```
System.out.println("A".compareTo("Z"));// -ve value
System.out.println("Z".compareTo("K"));// +ve value
System.out.println("Z".compareTo("Z"));// 0
```

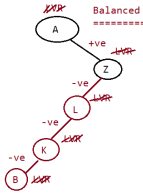
Rules while constructing a binary tree

-ve means in binary tree the node should be to the left
+ve means in binary tree the node should be to the right
zero means in binary tree the nodes are duplicated

```
TreeSet ts =new TreeSet();
    ts.add("A");
    ts.add("Z"); "Z".compareTo("A");
    ts.add("L"); "L".compareTo("A");"L".compareTo("Z");
    ts.add("K"); "K".compareTo("A");"K".compareTo("Z");
                                    "K".compareTo("L");

    ts.add("B"); "B".compareTo("A"); "B".compareTo(Z")
                 "B".compareTo("L"); "B".compareTo("K");

 System.out.println(ts); [ A B K L Z ]
```

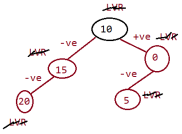


Balanced Binary Tree
====================

A B K L Z

```
TreeSet ts =new TreeSet(new MyCompartor());
   ts.add(10);
   ts.add(0); compare(0,10)
   ts.add(15);compare(15,10)
   ts.add(5); compare(5,10)
              compare(5,0)
   ts.add(20);compare(20,10)
              compare(20,15)
   ts.add(20); compare(20,10)
               compare(20,15)
               compare(20,20)

        20 15 10 5 0
```




```
compare(Object obj1,Object obj2)
 |=> returns -ve value, if obj1 has to come before obj2
 |=> returns +ve value, if obj1 has to come after  obj2
 |=> returns 0 value, if both obj1 and obj2 are equal
```

```
if (i1<i2)
    return +1; ⇦ reversing the logic
else if(i1>i2)
     return -1; ⇦ reversing the logic
else
     return 0;
```