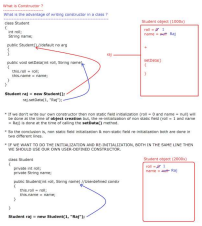

What is Constructor ?

What is the advantage of writing constructor in a class ?

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

public Student(){//default no arg
}

public void setData(int id, String name)
{
    this.id = id;
    this.name = name;
}
}

Student raj = new Student();
        raj.setData(1, "Raj");
```

Student object (1000x)

```
id = 0
name = null
raj

setData()
{
}
```

* If we don't write our own constructor then non static field initialization (id = 0 and name = null) will be done at the time of **object creation** but, the re-initialization of non static field (id = 1 and name = Raj) is done at the time of calling the **setData()** method.

* So the conclusion is, non static field initialization & non static field re-initialization both are done in two different lines.

* If WE WANT TO DO THE INITIALIZATION AND RE-INITIALIZATION, BOTH IN THE SAME LINE THEN WE SHOULD USE OUR OWN USER-DEFINED CONSTRUCTOR.

```
class Student
{
    private int id;
    private String name;

    public Student(int id, String name){//Userdefined constr
        this.id = id;
        this.name = name;
    }

}

Student raj = new Student(1, "Raj");
```

Student object (3000x)

```
id = 0
name = null
raj
```

* When we write our own user-defined constructor then non static field initialization (id = 0 and name = null) and non static field re-initialization both are done IN THE SAME LINE i.e at the time of creating the object.

Definition of Constructor :

* It is used to construct the object that is the reason it is called Constructor .

* Whenever the name of the class and name of the method, both are exactly same and it does not contain any return type then it is called Constructor.

Example:
```
public class Student
{
    public Student()
    {
    }
}
```

*** The main purpose of constructor is initialise the object [Initialising the non static fields]**

* A constructor never contain any return type including void also.

```
class Student
{
    public void Student() //Method
    {
        SO.println("Method");
    }
}
class Test
{
    void main()
    {
        Student st = new Student();
        //( Student();
    }
}
```

* A constructor may contain return statement but return statement with value.

```
class Student
{
    public Student()
    {
        SO.println("User-defined constructor");
        return ;//Valid
    }
}
```

* We need to call the method explicitly but a constructor will be automatically called & executed at the time of creating the object.

Types of Constructor :

* In java we have 3 types of constructor

  1) default no-argument Constructor [Automatically added by java compiler]

  2) Non parameterized OR zero argument OR Parameter less constructor [Written by user without parameter]

  3) Parameterized Constructor.

1) Default No Argument Constructor :

Whenever we write a class and if we don't write any type of constructor then automatically one default no argument constructor is added by the compiler.

The access modifier of default no argument constructor would be same as class access modifier.

It does not accept any parameter.

Example.java
```
public class Example
{

}
```

javac Example.java

Example.class
```
public class Example
{
    public Example(){//default no argument constructor
    }
}
```

2) Non parameterized OR zero argument OR parameter less constructor :

* If a constructor is written by user without any parameter then it is called non parameterized constructor.

* No argument constructor and default constructor, both look the same the only difference is, default constructor means added by compiler and no argument constructor means written by user.

* No argument constructor is mainly used to initialise our non static field with default value.

```
public class Student
{
    private int rollNumber;
    private String studentName;

    public Student(){ //No Argument Constructor
        rollNumber = 0;
        studentName = null;
    }

}
```

3) Parameterized Constructor :

If we pass one or more parameter to the constructor then it is called parameterized constructor.

By using parameterized constructor all the objects will be initialized with different values.

Example :
```
public class Employee
{
    private int id;
    private String name;

    public Employee(int id, String name)
    {
        this.id = id;
        this.name = name;
    }
}
```

//Program :
```
package com.ravi.complete_oop;

public class Dog
{
    private String name;
    private int age;
    private double height;
    private String color;

    public Dog(String name, int age, double height, String color)
    {
        this.name = name;
        this.age = age;
        this.height = height;
        this.color = color;
    }


    @Override
    public String toString()
    {
        return "Dog [name=" + name + ", age=" + age + ", height=" + height + ", color=" + color + "]";
    }
}
```

```
package com.ravi.complete_oop;

public class DogDemo
{
    public static void main(String[] args)
    {
        Dog tiny = new Dog("tiny", 2, 2.5, "White");
        SO.println(tiny);
    }

}
```