Constructor and Inheritance

Constructor

- It’s a special type of method whose name is same as class name.
- Purpose is to initialize the object.
- Every Java Class has constructor.
- Constructor is automatically called at time of object creation.
- A constructor does not contain return type including void.

```
Class class-name
{
        class-name ()                   //constructor starts
        {

        }                               //constructor ends
}
```

```
class A
{
    int a; String name;
   /*  A() //constructor comment constructor see the output of default constructor
    {
        a=1; name="DJS";
    }*/
    void show()
    {
        System.out.print("a="+a+" "+"Name="+name);
    }
}
class constuct
{
    public  static void main (String[] args)
    {
        A ref =new A();
        ref.show();
    }
}
```

Types of constructor

1. Private
2. Default
3. Parametrized
4. Copy

1. Default Constructor

- A constructor which does not have any parameter
- If user does not specify explicit constructor Java adds default constructor and initialize the value

2. Parametrized constructor

- A constructor through which we can pass one or More parameter is called parameterized constructor

```
class A
{
    int a; String b;
    A(int x, String y)
    {
        a=x; b=y;
    }
    void show()
    {
        System.out.print("a="+a+" "+"b="+b);
    }
}
class parametconstuct
{
    public  static void main (String[] args)
    {
        A ref = new A(1000,“DJS");
        ref.show();
    }
}
```

Copy Constructor

- Copies contains of another constructor
- Contents of one object is copied into another Object

```
class A
{
    int a;
    String b;
    A()
    {
        a=10; b="D J Sanghvi";
        System.out.println("First Object"+a+" "+b);
    }
    A(A ref)
    {
        a=ref.a;
        b=ref.b;
        System.out.println("Second Object"+a+" "+b);
    }
}
public class copyconst
{
    public static void main(String[] args)
    {
        A r=new A();
        A r2=new A(r);

    }
}
```

This Keyword

- This keyword refers to the current object inside a method or constructor
- It uses unique reference ID to refer the current object

```
class thiskeyword
{
   void show()
   {
      System.out.println(this);
   }
   public static void main(String[] args)
   {
      thiskeyword r=new thiskeyword();
      System.out.println(r);
      r.show();
      thiskeyword r1=new thiskeyword();
      System.out.println(r1);
      r1.show();
   }
}
```

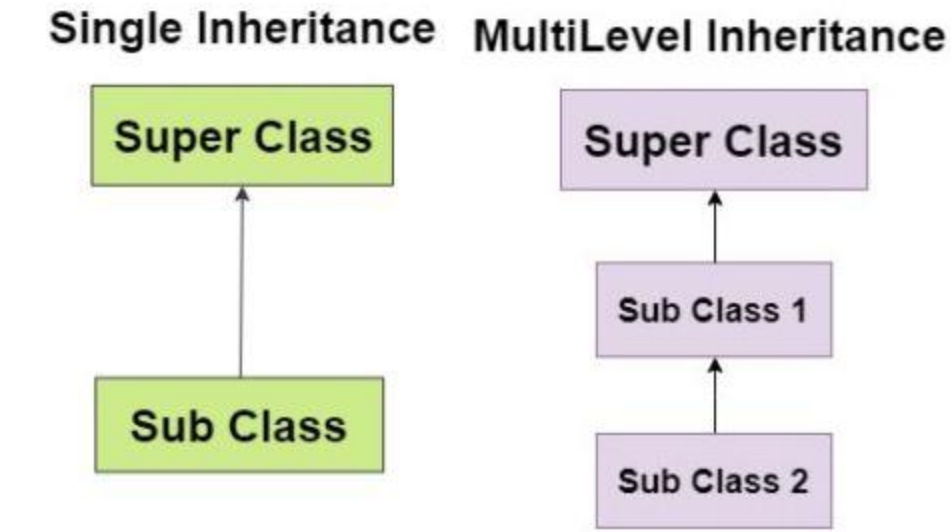
Inheritance

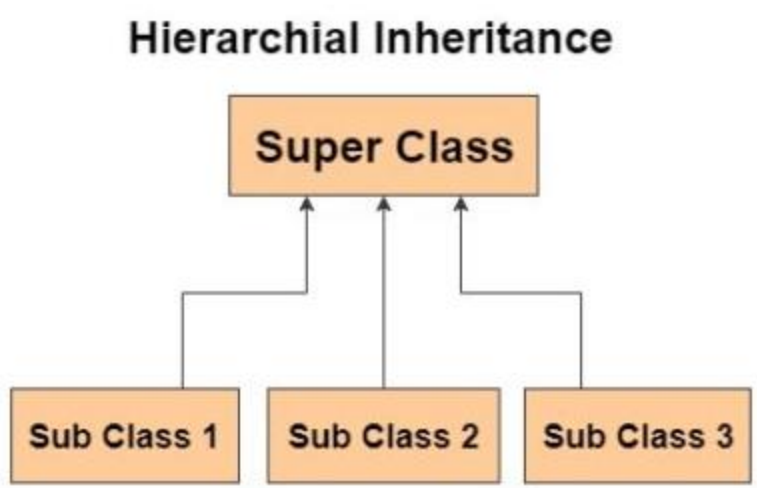
- When construct a new class from existing class in such a way that the new class access all the features and properties of existing class called inheritance.

