OOP-2

Objects(details)

Definition: An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

Class (details)

Def: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

A class in Java can contain:

- **Fields**
- **Methods**
- **Constructors**
- **Blocks**
- **Nested class and interface**

```
class <class_name>{
    field;
    method;
}
```

3 Ways to initialize object

There are 3 ways to initialize object in Java.

- By reference variable
- By method
- By constructor

1) Object and Class Example: Initialization through reference

```
class Student{
 int id;
 String name;
}
class TestStudent2{
 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Sonoo";
  System.out.println(s1.id+" "+s1.name);//printing members with a white space
 }
}
```

Continued…

2) Object and Class Example: Initialization through method

```
class Student{
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void displayInformation(){System.out.println(rollno+" "+name);}
}
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```

Continued…

3) Object and Class Example: Initialization through a constructor

```
class Demo{
     int  value1;
     int  value2;
     Demo(){
      value1 = 10;
      value2 = 20;
      System.out.println("Inside 1st Constructor");
    }
    Demo(int a){
     value1 = a;
     System.out.println("Inside 2nd Constructor");
   }
   Demo(int a,int b){
   value1 = a;
   value2 = b;
   System.out.println("Inside 3rd Constructor");
  }
```

Continued…

```
 public void display(){
     System.out.println("Value1 === "+value1);
     System.out.println("Value2 === "+value2);
  }
}

public static void main(String args[]){
    Demo d1 = new Demo();
    Demo d2 = new Demo(30);
    Demo d3 = new Demo(30,40);
    d1.display();
    d2.display();
    d3.display();
 }
```

Access Specifiers

Java Access Specifiers (also known as Visibility Specifiers) regulate access to classes, fields and methods in Java. These Specifiers determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class.

There are four types of Java access modifiers:

- **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Continued…

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Continued…

1. **Private**

```
class A{
private int data=40;
private void msg(){System.out.println("Hello java");}
}

public class Simple{
 public static void main(String args[]){
   A obj=new A();
   System.out.println(obj.data);//Compile Time Error
   obj.msg();//Compile Time Error
   }
}
```

Continued…

2. Default

```
//save by A.java
package pack;
class A{
  void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
import pack.*;
class B{
  public static void main(String args[]){
   A obj = new A();//Compile Time Error
   obj.msg();//Compile Time Error
  }
}
```

Continued…

3) Protected

```
//save by A.java
package pack;
public class A{
protected void msg(){System.out.println("Hello");}
}

//save by B.java
package mypack;
import pack.*;

class B extends A{
}
  public static void main(String args[]){
   B obj = new B();
   obj.msg();
  }
```

4. Public

```
package pack;
public class A{
public void msg(){System.out.println("Hello");}
}

package mypack;
import pack.*;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```