**Constructor** is a block of code that initializes the newly created object. A constructor resembles an instance method in java but it’s not a method as it doesn’t have a return type. In short constructor and method are different(More on this at the end of this guide). People often refer constructor as special type of method in Java.

Constructor has same name as the class and looks like this in a java code.

public class MyClass{

//This is the constructor

MyClass(){

}

..

}

**Need of Constructor**

Think of a Box. If we talk about a box class then it will have some class variables (say length, breadth, and height). But when it comes to creating its object(i.e Box will now exist in computer’s memory), then can a box be there with no value defined for its dimensions. The answer is no.

So constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor).

**When is a Constructor called ?**

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

A constructor is invoked at the time of object or instance creation. For Example:

class A

{

.......

// A Constructor

A() {}

.......

}

// We can create an object of the above class

// using the below statement. This statement

// calls above constructor.

A obj = new A();

**Rules for writing Constructor:**

Constructor(s) of a class must has same name as the class name in which it resides.

A constructor in Java can not be abstract, final, static and Synchronized.

Access modifiers can be used in constructor declaration to control its access i.e which other class can call the constructor.

**Types of constructor**

There are two type of constructor in Java:

**No-argument constructor**: A constructor that has no parameter is known as default constructor. If we don’t define a constructor in a class, then compiler creates default constructor(with no arguments) for the class. And if we write a constructor with arguments or no-arguments then the compiler does not create a default constructor.

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

// no-argument constructor

class A

{

int num;

String name;

// this would be invoked while an object

// of that class is created.

A()

{

System.out.println("Constructor called");

}

}

class Demo

{

public static void main (String[] args)

{

// this would invoke default constructor.

A x = new A();

// Default constructor provides the default

// values to the object like 0, null

System.out.println(x.name);

System.out.println(x.num);

}

}

Output :

Constructor called

null

0

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

// parameterized constructor.

import java.io.\*;

class A

{

// data members of the class.

String name;

int id;

// constructor would initialize data members

// with the values of passed arguments while

// object of that class created.

A(String name, int id)

{

this.name = name;

this.id = id;

}

}

class Demo

{

public static void main (String[] args)

{

// this would invoke the parameterized constructor.

A x = new A("adam", 1);

System.out.println("Name :" + x.name +

" and Id :" + x.id);

}

}

Output:

Name :adam and Id :1

**Does constructor return any value?**

There are no “return value” statements in constructor, but constructor returns current class instance. We can write ‘return’ inside a constructor.

**Constructor Overloading**

Like 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.

// Java Program to illustrate constructor overloading

// using same task (addition operation ) for different

// types of arguments.

class A

{

// constructor with one argument

A(String name)

{

System.out.println("Constructor with one " +

"argument - String : " + name);

}

// constructor with two arguments

A(String name, int age)

{

System.out.println("Constructor with two arguments : " +

" String and Integer : " + name + " "+ age);

}

// Constructor with one argument but with different

// type than previous..

A(long id)

{

System.out.println("Constructor with one argument : " +

"Long : " + id);

}

}

class Demo

{

public static void main(String[] args)

{

// Creating the objects of the class named A

// by passing different arguments

// Invoke the constructor with one argument of

// type 'String'.

A x= new A("raj");

// Invoke the constructor with two arguments

A y = new A("rahul", 10);

// Invoke the constructor with one argument of

// type 'Long'.

A z= new A(325614567);

}

}

Output:

Constructor with one argument - String : raj

Constructor with two arguments - String and Integer : rahul 10

Constructor with one argument - Long : 325614567

**How constructors are different from methods in Java?**

Constructor(s) must have the same name as the class within which it defined while it is not necessary for the method in java.

Constructor(s) do not return any type while method(s) have the return type or void if does not return any value.

Constructor is called only once at the time of Object creation while method(s) can be called any numbers of time.