



SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai

Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &

Accredited by NBA (B.E - CSE, EEE, ECE, Mech & B.Tech .IT)

COIMBATORE-641 035, TAMIL NADU



UNIT III OBJECT AND CLASSES

CONSTRUCTORS

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.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

Rules for creating Java constructor

There are two rules defined for the constructor.

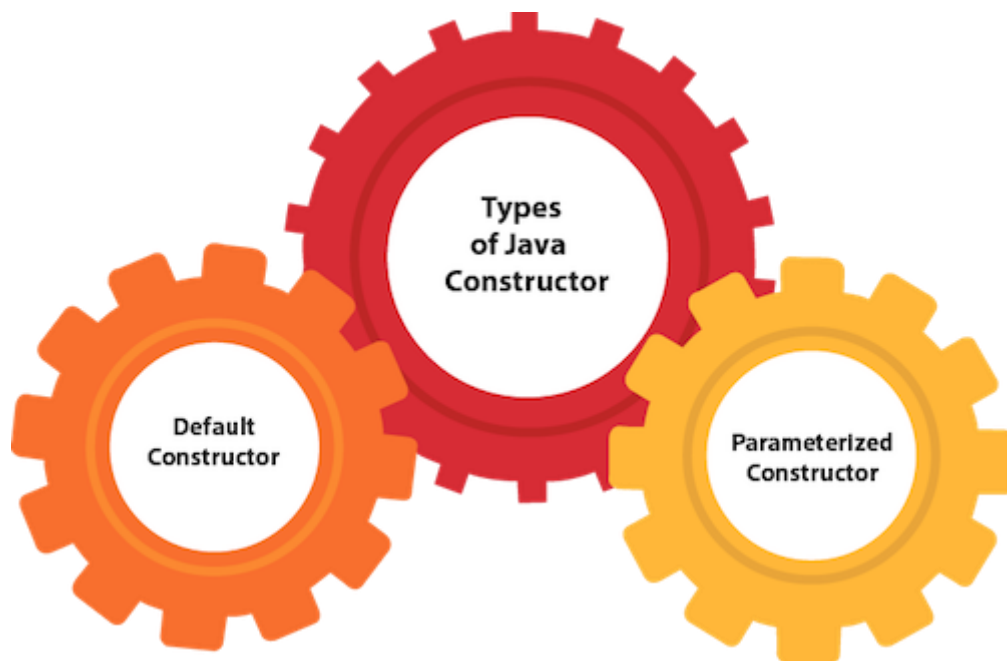
1. Constructor name must be the same as its class name
2. A Constructor must have no explicit return type
3. A Java constructor cannot be abstract, static, final, and synchronized

Note: We can use [access modifiers](#) while declaring a constructor. It controls the object creation. In other words, we can have private, protected, public or default constructor in Java.

Types of Java constructors

There are two types of constructors in Java:

1. Default constructor (no-arg constructor)
2. Parameterized constructor



Java Default Constructor

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

Syntax of default constructor:

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

Example of default constructor

//Java Program to create and call a default constructor

```
class Bike1{
```

```
//creating a default constructor
```

```

Bike1(){System.out.println("Bike is created");}
//main method
public static void main(String args[]){
//calling a default constructor
Bike1 b=new Bike1();
}
}

```

Test it Now

Output:

Bike is created

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.



Q) What is the purpose of a default constructor?

The default constructor is used to provide the default values to the object like 0, null, etc., depending on the type.

Example of default constructor that displays the default values

```

//Let us see another example of default constructor
//which displays the default values
class Student3{
int id;

```

```
String name;  
//method to display the value of id and name  
void display(){System.out.println(id+" "+name);}
```

```
public static void main(String args[]){  
//creating objects  
Student3 s1=new Student3();  
Student3 s2=new Student3();  
//displaying values of the object  
s1.display();  
s2.display();  
}  
}
```

Output:

```
0 null  
0 null
```

Explanation:In the above class,you are not creating any constructor so compiler provides you a default constructor. Here 0 and null values are provided by default constructor.

Java Parameterized Constructor

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

Why use the parameterized constructor?

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

Example of parameterized constructor

In this example, we have created the constructor of Student class that have two parameters. We can have any number of parameters in the constructor.

//Java Program to demonstrate the use of the parameterized constructor

```
.  
class Student4{  
    int id;  
    String name;  
    //creating a parameterized constructor  
    Student4(int i,String n){  
        id = i;  
        name = n;  
    }  
    //method to display the values  
    void display(){System.out.println(id+" "+name);}  
  
    public static void main(String args[]){  
        //creating objects and passing values  
        Student4 s1 = new Student4(111,"Karan");  
        Student4 s2 = new Student4(222,"Aryan");  
        //calling method to display the values of object  
        s1.display();  
        s2.display();  
    }  
}
```

Output:

```
111 Karan  
222 Aryan
```

Constructor Overloading in Java

In Java, a constructor is just like a method but without return type. It can also be overloaded like Java methods.

Constructor overloading in Java 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

//Java program to overload constructors

```
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

Difference between constructor and method in Java

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.

Java 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++.

Advertisement

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

In this example, we are going to copy the values of one object into another using Java constructor.

//Java program to initialize the values from one object to another object.

```
class Student6{
    int id;
    String name;
    //constructor to initialize integer and string
    Student6(int i,String n){
        id = i;
        name = n;
    }
    //constructor to initialize another object
    Student6(Student6 s){
        id = s.id;
        name =s.name;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
        Student6 s1 = new Student6(111,"Karan");
        Student6 s2 = new Student6(s1);
        s1.display();
        s2.display();
    }
}
```



```
} }
```

Output:

```
111 Karan  
111 Karan
```

Copying values without constructor

We can copy the values of one object into another by assigning the objects values to another object.

```
class Student7{  
    int id; String name;  
    Student7(int i,String n){  
        id = i; name = n;  
    }  
    Student7(){ }  
    void display(){System.out.println(id+" "+name);}  
    public static void main(String args[]){  
        Student7 s1 = new Student7(111,"Karan");  
        Student7 s2 = new Student7();  
        s2.id=s1.id;  
        s2.name=s1.name;  
        s1.display(); s2.display();  
    } }
```

Output:

```
111 Karan
```

```
111 Karan
```

