Different cases to call the contsructor of same class as well as super class :

* In order to call the constructor of same class as well as super class, We have four different cases :

Case 1 :

super() :

* Automatically added by java compiler to call the super class no argument OR default constructor.

```
 class Alpha
 {
     public Alpha()
     {
         System.out.println("Alpha Class Constructor");
     }
 }
 class Beta extends Alpha
 {
     public Beta()
     {
         System.out.println("Beta Class Constructor");
     }
 }
 public class ConstructorChainingDemo1
 {
     public static void main(String[] args)
     {
         new Beta();
     }
 }
```

```
class Alpha
{
    public Alpha()
    {
        System.out.println("Alpha Class Constructor");
    }
}
class Beta extends Alpha
{
}
class Gamma extends Beta
{
    public Gamma()
    {
        System.out.println("Gamma Class Constructor");
    }
}

public class ConstructorChainingDemo1
{
    public static void main(String[] args)
    {
        new Gamma();
    }
}
```

Note : From the above program, It is clear that even an empty class contains default constructor and super() to the first line.

Case 2 :

super("NIT") : Must be written by user explicitly to the first line of constructor. It is used to call parameterized constructor of super class which will accept one parameter of type String

```
 package com.ravi.inheritane;

 class Super
 {
    public Super(String name)
    {
       System.out.println("Institute name is :"+name);
    }
 }
 class Sub extends Super
 {
    Sub(String name)
    {
        super(name);
        System.out.println("Parameterized constructor of sub class");
    }
 }
 public class ConstructorChainingDemo2 {

    public static void main(String[] args)
    {
        new Sub("NIT");
    }
 }
```

Note : IN ORDER TO START THE **EXECUTION FLOW** OF NON STATIC MEMBER (NON STATIC FILED + NON STATIC METHOD) THE CONTROL MUST REACH TO **OBJECT CLASS FIRST**

```
 class Alpha
 {
     public Alpha(String name)
     {
         System.out.println("My name is :"+name);
     }
 }
 class Beta extends Alpha
 {
     public Beta()
     {
         super(getName());
     }

     public  String getName()
     {
         return "Ravi";
     }
 }

 public class ConstructorChainingDemo1
 {
     public static void main(String[] args)
     {
         new Beta();
     }
 }
```

Above program will generate Compilation error.

Case 3 :

this() : Expilicitly written by user to the first line of constructor. It is used to call current class no argument constructor

```
package com.ravi.inheritane;

class A
{
   A()
   {
      super();
     System.out.println("No Argument constructor of A class");
   }

   A(int x)
   {
      this();
      System.out.println("Parameterized constructor of A class :"+x);
   }
}
class B extends A
{
   public B()
   {
      super(100);
      System.out.println("No Argument constructor of B class");
   }
}
public class ConstructorChaningDemo3
{
   public static void main(String[] args)
   {
      new B();
   }
}
```

Assignment :

Case 4 :

this(29) : Explicitly written by user, Will call current class parameterized constructor which will accept one parameter of type int.

//Program on super() by using Hierarchical Inheritance :

```
 package com.ravi.hierarchical;

 class Shape
 {
    protected int x;

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

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

 public class HierarchicalDemo {

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

       Rectangle rr = new Rectangle(12, 10);
       System.out.println(rr.getAreaOfRectangle());
    }
 }
```