Constructor

Constructors in Java

In Java, a constructor is a block of codes similar to the method.

It is called when an instance of the class is created.

At the time of calling constructor, memory for the object is

allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword,

at least one constructor is called.

Rules for creating Java constructor

There are two rules defined for the constructor.

1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Difference between Constructor and Method:

Java Constructor	**Java Method**
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.

The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor name must be same as the class name.	The method name may or may not be same as the class name.

Types of constructors

1. Default Constructor

A constructor is called "Default Constructor" when it doesn't have any parameter.

Syntax of default constructor:

```
<class_name>()
{
}
```

2. Parameterized Constructor

A constructor which has a specific number of parameters is called a parameterized constructor.

The parameterized constructor is used to provide different values to distinct objects. However, you can provide the same values also.

```
Class class_name
{
int a,b;
<class_name>( int x, int y)
{
a=x;
b=y;
}
```

3. Copy constructor

There is no copy constructor in Java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in Java.

They are:

- By constructor
- By assigning the values of one object into another
- By clone() method of Object class

Example :

```
import java.util.*;

 public class constructor1

{

int a,b;


// Default Constructor


 constructor1()

{

System.out.println("Default");

a=0;

}
```

```
// Parameterized Constructor

constructor1(int x)
{
System.out.println("Parameterized Constructor");
a=x;
}


constructor1(int m,int n)
{
System.out.println("Parameterized Constructor");
a=m;
b=n;
}


// Copy Constructor
```

```
constructor1(constructor1 c)

{

        System.out.println("Copy constructor called");

        a = c.a;

        b = c.b;


}


void display()

{


System.out.println(a);

System.out.println(b);

}


public static void main (String args[])

{

constructor1 c1= new constructor1();

constructor1 c2= new constructor1(30);

constructor1 c3= new constructor1(10,20);
constructor1 c4 = new constructor1 (c2);

constructor1 c5 = new constructor1 (c3);

c1.display();

c2.display();

c3.display();

c4.display();

c5.display();

}

}
```

4. Constructor Overloading

In Java, a constructor is just like a method but without return type.

It can also be overloaded like Java methods.

It is a technique of having more than one constructor with different parameter lists. They are arranged in a way that each constructor performs a different task. They are differentiated by the compiler by the number of parameters in the list and their types.

Example of Constructor Overloading

```
class Student5{
    int id;
    String name;
    int age;
    //creating two arg constructor
    Student5(int i,String n){
    id = i;
    name = n;
    }
    //creating three arg constructor
    Student5(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
```

```
        void display(){System.out.println(id+" "+name+" "+age);}

        public static void main(String args[]){
        Student5 s1 = new Student5(111,"Karan");
        Student5 s2 = new Student5(222,"Aryan",25);
        s1.display();
        s2.display();
       }
     }
```

Output:

```
111 Karan 0
222 Aryan 25
```