IS-A Relationship [inheritance]

- It is the process of the creating the sub class by extending properties of super class
- In this process we have to make use “extends” keyword

Syn:

```
[modifiers]  <class>   <SuperClassName>
{
  Fields
    Methods
}
[modifiers] <class>  <SubClassName> extends
                                    <SuperClassName>
{
 Fields
  Methods
}
```

- **Private members are not inherited**
- **Inherited members can be used as regular member of the class**
- **Static members are inherited**
- **We can declare sub class instance field same as instance field of the super class**
- **We can also define sub class method same as super class method**

Example :

```
class SuperA
{
        private int x=111;  //private member
                int y=222;  //default variable
        static int z=333;  //static variables
}

class SubB extends SuperA
{
}

class IS_AEx1
{
        public static void main(String args[ ])
        {
                SubB sb=new SubB();
                // System.out.println("x val is : "+sb.x); CE
                System.out.println("y val is : "+sb.y);
                     System.out.println("z val is : "+SubB.z);
                     System.out.println("z val is : "+SuperA.z);
                     System.out.println("z val is : "+sb.z);

      SuperA sa=new SuperA();
  //  System.out.println("x val is : "+sa.x); CE
        }
}
```

Note :

- **Private members can't be accessed from outside of the class**
- **Creating an object of the sub class is doesn't mean creating an object of its super class**
- **For all the members of Super class and all the member of sub class memory will be allocated with in the object of sub class**

Note:

- **If both local and instance field are declared with the same name the priority is given is local variable only, if you want access the instance field then we have use "this" keyword.**

- **If both instance field of the sub and super class are declared with the same name the priority is given to instance field of the "subclass". If you want access instance field of the "superclass" then we have to make use of "super" keyword**

```
class SuperA
{
        int x=111;  //instance of SuperA
}
class SubB extends SuperA
{
        int x=222;  //instance of sub

        void method1()
        {   int x=333;
                System.out.println("M1 x : "+x);
                System.out.println("M1 Instance : "+this.x);
                System.out.println("Instance of SA x : "+super.x);
    }
```

```
}
class IS2
{
        public static void main(String args[ ])
        {
                SubB sb=new SubB();
                        sb.method1();
        }
}
```

Example :

```
class A
{
        int x=10;
}
class B extends A
{
        int x=20;
}

class C extends B
{
        int x=30;
        void method1()
        { System.out.println(super.super.x); }

        public static void main(String args[])
        { C c=new C( );
             c.method1();
    }
} //Error
```

Note:

- **this.this | super.this | this.super | super.super are invalid**

We can also define sub class method same as super class method

```
class SuperA
{
	void method1()
	{ System.out.println("M1 of SA "); }
}
class SubB extends SuperA
{
	void method1()
	{
		super.method1();
		System.out.println("M1 of SB ");
	}
}
class IS4
{
	public static void main(String args[ ])
	{
		SubB sb=new SubB();
			sb.method1();
	}
}
```

Note :

- **the keywords "this" and "super" should not be used the static context for Eg: ["main"]**

Rules of the Constructors in the Inheritance

- **the default constructor of the super class will be invoked by the default constructor of the super implicitly**

- **If the Super Class is defined with both default and parameterized constructor ,In this is case also the default constructor of super class will be invoked by the default constructor of the Subclass implicitly . If you want invoke the parameterized constructor of the super class by the default constructor of subclass then we have make to use of "super()"**

- **If Super class and Sub class both are defined with the parameterized constructors ,then we must call the parameterized constructor of the super class by the parameterized constructor of subclass explicitly. Other wise parameterized constructor of the sub class is trying to call default constructor of the super.**

Example :

```
class SuperA
{
        SuperA()
        { System.out.println("Def const of SA "); }
}
```

```
class SubB extends SuperA
{
        SubB()
        { System.out.println("Def const of SB "); }
}
class R1
{
        public static void main(String args[ ])
        {  SubB sb=new SubB(); }
}
```

Example 2:

```
class SuperA
{
        SuperA()
        { System.out.println("Def const of SA "); }

        SuperA(int x)
        {  this();
                System.out.println("Para const of SA : "+x); }
}
class SubB extends SuperA
{
        SubB()
        {
                super(123);
                System.out.println("Def const of SB ");
        }
}
class R2
{
```

```
        public static void main(String args[ ])
        {  SubB sb=new SubB(); }
}
```

Note:

- **this( ) vs super( )**
  - **if you want call an existed constructor inside of another constructor of the same class then we have to make user of "this( )"**
  - **If you want call an existed constructor of the super class into another constructor of the sub class then we have to make use of "super( )"**

Example 3:

```
class SuperA
{
        SuperA(int x)
        { System.out.println("Para const of SA "+x); }
}
class SubB extends SuperA
{
        SubB(int x)
        {  super(145);  //try by removing  this stmt
             System.out.println("Para const of SB : "+x); }
}
class R3
{
        public static void main(String args[ ])
```

```
    {
        SubB sb=new SubB(123);
    }
}
```

Note :

- If you call either default or parameterized constructor of the subclass then internally JVM will call "Default constructor " of the Super class only.