1. In Java *extends* keyword is used to perform inheritance.
2. It provide code reusability.
3. We can’t access private members of class through inheritance.
4. A subclass (Derived from superclass) contains all the features of super class so, we should create the object of sub class.
5. Method overriding only possible through Inheritance.

Syntax:

```
Class A     //super class
{
            -----
            -----
            ------
}
Class B extends A     //sub class
{
            -----
            -----
            ------
}
```


Single Inheritance
Super Class
Sub Class
MultiLevel Inheritance
Super Class
Sub Class 1
Sub Class 2

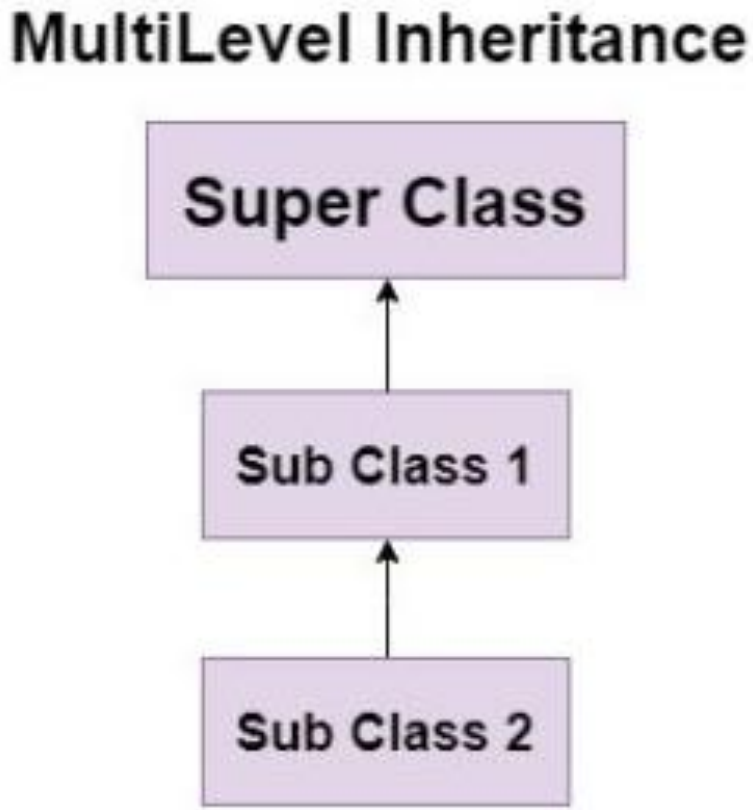

Hierarchial Inheritance
Super Class
Sub Class 1
Sub Class 2
Sub Class 3

```
        /*Simple Inheritance*/
class student                       //super class
{
        int roll, marks;
        String name;
        void input()
        {
                System.out.println("Roll No. and Marks:");
        }
}
class simpleinherit extends student     //sub class
{
        void disp()
        {
                roll=1; name="ankit"; marks=89;
                System.out.println(roll+" "+name+" "+" "+marks);
        }
        public static void main(String[] args){

                simpleinherit r=new simpleinherit();
                r.input();
                r.disp();
        }
}
```

