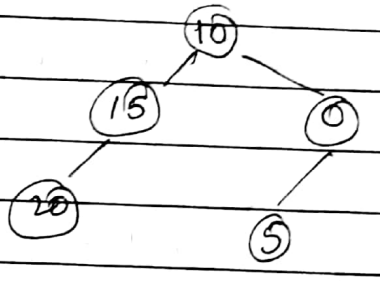
```
class MyComparator implements Comparator
{
    public int compare(Object obj1, Object obj2)
    {
        Integer I1 = Integer(obj1);
        Integer I2 = Integer(obj2);
        if (I1 < I2)
            return +1;
        else if (I1 > I2)
            return -1;
        else
            return 0;
    }
}
```

```
t.add(10);
t.add(0);  ⇒ +ve compare(0,10)
t.add(15); ⇒ -ve compare(15,10)
t.add(5);  ⇒ +ve compare(5,10)
           ⇒ -ve compare(5,0)
t.add(20); ⇒ -ve compare(20,10)
           ⇒ -ve compare(20,15)
t.add(20); ⇒ -ve compare(20,10)
           ⇒ -ve compare(20,15)
           ⇒ 0   compare(20,20)
S.O.p(t) ⇒ [20,15,10,5,0]
```



Left - Node - Right-

[20,15,10,5,0]

⇒ At line ①, if we are not passing comparator object then internally JVM will call compareTo() method which is maint for Default Natural sorting order.

In this case, the o/p is [0,5,10,15,20]

→ At line ①, if we are passing the comparator object, in this case JVM will call 'compare' method, which is meant for customized sorting. In this case o/p is [20, 15, 10, 5, 0].

epi-9
Time: 42:50 ⇒ Various possible implementations of compare() method.

```
public class TreeSetDemo4
{
    p s v m (String[] args)
    {
        TreeSet t = new TreeSet(new MyComparator());
        t.add(10); t.add(0); t.add(15);
        t.add(5); t.add(20); t.add(20);
    }
}

~~public~~ class MyComparator extends Comparator
{
    public ~~int~~ compare(Object obj1, Object obj2)
    {
        Integer I1 = (Integer) obj1;
        Integer I2 = (Integer) obj2;

        return I1.compareTo(I2);
    }
}
```

↳ Default Natural sorting order
[Ascending order] → [0, 5, 10, 15, 20]

② return -I1.compareTo(I2);

↳ [Descending order] [20, 15, 10, 5, 0]