Constructor

- It is a special kind of method of a class whose name is same as that of class name.
- It is used to initialize the object ie to initialize the attributes of the class.
- It does not have any return type but implicitly it returns the instance of the current class.
- Constructors are invoked implicitly when the objects are created.
- Constructors are used to create a new object. They are used to initialize the object.

Syntax

```
classname()
{
//body of constructor
}
```

Rules for Constructors

- The name of the constructor must be the same as the name of its class.
- A constructor must have no return type. It can not have not even void as its return type.
- We can use the access modifiers with a constructor to control its access so that other classes can call the constructor.
- We can not declare a constructor as final, abstract, abstract and synchronized.

Types of Constructor

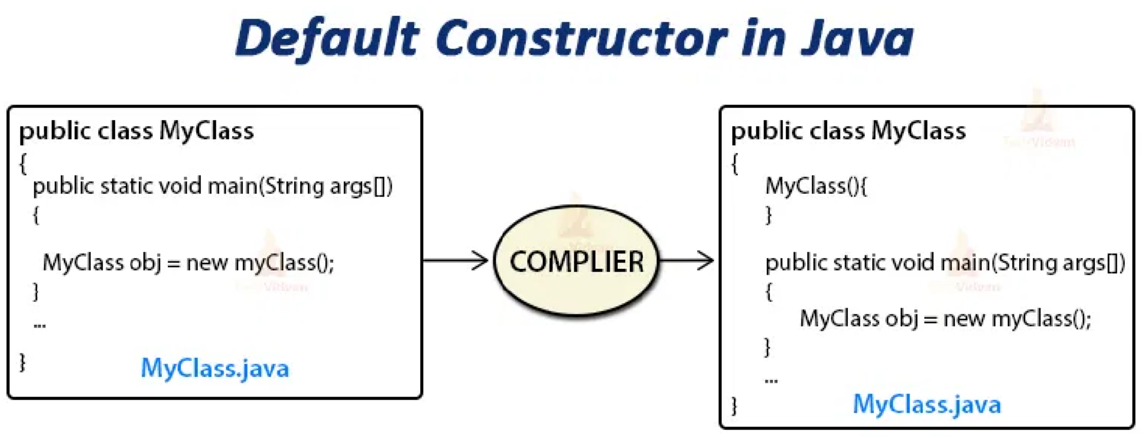
- **Default Constructor/no argument constructor**
- **Parameterized Constructor**

Default Constructor

A Default Constructor is a constructor with no parameter. The Java compiler automatically creates a default constructor if we do not write any constructor in our program.

The compiler initializes data fields to its default values such as:

- numeric data types set to 0
- char data types set to a null character ('\0')
- reference variables set to null



```
public class MyClass
{
  public static void main(String args[])
  {
    MyClass obj = new myClass();
  }
  ...
}
```

```
public class MyClass
{
    MyClass(){
    }
    public static void main(String args[])
    {
        MyClass obj = new myClass();
    }
    ...
}
```

```
class Student
{
 int roll_no;
 String name;
 static String college="cime";
}
class constructor
{
 public static void main(String[] s)
 {
  Student s1=new Student();
  System.out.println("Roll is"+s1.roll_no);
  System.out.println("Name  is"+s1.name);
Student s2=new Student();
Student s3=new Student();



 }
}
```

o/p

```
Roll is0
Name  isnull
```

Parameterized Constructor

A Parameterized constructor is a constructor with a specific number of parameters. We can use parameterized constructor mainly to initialize the members of the class with different values or objects.

```
class Student
{
 int roll_no;
 String name;
Student(int r,String n)
{
roll_no=r;
name=n;
}
}
class constructor
{
 public static void main(String[] s)
 {
  Student s1=new Student(1,"sai");
  System.out.println("Roll is"+s1.roll_no);
  System.out.println("Name  is"+s1.name);
 }
}
```

Parameterized Constructor where reference of same class is passed as parameter(similar to copy constructor)

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

```
Student(int r,String n)
{
roll_no=r;
name=n;
}
Student(Student o)
{
roll_no=o.roll_no;
name=o.name;
}
}
class constructor
{
 public static void main(String[] s)
 {
  Student s1=new Student(1,"sai");
  System.out.println("Roll is"+s1.roll_no);
  System.out.println("Name  is"+s1.name);

  Student s2=new Student(s1);
System.out.println("Roll is"+s2.roll_no);
  System.out.println("Name  is"+s2.name);
 }
}
```

Constructor Overloading

- One class having more than one constructor with different signature used for different purpose is known as constructor overloading

In the above example student class constructors are overloaded.

```
Student(int r,String n)
{
roll_no=r;
name=n;
}
Student(Student o)
{
roll_no=o.roll_no;
name=o.name;
}
```

this reference variable

- **this is a reference variable that holds the address of currently invoking object.**
- **It always refers to the current object.**

When this is used

- **If the instance variable names of the class and the parameter names passed in the parameterized constructors are same then this is used to access the instance variable.**

this.attributename=parametername;

- **this() is used to call one constructor within another constructor.**

this(paremeter);

Note: this statement should be the first statement in the method.

Exa

```
student()
{
roll=0; //compilation error ; this() should be first line
 this(1,"sai");
}
```

How to destroy an object created by constructor

- **Garbage collector** is a program running inside jvm destroy the object which is not used .
- **Garbage collector** is executed when program is closed.or when any object goes out of reach.
- mark()
- sweep()
- System.gc() is the method for **Garbage collector .**
- Before gc() is closed one special method is called automatically to do some closing operation like file close, connection close etc.

```
protected void finalize()
{
//close methods are called
```

}

- finalize() is an inbuilt method present in **Object class.**
- **Object class** is known as cosmic super class ie it is super class of every class.