes and use it from another class. It is a better approach than previous one. Let's see a simple example, where we are having main() method in another class.

We can have multiple classes in different Java files or single Java file. If you define multiple classes in a single Java source file, it is a good idea to save the file name with the class name which has main() method.

File: TestStudent1.java

```
1.//Java Program to demonstrate having the main method in
2.//another class
3.//Creating Student class.
4.class Student{
5. int id;
6. String name;
7.}
8.//Creating another class TestStudent1 which contains the main method
9.class TestStudent1{
10.       public static void main(String args[]){
11.        Student s1=new Student();
12.        System.out.println(s1.id);
13.        System.out.println(s1.name);
14.       }
15.      }
```

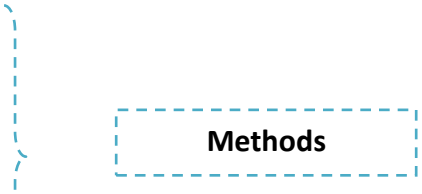
Output:

```
0
null
```

3. Member Functions/Method

In Java, a member function/method is like a function which is used to expose the behavior of an object. Members functions are defined with in class and can only be called with object name using dot operator as shown in given example.

```
public class Bicycle {

    int gear;
    int speed;

    int getGear() {
        //code goes here
    }

    setGear() {
      //code goes here
    }

    int getSpeed() {
        //code goes here
    }

    void applyBrake() {
       //code goes here
    }

    void speedUp) {
        //code goes here
    }
}
```



4. Ways to initialize object

There are 3 ways to initialize object in Java.

1. By reference variable
2. By method
3. By constructor

4.1) Object and Class: Initialization through reference

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

File: TestStudent2.java

```
1.  class Student{
2.   int id;
3.   String name;
4.  }
5.  class TestStudent2{
6.   public static void main(String args[]){
7.    Student s1=new Student();
8.    s1.id=101;
9.    s1.name="Ali Ahmed";
10.   System.out.println(s1.id+" "+s1.name);//printing members with a white space
11.  }
12. }
```

Output:

```
101 Ali Ahmed
```

We can also create multiple objects and store information in it through reference variable.

File: TestStudent3.java

```
1.  class Student{
2.   int id;
3.   String name;
4.  }
5.  class TestStudent3{
6.   public static void main(String args[]){
7.    //Creating objects
8.    Student s1=new Student();
9.    Student s2=new Student();
10.   //Initializing objects
11.   s1.id=101;
12.   s1.name="Ali";
13.   s2.id=102;
14.   s2.name="Asad";
15.   //Printing data
16.   System.out.println(s1.id+" "+s1.name);
17.   System.out.println(s2.id+" "+s2.name);
18.  }
19. }
```

Output:

```
101 Ali
102 Asad
```

4. 2) Object and Class: Initialization through method

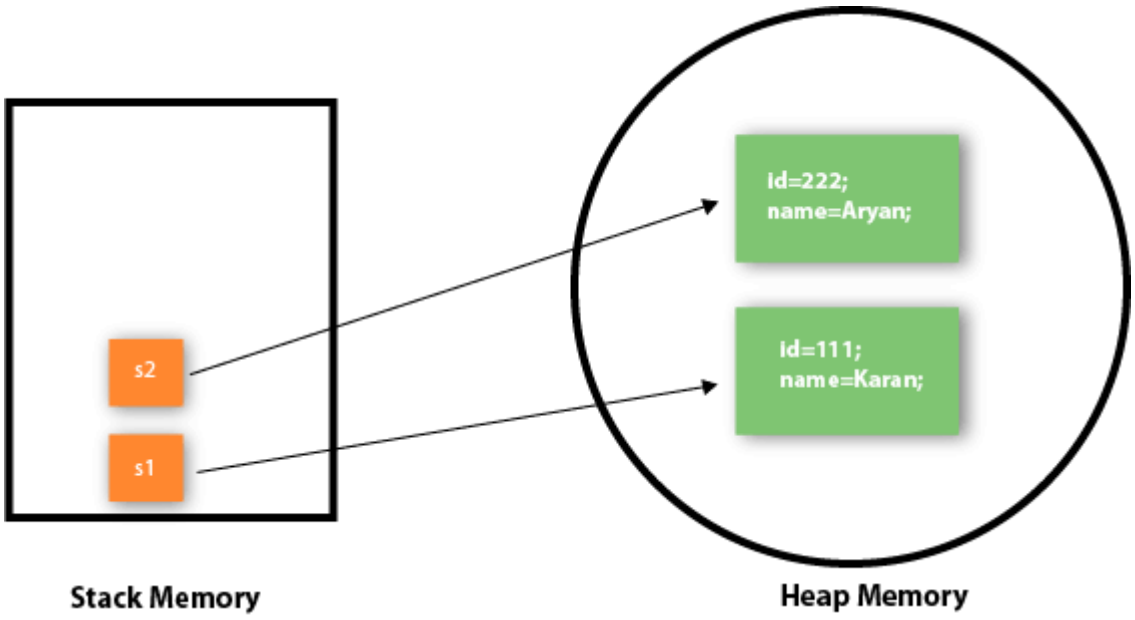
In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

