* Constructor -

A constructor is a special type of member function that is called automatically when an object is created.

→ Constructor is required for initialization of properties at the time of construction of an object.

→ Every class will have a default constructor provided by java compiler.

→ Constructor will not have any return type.

Syntax :

```
Public constructorName (arg/Noarg)
{
    // Body of constructor
}
```

* Rules for defining Constructor -

- Constructor can be public, private, protected or default.
- Constructor can not be static, non-static, final or abstract.
- Constructor name must be same as that of classname
- It does not have any return type not even void.
- Constructor can be with arguments or without arguments.

* There are two types of constructors -

i) Parameterized Constructor

ii) Non-parameterized Constructor

i) Non-parameterized Constructor - (Default Constructor)

If a constructor does not have any argument it is called as non-parameterized constructor.

Eg:-
```
Class Person
{
    String name;
    int age;
    Public Person()   // Non-parameterized Constructor
    {
        name = "Abhi";
        age = 19;
    }
}
Class Test
{
    Public Static void main(String arg[])
    {
        Person P1 = new Person();
        S.o.p("Name :" + P1.name + " Age :" + P1.age);
        Person P2 = new Person();
        S.o.p("Name :" + P2.name + "Age :" + P2.age);
    }
}
```

o/p - Name: Abhi Age: 19

Name: Abhi Age: 19

* One of the drawbacks of Non-parameterized constructor is, it provides same value for every object.
* Therefore to overcome this drawback, we go for parameterized constructor.

ii) Parameterized Constructor -

If a constructor has one or more parameters, it is called 'Parameterized Constructor'.

Eg:-

```
class Person
{
    String name;
     int age;

   Public Person(String name , int age)
     {
         this.name = name;
         this.age = age;
     }
}
class Test
{   P. s. v. main (String a[]) {
   Person P1 = new Person ("Mohan", 25);
    S.o.P( P1.name + " " + P1.age);

   Person P2 = new Person ("Abhi", 19);
    S.o.P( P2.name + " " + P2.age);
  }
}
```

o/p - Mohan 25
Abhi 19

Q. When a constructor called, before or after creating the object?

A constructor is called concurrently when the object creation is goin on. JVM first allocates memory for the obj and then executes the constructor to initialize the instance variable. By the time, object creation is completed, the constructor execution is also completed.

* Difference b/w Constructor and Method -

Java Constructor	Java Method
1. Constructor is used to initialize the state of an object.	1. Method is used to expose behaviour of an object.
2. Constructor must not have return type.	2. Method must have return type.
3. Constructor is invoke implicitly	3. Method is invoked explicitly
4. The java compiler provides a default constructor if you don't have any constructor.	4. Method is not provided by compiler in any case.
5. Constructor name must be same as the class name.	5. Method name may or may not be same as class name.

* 'this' Keyword -

If there is any name conflict b/w the parameters and property (member of the class) then to refer the properties or the data members of the class, we use 'this' keyword.

Eg:-

```
class Rectangle
{
    int length;
    int breadth;
Rectangle (int length, int breadth)
    {
        this.length = length;
        this.breadth = breadth;
    }
    void display()
    {
        S.o.p("Length : " + this.length);
        S.o.p("Breadth : " + this.breadth);
    }
}
Public class Test
{
    Public staic void main(String a[])
    {
        Rectangle r1 = new Rectangle(10, 5);
        r1.display();
    }
}
```

(Here parameters and data members have same name that's why we are using 'this')

* Constructor Overloding -

Constructor overloading is a feature that allows a class to have multiple constructors with different parameter list.

→ In java, constructor overloading is achieved by creating multiple constructors with different parameter list.

Eg:-

```
class Person
{
    String name;
    int age;

    Person()    // Non-parameterized (No-argument)
    {
        this.name = "Unknown";
        this.age = 0;
    }

    Person(String name)  // Parameterized constructor with one argument
    {
        this.name = name;
        this.age = 0;
    }

    Person(String name, int age)  // Parameterized constructor with two argument.
    {
        this.name = name;
        this.age = age;
    }

    void printDetails()
    {
        S.o.p("Name:" + this.name);
        S.o.p("Age :" + this.age);
    }
}
```

```
Public class Test
{
    Public static void main(String a[])
    {
        Person P1 = new Person();
        Person P2 = new Person("Abhishek");
        Person P3 = new Person("Abhi", 19);
        S.o.p(P1.printDetails());
        S.o.p(P2.printDetails());
        S.o.p(P3.PrintDetails());
```

It will use no-argument constructor

Uses the constructor with one argument

Uses the constructor with two argument