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.

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.

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

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:

```
1. <class_name>(){}
2.
```

Example of default constructor

```
1. //Java Program to create and call a default constructor
2. class Bike1{
3. //creating a default constructor
4. Bike1()
5. {
6. System.out.println("Bike is created");
```

```
7.  }
8.  //main method
9.  public static void main(String args[])
10. {
11. //calling a default constructor
12. Bike1 b=new Bike1();
13. }
14. }
```

Example of default constructor that displays the default values

```
1.  //Let us see another example of default constructor
2.  //which displays the default values
3.  class Student3
4.  {
5.  int id;
6.  String name;
7.  //method to display the value of id and name
8.  void display(){System.out.println(id+" "+name);
9.  }
10.
11. public static void main(String args[])
12. {
13. //creating objects
14. Student3 s1=new Student3();
15. Student3 s2=new Student3();
16. //displaying values of the object
17. s1.display();
18. s2.display();
19. }
20. }
```

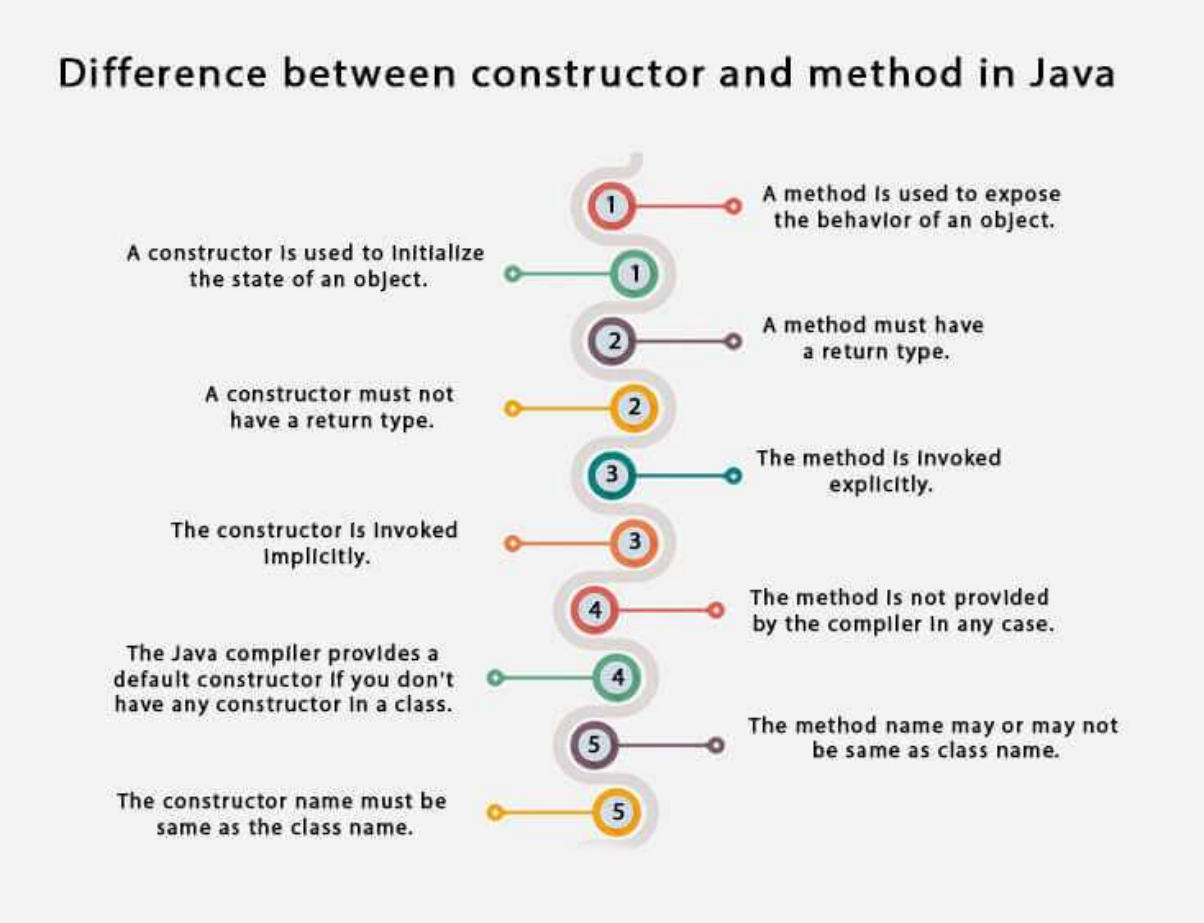
Java Parameterized Constructor

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.

```
//Java Program to demonstrate the use of the parameterized constructor.
1.  class Student4{
2.      int id;
3.      String name;
4.      Student4(int i,String n)
5.  {
6.      id = i;
7.      name = n;
8.      }
9.      void display(){System.out.println(id+" "+name);
10. }
11.     public static void main(String args[])
12. {
13.     Student4 s1 = new Student4(111,"Karan");
14.     Student4 s2 = new Student4(222,"Aryan");
15.     s1.display();
16.     s2.display();
17.    }
18. }
```



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++.

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

```
class Student6
{
   int id;
   String name;

   Student6(int i,String n)
{
```

```
    id = i;
    name = n;
    }
        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();
   }
}
```

Copying values without constructor

```
1.  class Student7{
2.      int id;
3.      String name;
4.      Student7(int i,String n){
5.      id = i;
6.      name = n;
7.      }
8.      Student7(){}
9.      void display(){System.out.println(id+" "+name);}
10.
11.     public static void main(String args[]){
12.     Student7 s1 = new Student7(111,"Karan");
13.     Student7 s2 = new Student7();
```

```
14.     s2.id=s1.id;
15.     s2.name=s1.name;
16.     s1.display();
17.     s2.display();
18.    }
19. }
```