File: TestStudent4.java

```
1.  class Student{
2.   int rollno;
3.   String name;
4.   void insertRecord(int r, String n){
5.    rollno=r;
6.    name=n;
7.   }
8.   void displayInformation(){System.out.println(rollno+" "+name);}
9.  }
10. class TestStudent4{
11.  public static void main(String args[]){
12.   Student s1=new Student();
13.   Student s2=new Student();
14.   s1.insertRecord(111,"Asad");
15.   s2.insertRecord(222,"Ali");
16.   s1.displayInformation();
17.   s2.displayInformation();
18.  }
19. }
```

Output:

```
111 Asad
222 Ali
```



As you can see in the above figure, object gets the memory in heap memory area. The reference variable refers to the object allocated in the heap memory area. Here, s1 and s2 both are reference variables that refer to the objects allocated in memory.

4.3) Initialization through a constructor

We will learn about constructors in Java later.

Examples:

Example #1. Object and Class: Employee

Let's see an example where we are maintaining records of employees.

File: TestEmployee.java

```
1.  class Employee{
2.     int id;
3.     String name;
4.     float salary;
5.     void insert(int i, String n, float s) {
6.        id=i;
7.        name=n;
```

```
8.          salary=s;
9.      }
10.     void display(){System.out.println(id+" "+name+" "+salary);}
11. }
12. public class TestEmployee {
13. public static void main(String[] args) {
14.     Employee e1=new Employee();
15.     Employee e2=new Employee();
16.     Employee e3=new Employee();
17.     e1.insert(101,"ajeet",45000);
18.     e2.insert(102,"irfan",25000);
19.     e3.insert(103,"Asad",55000);
20.     e1.display();
21.     e2.display();
22.     e3.display();
23. }
24. }
```

Output:

```
101 ajeet 45000.0
102 irfan 25000.0
103 Asad 55000.0
```

Example #2. Object and Class: Rectangle

There is given another example that maintains the records of Rectangle class.

File: TestRectangle1.java

```
1.  class Rectangle{
2.   int length;
3.   int width;
4.   void insert(int l, int w){
5.    length=l;
6.    width=w;
7.   }
8.   void calculateArea(){System.out.println(length*width);}
9.  }
10. class TestRectangle1{
11.  public static void main(String args[]){
12.   Rectangle r1=new Rectangle();
13.   Rectangle r2=new Rectangle();
14.   r1.insert(11,5);
```

```
15.    r2.insert(3,15);
16.    r1.calculateArea();
17.    r2.calculateArea();
18.  }
19.  }
```

Output:

```
55
45
```

5. Creating multiple objects by one type only

We can create multiple objects by one type only as we do in case of primitives.

Initialization of primitive variables:

```
1.  int a=10, b=20;
```

Initialization of refernce variables:

```
1.  Rectangle r1=new Rectangle(), r2=new Rectangle();//creating two objects
```

Let's see the example:

```
1.  //Java Program to illustrate the use of Rectangle class which
2.  //has length and width data members
3.  class Rectangle{
4.   int length;
5.   int width;
6.   void insert(int l,int w){
7.    length=l;
8.    width=w;
9.   }
10.  void calculateArea(){System.out.println(length*width);}
11. }
12. class TestRectangle2{
13.  public static void main(String args[]){
14.   Rectangle r1=new Rectangle(),r2=new Rectangle();//creating two objects
15.   r1.insert(11,5);
16.   r2.insert(3,15);
17.   r1.calculateArea();
```

```
18.   r2.calculateArea();
19. }
20. }
```

