What is a Constructor [Constructor Introduction] :

* If the **name of the class** and **name of the method** both are **exactly same and It should not contain any return type then It is called Constructor.**
Example :

```
public class Example                          public class Example
{                                             {
   public  Example()  //Constructor             public void Example()  //Method
   {                                            {
   }                                            }
}                                             }
```

* In Java, Whenever we write a class (either BLC OR ELC) then automatically a **default no argument constructor will be added by java compiler. If a user does not write any constructor in the class. [WE CANNOT THINK ABOUT JAVA CLASS WITHOUT CONSTRUCTOR]**

* We can say, Every Java class **must contain at-least one constructor** either explicitly written by developer OR implicitly added by java compiler.

* The **access modifier of default no argument constructor** added by java compiler will depend upon the class access modifier that means, If the class is public then compiler added constructor is also public, on the other hand, if the class is non-public then compiler added constructor is also non-public

Case 1 :

Test.java

```
public class Test
{
}
```

javac

Test.class

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

Case 2 :

Test.java

```
class Test
{
}
```

javac

Test.class

```
class Test
{
   Test() //default no argument constructor
   {
   }
}
```

Case 3 :

Test.java

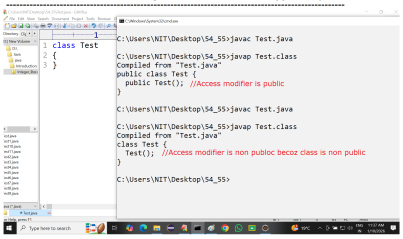
```
public class Test
{
   public Test()
   {}
}
```

javac

Test.class

```
public class Test
{
   public Test() //Developer already written a
   {                constructor so compiler
   }                will not add
}
```

Command to show that default no argument constructor is added by java compiler :



Why default no argument constructor is added by Java compiler:

Program + Diagram :

```
public class Student
{
   int roll;
   String name;

   public Student() //default no argument
   {
   }

   public void show()
   {
     IO.println(roll);
     IO.println(name);
   }
}
public class StudentELC
{
   void main()
   {
      Student raj = new Student();
      raj.show();
   }
}
```

Work of new keyword :

* It is used to allocate the memory for non static members (non static fields + non static methods) which is known as **"Instantiation process".**

* It is responsible to provide **default value for all the non static fields during object creation, For data type information, It will take the help of java compiler and accordingly It will allocate the memory and provide default values which is known as "Initialization Process".**

new keyword = Instantiation + Initialization

default value provided by new keyword :

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

* The default no argument constructor is added by java compiler so, Object creation is possible in java, without writing user-defined constructor.

* Whenever we create an object by **using new keyword**, at-least one constructor must be invoked either the constructor added by java compiler OR the constructor written by user.

WITHOUT CONSTRUCTOR SUPPORT WE CANNOT CREATE AN OBJECT USING NEW KEYWORD

*** reference variable is required to access OR to refer the members (non static field + non static method) of the object**


```
raj.roll = 101;
raj.name = "Raj";
raj.show();
```

WAP to show that the default values are provided by new keyword with the support of compiler :

```
package com.nit.elc;

public class StudentELC
{
    void main()
    {
       new Student();  //Anonymous OR Nameless Object
    }
}

class Student
{
   int roll;
   String name;

   public Student() //user written constructor
   {
       IO.println(roll);
       IO.println(name);
   }
}
```