→ An Object is said to be comparable iff corresponding class implements comparable interface.

→ String class and all wrapper classes already implement comparable interface but StringBuffer class does not implements Comparable interface. Hence, we got CCE in the above example.

⇒ Comparable Interface (I) :-

→ It is present in java.lang package and it contains only one method. i.e. compareTO():

Syntax :-

```
public int compareTo(Object obj).
```

obj1.compareTo(obj2)

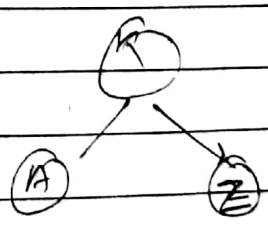
- → returns -ve :- iff obj1 has to come before obj2
- → returns +ve :- iff obj1 has to come after obj2
- → return 0 : iff obj1 and obj2 are equal.

Ex :-

```
S.O.p("A".compareTo("Z"));  ⇒ -ve (-25)
S.O.p("Z".compareTo("K"));  ⇒ +ve (15)
S.O.p("A".compareTo("A"));  ⇒ 0   (0)
S.O.p("A".compareTo(null));  ⇒ NPE
```

⇒ If we are depending on default Natural Sorting order then while adding Objects into the treeSet, JVM will call compareTO() method.

```
TreeSet t = new TreeSet();
t.add("K");
t.add("Z");  ⇒ +ve "Z".compareTo("K");
t.add("A");  ⇒ -ve "A".compareTo("K");
t.add("A");  ⇒ 0   "A".compareTo("A");
S.O.p(t) ⇒ [A,K,Z]
```



Obj1.compareTo(obj2)

↳ which is the object which already inserted.

NOTE :—

If Default Natural Sorting order not available or if we are not satisfied with default Natural sorting order then we can go for customized sorting by using Comparator interface.

eg-9 ⇒ Comparator interface :—

⇒ Comparator present in java.util package and it defines 2 methods : compare() and equals().

Syntax :—

① public int compare(Object obj1, Object obj2);
- returns -ve iff obj1 has to come before obj2
- " +ve " " " " " " after obj2
- " 0 iff obj1 and obj2 are equals.

② public boolean equals(Object obj);

⇒ Whenever we are implementing comparator interface, compulsary we should provide the implementation only for compare() method. and we are not required to provide implementation for equals() method because it is already available to our class from Object class through inheritance.

⇒ WAP to insert integer objects into the TreeSet where the sorting order is descending order

Soln ⇒
```
import java.util.*;
public class TreeSetDemo3
{
  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);
```
(1)