Comparable interface

14 April 2025 19:00

It is present in Java.lang package and it can contains only one method. compareTo():

```
public int compareTo(Object o);

Obj1.compareTo(obj2)
```

- → returns -ve iff obj1 has to come before obj2.
- → returns -ve iff obj1 has to come after obj2
- → return 0 iff obj1 and obj2 are equal.

"A".compareTo("Z") ⇒ -ve
"Z".compareTo("A") ⇒ +ve
"A".compareTo("A") ⇒ 0
"A".compareTo(null) ⇒ NullPointerException

```
TreeSet t = new TreeSet();
    t.add("K");
    t.add("Z");          "Z".compareTo("K")       +ve
    t.add("A");          "A".compareTo("Z")       -ve
                         again,
                             "A".compareTo("K")   -ve
```

| "A" | "K" | "Z" | |

If we are depending on "default natural sorting order", then while adding objects into the TreeSet, JVM will call comparTo() method.

Note: If DNSO is not available or if we are not satisfied with DNSO then we can go for customized sorting by using Comparator(I).

Comparable(I) is ment for DNSO, whereas Comparator(I) is ment for customized sorting order.

Comparator(I)

Present in java.util package. And, it define two abstract methods.

① compare() ② equals().

① public int compare(Object ob1, Object ob2)
- → returns -ve iff obj1 has to come before obj2.

→ returns -ve iff obj1 has to come before obj2.
→ returns -ve iff obj1 has to come after obj2
→ return 0 iff obj1 and obj2 are equal.

2 public boolean equals(Object obj)

Wherever we are implementing comparator interface, compulsory we should provide only. compare() method and we are not required to provide equals() method because it is already available to our class from object class through inheritance.

```
public class MyComparator implements Comparator {
        public int compare(Object obj1, Object obj2){
              - - - - -
              - - - - -
        }
}
```

```
package Chapter1;
import java.util.Comparator;
public class MyComparator implements
Comparator{
    @Override
    public int compare(Object ob1, Object ob2){
        Integer i1 = (Integer)ob1;
        Integer i2 = (Integer)ob2;
        return (i1 < i2)?1:(i1 > i2)? -1: 0;
    }
}
```

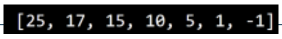
← logic of descending

```
package Chapter1;
import java.util.TreeSet;
public class IntegerTreeSetDemo {
    public static void main(String[] args) {
        MyComparator c = new MyComparator();
        TreeSet t = new TreeSet<>(c);
        t.add(10);
        t.add(-1);
        t.add(15);
        t.add(17);
        t.add(1);
        t.add(5);
        t.add(25);
        System.out.println(t);
    }
}
```

Comparator object is passed here.

← here compare function is called automatically for comparison of value.

Output:



```
[25, 17, 15, 10, 5, 1, -1]
```