Java Constructors

In Java, constructors play an important role in object creation. A constructor is a special block of code that is called when an object is created. Its main job is to initialize the object, to set up its internal state, or to assign default values to its attributes. This process happens automatically when we use the "new" keyword to create an object.

Characteristics of Constructors:

- **Same Name as the Class: A constructor has the same name as the class in which it is defined.**
- **No Return Type: Constructors do not have any return type, not even void. The main purpose of a constructor is to initialize the object, not to return a value.**
- **Automatically Called on Object Creation: When an object of a class is created, the constructor is called automatically to initialize the object's attributes.**
- **Used to Set Initial Values for Object Attributes: Constructors are primarily used to set the initial state or values of an object's attributes when it is created.**

Example: This program demonstrates how a constructor is automatically called when an object is created in Java.

```
// Java Program to demonstrate
// Constructor usage
import java.io.*;
// Driver Class
class Speed {
  // Constructor
  Speed()
  {
    System.out.println("Constructor Called");
  }
  // main function
  public static void main(String[] args)
  {
    Speed geek = new Speed();
  }
}
```

Why Do We Need Constructors in Java

Constructors play a very important role, it ensures that an object is properly initialized before use.

What happens when we don't use constructors:

Without constructors:

- **Objects might have undefined or default values.**
- **Extra initialization methods would be required.**
- **Risk of improper object state**

Types of Constructors in Java

Now is the correct time to discuss the types of the constructor, so primarily there are three types of constructors in Java are mentioned below:

- **Default Constructor**
- **Parameterized Constructor**
- **Copy Constructor**

1. Default Constructor in Java

A constructor that has no parameters is known as default constructor. A default constructor is invisible. And if we write a constructor with no arguments, the compiler does not create a default constructor. Once you define a constructor (with or without parameters), the compiler no longer provides the default constructor. Defining a parameterized constructor does not automatically create a no-argument constructor, we must explicitly define if needed. The default constructor can be implicit or explicit.

- **Implicit Default Constructor: If no constructor is defined in a class, the Java compiler automatically provides a default constructor. This constructor doesn't take any parameters and initializes the object with default values, such as 0 for numbers, null for objects.**
- **Explicit Default Constructor: If we define a constructor that takes no parameters, it's called an explicit default constructor. This constructor replaces the one the compiler would normally create automatically. Once you define any constructor (with or without parameters), the compiler no longer provides the default constructor for you.**

2. Parameterized Constructor in Java

A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with our own values, then use a parameterized constructor.

Remember: Does constructor return any value?

There are no "return value" statements in the constructor, but the constructor returns the current class instance. We can write 'return' inside a constructor.

3. Copy Constructor in Java

Unlike other constructors copy constructor is passed with another object which copies the data available from the passed object to the newly created object.

Note: Java does not provide a built-in copy constructor like C++. We can create our own by writing a constructor that takes an object of the same class as a parameter and copies its fields.