3) How to call super class constructor :

Whenever we write a class in java and we don't write any kind of constructor to the class then the java compiler will automatically add one default no argument constructor to the class.

THE FIRST LINE OF ANY CONSTRUCTOR IS RESERVERD EITHER FOR super() or this() [super of OR this of] that means first line of any constructor is used to call another constructor of either same class (this()) OR super class(super()). From Java 25V we can only perform data validation before super() OR this()

In the first line of any constructor if we don't specify either super() or this() then the java compiler will automatically add super() to the first line of constructor.

super keyword always refers to immediate super class :

In order to call super class constructor and current class constructor, We have total 4 cases :

Case 1:

super() : Automatically added by Java compiler, It is used to call super class default OR No argument Constructor.

```java
package com.ravi.inheritance_intro;

class Super
{
    public Super()
    {
        IO.println("No Argument Constructor of Super class");
    }
}
class Sub extends Super
{
    public Sub()
    {
        IO.println("No Argument Constructor of Sub class");
    }
}

public class ConstructorChainingDemo1
{
   void main()
   {
      new Sub();  //Nameless OR Anonymous object
   }
}
```

The following program explains a default no argument constructor and super() will be automatically added by java compiler even the class is empty.

```java
class Alpha
{
  public Alpha()
  {
      IO.println("Alpha class");
  }
}
class Beta extends Alpha
{
}
class Gamma extends Beta
{
  public Gamma()
  {
    IO.println("Gamma class");
  }
}
public class ConstructorChaining
{
    void main()
    {
        new Gamma();
    }
}
```

Case 2 :

super(lionName) : Must be written by developer, It is used to call parameterized constructor of super class where super class parameterized constructor will accept one parameter of type String.

```java
package com.ravi.inheritance_intro;

class Animal
{
    protected String name;

    public Animal(String name)
    {
        super();
        this.name = name;
    }
}
class Lion extends Animal
{

      public Lion(String lionName)
      {
          super(lionName); //Calling Parameterized constructor of super class
      }

      public String getName()
      {
          return super.name;
      }
}

public class ConstructorChainingDemo2
{
   void main()
   {
      Lion lion = new Lion("Simba");
      IO.println(lion.getName());
   }
}
```

Case 3 :

this() : Must be explicitly written by user. It is used to call no argument constructor of current class.

```java
package com.ravi.inheritance_intro;

class Alpha
{
    public Alpha()
    {
        IO.println("No Argument Constructor of Alpha class");
    }

    public Alpha(int x)
    {
        this();
        IO.println("Parameterized Constructor of Alpha class :"+x);
    }
}
class Beta extends Alpha
{
    public Beta()
    {
        super(9);
        IO.println("No Argument constructor of Beta class");
    }
}

public class ConstructorChaningDemo3
{
   void main()
   {
       new Beta();
   }
}
```

Case 4 :

this(15);

this("java") : Must be explicitly written by user. It is used to call parameterized constructor of current class.

```java
package com.ravi.inheritance_intro;

class Alpha
{
    public Alpha()
    {
        this(15);
        IO.println("No Argument Constructor of Alpha class");
    }

    public Alpha(int x)
    {
        super();
        IO.println("Parameterized Constructor of Alpha class :"+x);
    }
}
class Beta extends Alpha
{
    public Beta()
    {
        IO.println("No Argument constructor of Beta class");
    }
}

public class ConstructorChaningDemo3
{
   void main()
   {
       new Beta();
   }
}
```

//Program on Hierarchical Inheritance :

```java
package com.ravi.hierarchical;

class Shape
{
    protected int x;

    public Shape(int x)
    {
        super();
        this.x = x;
        IO.println("x value is :"+x);
    }
}
class Square extends Shape
{

    public Square(int side) //Flexible Constructor body [Java 25v]
    {
        if(side <=0)
        {
            System.err.println("Error : Square side cannot be negative");
            System.exit(0);
        }
        super(side);
    }

    public double getAreaOfSquare()
    {
        double area = x * x;
        return area;
    }
}
class Rectangle extends Shape
{
    protected int breadth;
    public Rectangle(int length, int breadth)
    {
        super(length);
        this.breadth = breadth;
    }

    public double getAreaOfRectangle()
    {
        double area = x * this.breadth;
        return area;
    }
}

public class HierarchicalDemo {

    public static void main(String[] args)
    {
        Square ss = new Square(5);
        IO.println("Area of square is :"+ss.getAreaOfSquare());

        Rectangle rr = new Rectangle(9, 4);
        IO.println("Area of Rectangle is :"+rr.getAreaOfRectangle());

    }

}
```