Constructor: A constructor initializes an object when it is created. It has the same name as its class and is syntactically similar to a method. However, constructors have no explicit return type. Typically, you will use a constructor to give initial values to the instance variables defined by the class, or to perform any other start-up procedures required to create a fully formed object. All classes have constructors, whether you define one or not, because Java automatically provides a default constructor that initializes all member variables to zero. However, once you define your own constructor, the default constructor is no longer used.

Purpose: The purpose of constructor is to initialize the object of a class while the purpose of a method is to perform a task by executing java code. Constructors cannot be abstract, final, static and synchronized while methods can be. Constructors do not have return types while methods do.

Special Method: A constructor is a special method of a class that initializes new objects or instances of the class. Without a constructor, you can't create instances of the class. Imagine that you could create a class that represents files, but without constructors, you couldn't create any files based on the class

Syntax:

```
Class ClassName
{
ClassName()
 {block-statement;
   }
}
```

For Example:

```
class hello
{
hello ()
    {
System.out.println("Hello World");

    }
public static void main(String[] args)
    {
hello t1 = new hello();

    }
}
```

Types of Constructor

Default Constructor: A default constructor is a constructor created by the compiler if we do not define any constructor(s) for a class.

For Example:

```
class A
{
int a;
String b;
boolean c;
    }
class b {
public static void main(String[] args)
    {
        A a1=new A();
System.out.println("value of a="+a1.a);
System.out.println("Text Message is="+a1.b);
System.out.println("Value of c="+a1.c);
    }
}
```

No argument constructor: Similar to methods, a Java constructor may or may not have any parameters (arguments). If a constructor does not accept any parameters, it is known as a no-argument constructor.

Syntax:

```
private Constructor() {
   // body of the constructor
}
```

For Example:

```
class A
{
A()
{
System.out.println("this is no argument constructor");
}
    }
class b {
public static void main(String[] args)
```

```
    {
        A a1=new A();
    }
}
```

Parameterized Constructor: Most often, you will need a constructor that accepts one or more parameters. Parameters are added to a constructor in the same way that they are added to a method, just declare them inside the parentheses after the constructor's name.

For Example:

```
public class P
{
int a;
    String b;
boolean c;
P(int x, String y, boolean z)
    {
        a=x;
        b=y;
        c=z;
    }
}
class q {
public static void main(String args[])
    {
        P p1=new P(123,"CCIT", true);
System.out.println("value of a="+p1.a);
System.out.println("Text Message is="+p1.b);
System.out.println("Value of c="+p1.c);
}
}
```

```
For Example:
class filefirst
{
    filefirst(int a,String b, int h)
    {
        System.out.println("Id "+a+" Name "+b+" And Marks "+h);

    }
```

```
    public static void main(String[] args) {

        filefirst s=new filefirst(10,"ccit",1993);
}
    }
```

Copy Constructor: A Copy Constructor in Java is a special type of constructor that is used to create a new object using the existing object of a class that we have created previously. It creates a new object by initializing the object with the instance of the same class.

For Example:

```
class L
{
int t;
    String name;
L(int a, String b)
    {
        t=a;
name=b;
    }
L(L o1)
    {
        t=o1.t;
name=o1.name;
    }
void display()
    {
System.out.println("Value of t="+t);
System.out.println("Value of text="+name);
    }
public static void main(String[] args)
{
    L o2 = new L(1994,"CCIT");
o2.display();
System.out.println();
System.out.println("After copy values then print");
System.out.println();
    L o3=new L(o2);
o3.display();
}
```

```
}
```

For Example: In this example use the constructor and using multiple constructor but having different argument.

```
class multiple
{
    multiple()
    {
        System.out.println("Without argument Constructor");
    }
    multiple(String name)
    {
        System.out.println("With argument Constructor");
        System.out.println("Name is "+name);
    }
    multiple(int age)
    {
        System.out.println("With argument Constructor But different");
        System.out.println("Age is "+age);
    }
    public static void main(String[] args) {
        multiple mul1=new multiple();
        multiple mul2=new multiple("CCIT");
        multiple mul3=new multiple(1993);
    }
}
```