Constructors in Java

Difficulty Level : Easy ● Last Updated : 09 Feb, 2022

Java constructors or constructors in Java is a terminology been used to construct something in our programs. A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.

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 the 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.

> ***Note:*** *It is not necessary to write a constructor for a class. It is because java compiler creates a default constr uctor if your class doesn't have any.*

times.

Now let us come up with the syntax for the constructor being invoked at the time of object or instance creation.

```
class Geek
{
  .......

  // A Constructor
  new Geek() {}

  .......
}

// We can create an object of the above class
// using the below statement. This statement
// calls above constructor.
Geek obj = new Geek();
```

Need of Constructor

done by the programmer or by Java itself (default constructor).

When is a Constructor called?

Each time an object is created using a **new()** keyword, at least one constructor (it could be the default constructor) is invoked to assign initial values to the **data members** of the same class.

The rules for writing constructors are as follows:

- Constructor(s) of a class must have the same name as the class name in which it resides.
- A constructor in Java can not be abstract, final, static, or Synchronized.
- Access modifiers can be used in constructor declaration to control its access i.e which other class can call the constructor.

So by far, we have learned constructors are used to initializing the object's state. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation.

Types of Constructors in Java

Now is the correct time to discuss types of the constructor, so primarily there are two types of constructors in java:

- No-argument constructor
- Parameterized Constructor

Note: *Default constructor provides the default values to the object like 0, null, etc. depending on the type.*

Example:

Java

```
// Java Program to illustrate calling a
// no-argument constructor

import java.io.*;

class Geek {
    int num;
    String name;

    // this would be invoked while an object
    // of that class is created.
    Geek() { System.out.println("Constructor called"); }
}

class GFG {
    public static void main(String[] args)
    {
```

```
null
0
```

2. Parameterized Constructor

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.

Example:

Java

```
// Java Program to Illustrate Working of
// Parameterized Constructor

// Importing required inputoutput class
import java.io.*;
```

```
    // Object of that class created
    Geek(String name, int id)
    {
        this.name = name;
        this.id = id;
    }
}

// Class 2
class GFG {
    // main driver method
    public static void main(String[] args)
    {
        // This would invoke the parameterized constructor.
        Geek geek1 = new Geek("adam", 1);
        System.out.println("GeekName :" + geek1.name
                           + " and GeekId :" + geek1.id);
    }
}
```

Output

```
GeekName :adam and GeekId :1
```

Now the most important topic that comes into play is the strong incorporation of OOPS with constructors known as constructor overloading. JustLike methods, we can overload constructors for creating objects in different ways. Compiler differentiates constructors on the basis of numbers of parameters, types of the parameters, and order of the parameters.

Example:

Java

```
// Java Program to illustrate constructor overloading
// using same task (addition operation ) for different
// types of arguments.

import java.io.*;

class Geek
{
    // constructor with one argument
    Geek(String name)
    {
        System.out.println("Constructor with one " +
                           "argument - String : " + name);
    }

    // constructor with two arguments
```

```
    // Constructor with one argument but with different
    // type than previous..
    Geek(long id)
    {
        System.out.println("Constructor with one argument : " +
                                            "Long : " + id);
    }
}

class GFG
{
    public static void main(String[] args)
    {
        // Creating the objects of the class named 'Geek'
        // by passing different arguments

        // Invoke the constructor with one argument of
        // type 'String'.
        Geek geek2 = new Geek("Shikhar");

        // Invoke the constructor with two arguments
        Geek geek3 = new Geek("Dharmesh", 26);

        // Invoke the constructor with one argument of
        // type 'Long'.
        Geek geek4 = new Geek(325614567);
    }
```

In order to know to deep down into constructors there are two concepts been widely used as listed below:

- Constructor Chaining
- Copy constructor

This article is contributed by **Nitsdheerendra**. If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Previous

Need of Wrapper Classes in Java

Next

Copy Constructor in Java

RECOMMENDED ARTICLES

Page : 1 2 3

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : deepak_jain, Kirti_Mangal, kushvillia, solankimayank, anikakapoor, nishkarshgandhi

Article Tags : Java, School Programming

Practice Tags : Java

Improve Article | Report Issue

Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org