Constructors

What is a Constructor?
The first question that arises in your mind.
In Simple words, we can say Constructor is a special type of method that we use to initialize an Object.
We create a Constructor by always using the Class Name. It has some similarity like methods, but it doesn't have return type like methods. If we add a return type in a Constructor then it acts like methods.

Why is the Constructor used?
Constructor is used because it provides Code Reusability.

Let us understand with an example to understand how to construct a constructor --

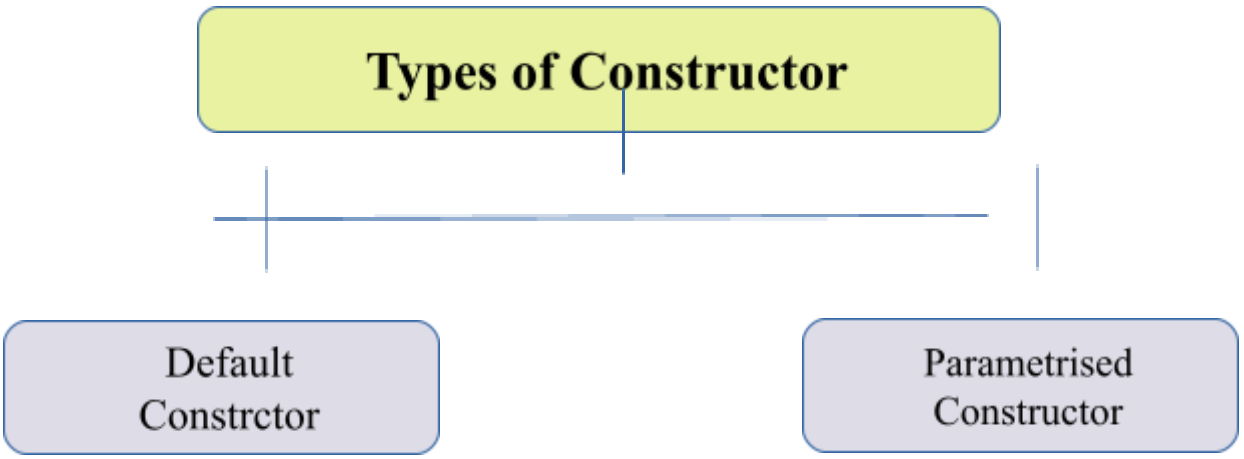
```java
public class animals {
    // here animals() is a constructor
    animals() {
        System.out.println("\n This is the Example of Constructor");
        // the above statement will automaticall execute when we make object
    }
    Run | Debug
    public static void main(String[] args) {
        // Now we have to call the Constructor by making an Object
        animals obj = new animals();
    }
}
```

Output --

This is the Example of Constructor

So in the Previous example we can see, we create a constructor name as class name, then we simply write a print statement inside the constructor then I make an Object of that Constructor in the main class, and the statement inside the Constructor is Printed default.

In Java, there are two types of Constructor ---



Default Constructor -

When you don't give any parameter or argument to the Constructor then that Constructor is known as Default Constructor.

Did you know, if you don't create any Constructors in your Class then at the time of Compilation, Compiler automatically creates a default Constructor.

Parametrised Constructor -

When you pass any argument or parameter in the Constructor then that Constructor is known as Parameterized Constructor.

Let's understand more by a Simple Example -

```
public class animals {
    String name;
    String type;
    public animals(String n, String t) {
        name = n;
        type = t;
        System.out.println("\n Animal Name is -> "+name);
        System.out.println(" Animal Belongs to -> "+type);
    }
    Run | Debug
    public static void main(String[] args) {
        animals obj = new animals("Cow","Carnivorus");
    }
}
```

Output --

Animal Name is -> Cow

Animal Belongs to -> Carnivorous

Constructor Overloading --

Constructor Overloading is Similar to Method Overloading.

In Constructor Overloading we create multiple Constructor and pass different types of data-types.

Points of constructor overloading:

It is always necessary to pass parameters in the Constructor for Constructor Overloading.

When we pass different arguments then the compiler identifies a constructor on the basis of data-types that we provided at a time of making an object and declaring the constructor.

Lets understand with a simple example --

```java
public class students {
    int enroll;
    String name;
    String batch;
    String branch;

    students(int e, String n) {
        enroll = e;
        name = n;
    }
    students(String ba, String br) {
        batch = ba;
        branch = br;
    }
    void display() {
        System.out.println("\n Student Name is -> "+name);
        System.out.println(" Student Enrollnment No is -> "+enroll);
    }
    void display1() {
        System.out.println("\n Student Belongs to "+batch+" Batch");
        System.out.println(" Student Belongs to "+branch+" Branch");
    }
    Run | Debug
    public static void main(String[] args) {
        students obj = new students(159, "Aman");
        students obj1 = new students("C", "BCA");
        obj.display();
        obj1.display1();
    }
}
```

Output --

```
Student Name is -> Aman
Student Enrollnment No is -> 159

Student Belongs to C Batch
Student Belongs to BCA Branch
```

Note:-

At the time of Object Creation, we create or call a Constructor after using a new keyword.

From Previous Example, animals obj = new animals();

So here animals() is a constructor, and we are creating it by using a new keyword as default.