Multi level Inheritance

- Multi level inheritance has One super class and Multiple sub class



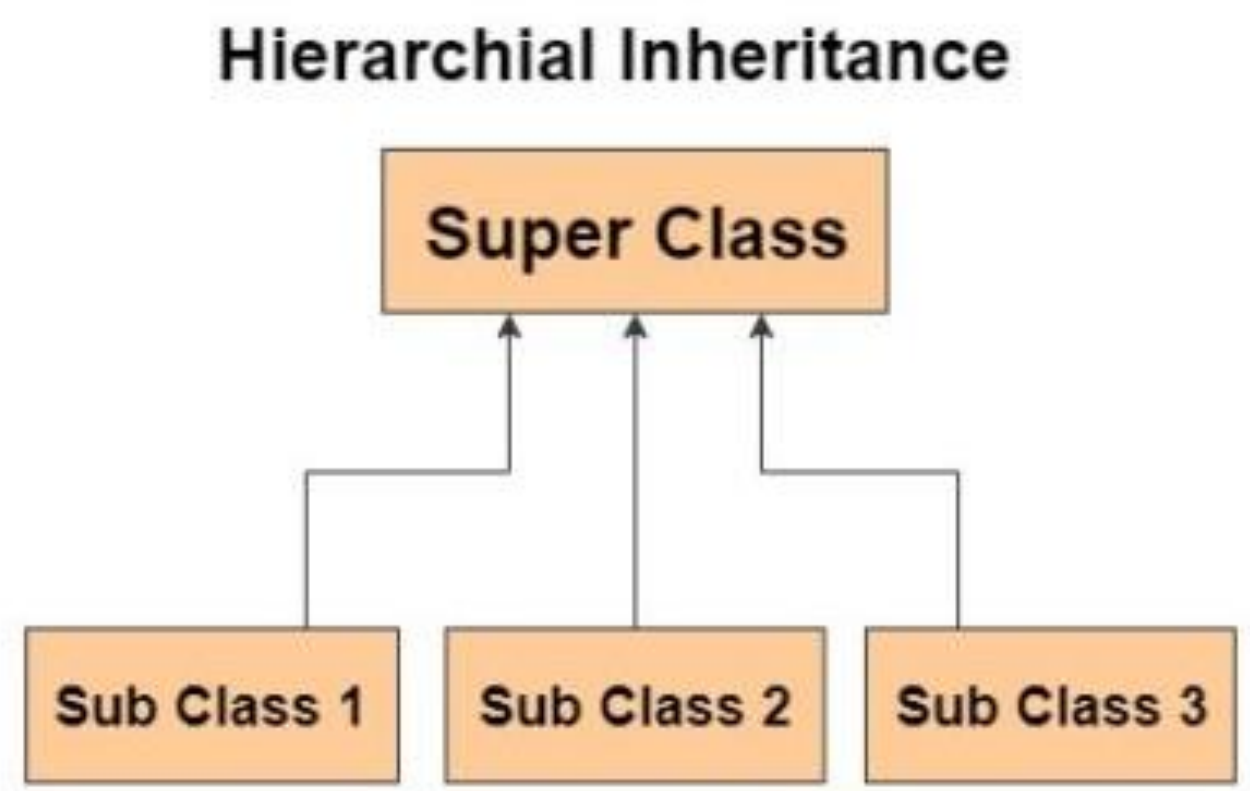
untitled

```
/*  Multi-level Inheritance  */
class A// Super
{
    int a,b,c;
    void add()
    {
        a=10; b=20;
        c=a+b;
        System.out.println("Sum of two Numbers: "+c);
    }
    void sub()
    {
        a=200; b=100;
        c=a-b;
        System.out.println("Sub of two Numbers: "+c);
    }
}
class B extends A//sub1
{
    void multi()
    {
        a=10; b=20;
        c=a*b;
        System.out.println("Multiplication of two Numbers: "+c);
    }
    void div()
    {
        a=10; b=2;
        c=a/b;
        System.out.println("Division of two Numbers: "+c);
    }
}
class C extends B//sub2
{
    void rem()
    {
        a=10; b=3;
        c=a%b;
        System.out.println("Remainder of two Numbers: "+c);
    }
}
```

```
class Test
{
    public static void main(String[] args) {
        C r=new C();
        r.add();  r.sub();  r.multi();  r.div();  r.rem();
    }
}
```

Hierarchical inheritance

- A inheritance which contains only one super class and multiple sub class and all sub class directly extends super class called hierarchical inheritance.



/* Hierarchical Inheritance */

```
class A         //super class
{
                void input()
                {
                                System.out.println("My Name is Ganesh ");
                }
}
class B extends A           // sub class 1
{
                void show()
                {
                                System.out.println("My Name is Ravi ");
                }
}
class C extends A           //sub class 2
{
                void disp()
                {
                                System.out.println("My Name is Mahesh");

                }
}
class Hierainh
{
                public static void main(String[] args)
                {
                                B r= new B();
                                C r2= new C();
                                r.input(); r.show();
                                r2.input(); r2.disp();
                }
}
```

Super keyword

- Super Keyword refers to object of the super class, it is used when we want to call the super class variable, method and constructor through sub class object.

1. It is used only when the super class and sub class variable or method has same name
2. To avoid the confusion between super class and sub class variable and methods that have same name then super keyword can be used.

```
class A
{
		int a=20;
		void show()
		{
			System.out.println("SVKM");
		}
}
class B extends A
{
		int a=10;
		void show()
		{
			System.out.println("value of a:"+a);
			System.out.println("value of a from Parent class:"+super.a);
			System.out.println("DJ Sanghvi");
			super.show();
		}
}
```

```
class superkeyword
{
	public static void main(String[] args){
		B r=new B();
		r.show();
	}
}
```