Constructor

A constructor is a *special* non static method which when called, will create an instance.

So basically, the memory created by the new Keyword is filled up with a reference that is generated by a constructor.

new -> An empty memory block in the heap

Constructor -> Fills up the created memory with the reference of a class

```
public ClassName(args)
{
  …
}
```

- There is no return type.
- The name of a constructor is same as className.

Types of Constructors

1. Default Constructor

Whenever a programmer fails write a constructor in the program, the compiler/JVM adds a constructor in the program.

The default constructor is added in the program at the time compiling the program so any programmer cannot view it.

```
class A1
{
   //a default constructor is added by the JVM
   public static void main(String[] args)
   {
      A1 obj = new A1();
      System.out.println( obj );//instance
                              //A1@address
   }
}
```

2. Custom Constructor

These types of constructor are added by the programmer, so the programmer can write anything in the constructor block.

a) No argument:

As the name itself specifies the constructor can have statement written but it cannot have any arguments.

Syntax:

public ClassName() //O arguments
{
.....
}

Ex:

```
public class Emp
{
   String E_id;
   static int E_count=0, year=2020;
   static String Cname="Qsp";

   public Emp()
   {
      /*no argument constructor can be used,
      to auto initialize the properties of an Object*/
      E_id = Cname+ year+ ++E_count;
   }
   public void work()
   {
      System.out.println("Employee with id: "+E_id+" is working");
   }
   public static void main(String[] args)
   {
      Emp e1=new Emp();
      Emp e2=new Emp();
      e1.work();
      e2.work();
   }
}

class Car
{
   int TopSpeed, SerialNo;

   static int count=0;

   public Car()
   {
      SerialNo = ++count;
      //We cannot cross a limit of 100KMPH
      TopSpeed=100;
   }

   public void drive()
   {
      System.out.println("Car with "+SerialNo+
                  " is being driven @ "+TopSpeed);
   }

   public static void main(String[] args)
   {
      Car a =new Car();
      Car b =new Car();
      Car c =new Car();
```

```
        a.drive();
        b.drive();
        c.drive();
    }

}
```

Note: It is recommended that in every object oriented program we use a no argument constructor.

this keyword:

"this" keyword is a non static keyword, is used to call the global members of a current class.

Basically this keyword, is pointing to the class.

"this" keyword can be used only in non-static blocks/methods.

```
class A2
{
    String name;
    public A2(){}
    public void initializeName(String name)
    {
        this.name=name;
        // global=local
        //here we are pointing to the current
                        object variable
    }

    public static void main(String[] args)
    {
        A2 a=new A2();

        a.initializeName("Qspiders");
        System.out.println(a.name);
    }
}
```

b) Parametrized

Parameters basically means arguments and parametrized constructors means, here our constrcutor can have arguments.

We can use these arguments to initialize the properties of an object.

Syntax:

```
public ClassName( arguments )
{
        this.properties = arguments;
}
```

```
class Student
{
   static int count=0;

   //Properties of student object
   int roll_no =100;
   String name;
   char section;


   //Constructor is used to initialize properties

   public Student()//no argument
   {
      //autogenerating properties
      this.roll_no = roll_no+ ++count;
   }

   public Student(String name, char section)
   {
      //storing user-defined properties
      this.name=name;
      this.section=section;
   }

   public void printDetails()
   {
      System.out.println("Student details are:");
      System.out.println("Name    : "+name);
      System.out.println("Section : "+section);
   }

   public static void main(String[] args)
   {
      System.out.println("----------------------");

      Student s1;
      //user defined values for name and section
      (s1 = new Student("Abcde",'A')).printDetails();
      //autogenerated values for roll_no
      System.out.println("Roll no : "+(s1 = new Student()).roll_no);

      System.out.println("----------------------");

      Student s2;
      //user defined values for name and section
      (s2 = new Student("Defg",'C')).printDetails();
      //autogenerated values for roll_no
      System.out.println("Roll no : "+(s1 = new Student()).roll_no);

      System.out.println("----------------------");
   }
}
```

If we follow the basic coding conventions

OR

***If we are writing program according to industrial standards (use a feature for its actual purpose), then every object oriented program is going to perform* Constructor Overloading.**

Constructor overloading means writing multiple constructors with different arguments.

In the program given below, the object is not instantiated properly.

