UNIT IV
INHERITANCE AND POLYMORPHISM

Constructors in Subclass

- A constructor in Java is similar to a method with a few differences. Constructor has the same name as the class name. A constructor doesn't have a return type.
- A Java program will automatically create a constructor if it is not already defined in the program. It is executed when an instance of the class is created.
- A constructor cannot be static, abstract, final or synchronized. It cannot be overridden.

Java has two types of constructors:

1. Default constructor
2. Parameterized constructor

What is the order of execution of constructor in Java inheritance?

While implementing inheritance in a Java program, every class has its own constructor. Therefore the execution of the constructors starts after the object initialization. It follows a certain sequence according to the class hierarchy. There can be different orders of execution depending on the type of inheritance.

Different ways of the order of constructor execution in Java

1. Order of execution of constructor in Single inheritance

In single level inheritance, the constructor of the base class is executed first.

OrderofExecution1.java

```
/* Parent Class */
class ParentClass
{
   /* Constructor */
   ParentClass()
   {
      System.out.println("ParentClass constructor executed.");
   }
}
```

```
/* Child Class */
class ChildClass extends ParentClass
{
   /* Constructor */
   ChildClass()
   {
      System.out.println("ChildClass constructor executed.");
   }
}

public class OrderofExecution1
{
   /* Driver Code */
   public static void main(String ar[])
   {
      /* Create instance of ChildClass */
      System.out.println("Order of constructor execution...");
      new ChildClass();
   }
}
```

Output:

```
Order of constructor execution...
ParentClass constructor executed.
ChildClass constructor executed.
```

In the above code, after creating an instance of ***ChildClass*** the ***ParentClass*** constructor is invoked first and then the ***ChildClass.***

2. Order of execution of constructor in Multilevel inheritance

In multilevel inheritance, all the upper class constructors are executed when an instance of bottom most child class is created.

OrderofExecution2.java

```
class College
{
   /* Constructor */
   College()
   {
      System.out.println("College constructor executed");
   }
}

class Department extends College
```

```
{
   /* Constructor */
   Department()
   {
      System.out.println("Department constructor executed");
   }
}

class Student extends Department
{
   /* Constructor */
   Student()
   {
   System.out.println("Student constructor executed");
   }
}
public class OrderofExecution2
{
      /* Driver Code */
   public static void main(String ar[])
   {
      /* Create instance of Student class */
      System.out.println("Order of constructor execution in Multilevel inheritance...");
      new Student();
   }
}
```

Output:

```
Order of constructor execution in Multilevel inheritance...
College constructor executed
Department constructor executed
Student constructor executed
```

In the above code, an instance of ***Student*** class is created and it invokes the constructors of ***College, Department*** and ***Student*** accordingly.

3. Calling same class constructor using this keyword

Here, inheritance is not implemented. But there can be multiple constructors of a single class and those constructors can be accessed using **this** keyword.

OrderofExecution3.java

```
public class OrderofExecution3
{
   /* Default constructor */
   OrderofExecution3()
```

```
    {
       this("CallParam");
       System.out.println("Default constructor executed.");
    }
    /* Parameterized constructor */
    OrderofExecution3(String str)
    {
       System.out.println("Parameterized constructor executed.");
    }
    /* Driver Code */
    public static void main(String ar[])
    {
       /* Create instance of the class */
       System.out.println("Order of constructor execution...");
       OrderofExecution3 obj = new OrderofExecution3();
    }
}
```

Output:

```
Order of constructor execution...
Parameterized constructor executed.
Default constructor executed.
```

In the above code, the parameterized constructor is called first even when the default constructor is called while object creation. It happens because ***this*** keyword is used as the first line of the default constructor.

4. Calling superclass constructor using super keyword

A child class constructor or method can access the base class constructor or method using the super keyword.

OrderofExecution4.java

```
/* Parent Class */
class ParentClass
{
    int a;
    ParentClass(int x)
    {
       a = x;
    }
}

/* Child Class */
class ChildClass extends ParentClass
{
    int b;
    ChildClass(int x, int y)
    {
        /* Accessing ParentClass Constructor */
        super(x);
        b = y;
    }
    /* Method to show value of a and b */
    void Show()
    {
        System.out.println("Value of a : "+a+"\nValue of b : "+b);
    }
}

public class OrderofExecution4
{
    /* Driver Code */
    public static void main(String ar[])
    {
        System.out.println("Order of constructor execution...");
        ChildClass d = new ChildClass(79, 89);
        d.Show();
    }
}
```

Output:

```
Order of constructor execution...
Value of a : 79
Value of b : 89
```

In the above code, the ***ChildClass*** calls the ***ParentClass*** constructor using a ***super*** keyword that determines the order of execution of constructors.