Output:

```
55
45
```

Real World Example: Account

File: TestAccount.java

```
1.  //Java Program to demonstrate the working of a banking-system
2.  //where we deposit and withdraw amount from our account.
3.  //Creating an Account class which has deposit() and withdraw() methods
4.  class Account{
5.  int acc_no;
6.  String name;
7.  float amount;
8.  //Method to initialize object
9.  void insert(int a,String n,float amt){
10. acc_no=a;
11. name=n;
12. amount=amt;
13. }
14. //deposit method
15. void deposit(float amt){
16. amount=amount+amt;
17. System.out.println(amt+" deposited");
18. }
19. //withdraw method
20. void withdraw(float amt){
21. if(amount<amt){
22. System.out.println("Insufficient Balance");
23. }else{
24. amount=amount-amt;
25. System.out.println(amt+" withdrawn");
26. }
27. }
28. //method to check the balance of the account
29. void checkBalance()
```

```
30. {System.out.println("Balance is: "+amount);}
31. //method to display the values of an object
32. void display(){System.out.println(acc_no+" "+name+" "+amount);}
33. }
34. //Creating a test class to deposit and withdraw amount
35. class TestAccount{
36. public static void main(String[] args){
37. Account a1=new Account();
38. a1.insert(832345,"Ali",1000);
39. a1.display();
40. a1.checkBalance();
41. a1.deposit(40000);
42. a1.checkBalance();
43. a1.withdraw(15000);
44. a1.checkBalance();
45. }
46. }
```

Output:

```
832345 Ali 1000.0
Balance is: 1000.0
40000.0 deposited
Balance is: 41000.0
15000.0 withdrawn
Balance is: 26000.0
```

6. Access Modifiers/Specifiers

The first (left-most) modifier used lets you control what other classes have access to a member field. For the moment, consider only **public** and **private**. Other access modifiers will be discussed later.

public modifier—the field is accessible from all classes.

private modifier—the field is accessible only within its own class.

In view of OOP concepts, it is common to make fields private and methods public. This means that they can only be *directly* accessed within Bicycle class and not from outside class. We still need access to these values, however. This can be done *indirectly* by adding public methods that obtain the field values for us.

Example: Use of access modifiers:

```
public class Bicycle {
    private int gear;
```

```
    private int speed;

    public int getGear() {
        return gear;
    }

    public void setGear(int newValue) {
        gear = newValue;
    }

    public int getSpeed() {
        return speed;
    }

    public void applyBrake(int decrement) {
        speed -= decrement;
    }

    public void speedUp(int increment) {
        speed += increment;
    }

}
class MainClass {
    public static void main(String args[]) {
        Bicycle b1 = new Bicycle();
        //b1.gear = 1; // not possible because of private
        b1.setGear(1);
        b1.speedUp(10);
        b1.setGear(2);
        b1.speedUp(10);
        b1.setGear(3);
        b1.speedUp(10);
        b1.setGear(2);
        b1.speedUp(30);
        b1.applyBrake(20);
        System.out.println("Speed is: "+b1.getSpeed());
        b1.setGear(2);
        System.out.println("Gear is: "+b1.getGear());
    }
}
```

Output:

```
Speed is: 40
Gear is: 2
```

Note: "Account class" data is public so we need to set their access modifier as private and functions as public. Another important point to consider is: "Default access status of java class attributes and functions is public"

Assignment # 4

Q#1: Write a program that will allow the user to input his/her name, user id, and password and perform following.

1. Show user details (Name, ID and Password)
2. Change User Password

Write required functions to complete given requirements. Try to secure direct access of attributes, so no one can update them without methods.

Note: User while updating user password, the system should ask for old password, match with existing password and only update if old password matches, otherwise print a message "INVALID ATTEMP" and allow re-entering password. After three invalid attempts, print a message "YOUR ACCOUNT IS IN RECOVERY MODE PLEASE CONTACT RECOVERY TEAM".

Q#2: You are required to write a system to maintain information of cases reported of Corona Virus in Pakistan. Consider the following requirements and implement the system.

1. *Save record of total reported cases (all Pakistan)*
2. *total of each province*
3. *total deaths*
4. *total recovered*

Write required functions and data members. Your program should answer given requirements.

Note: **Also write functions to add new patients or remove recovered or died patients from count whenever required.**