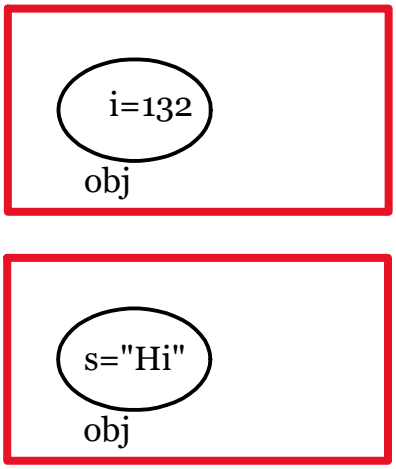
```
public class S
{
   int i;
   String s;

   public S()
   {
```

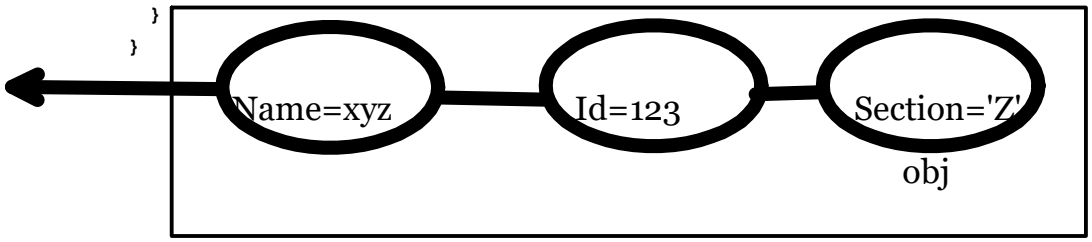
```
    this.i = (int)(Math.random() *2);
}
public S(String s)
{
    this.s=s;
}
void print()
{
    System.out.println(i+" "+s);
}
public static void main(String[] args)
{
    S obj;

    obj=new S();//here we only get i
    obj.print();

    obj=new S("hi");//her, we only get s
    obj.print();
```



The solution for this is that no matter how many instances we create, they all should be connected with each other.



Constructor Chaining

Chaining is a concept of a constructor calling another constructor, so that N number of instance are created for only one object.

Constructor chaining is a possible with the help of ***this call statement.***

this call statement *is used to call constructors of current class.*

```
class Car
```

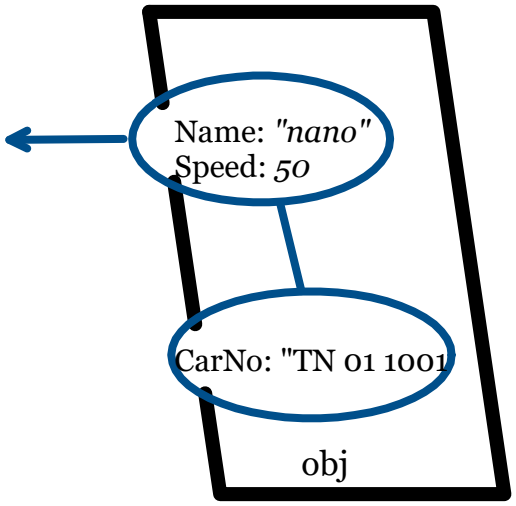
```
{
   static int CarCount=1000;

   String CarNo="TN 01 ";
   int speed;
   String name;

   public Car(String name, int speed)
   {
      this();//this call statement
      this.name=name;
      this.speed=speed;
   }

   public Car()
   {
      this.CarNo = CarNo+ ++CarCount;
   }

   public void details()
   {
      System.out.println("Car details are");
      System.out.println("Name     : "+name);
      System.out.println("Number   : "+CarNo);
      System.out.println("TopSpeed : "+speed);
   }
   public static void main(String[] args)
   {
      Car nano = new Car("Nano",50);
      nano.details();
   }
}
```




```
public class Bank
{
   static
   {
      System.out.println("Welcome to CitiBank");
   }

   static int Ccount=300, year=2020;
   static double minimumBalance=5000;

   int acc_no;
   String name;
   double balance;

   public Bank()
   {
      this.acc_no = year*100 + ++Ccount;
   }

   public Bank(String name, double balance)
   {
      this();
      this.name=name;
      this.balance = minimumBalance + balance;
   }


   {
      System.out.println("Account is created");
   }

   public void viewDetails()
   {
      System.out.println("--------------------");
      System.out.println("Name       : "+name);
      System.out.println("Account no : "+acc_no);
      System.out.println("Balance    : "+balance);
      System.out.println("--------------------");
   }
```

```
    public static void main(String[] args)
    {
        Bank C1 = new Bank("Vijay Mallya",50000);
        Bank C2 = new Bank("Ambani",500000);

        C1.viewDetails();
        C2.viewDetails();
    }

}
```