Comparable Interface

/swap number [20 , 15, 10 , 5 , 25 , 8]

```
for (     ) {
    (20 > 15) {
} swap
```

Note:

→ Comparable interface is meant for default sorting order

→ There is one method present inside the comparable interface called compareTo()

return type integer

→ java.lang package it was present

obj1 . compareTo (obj2) [1 , -1 , 0]

"abc" , "a"

5 − 3 +ve [5, 3] → [3, 5] swapping

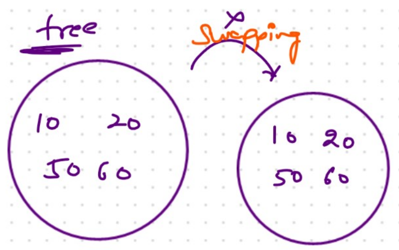
3 − 5 −ve [3 , 5] → [3, 5] No swapping

5 − 5 0 [5 , 5] → [5] eliminate

By default:

✓ Number : asc

✓ alpha/String : dictionary order



→ if we want to create our own sorting order customized then we need to implement "Comparator interface"

Comparator (java.util)

public int — compare (object o1, object o2) ✓

public boolean — equals (Object o) ×

Object → equals()

↑

A

→ when we are implementing comparator (I), we need to provide the implementation of compare()

```
Abstract A extends Object ~~implements Comparator~~ {
    @override
    compare () {
    }
}
```

→ It is not compulsory to override equals(), because Object class equals() would present in our class

① WAP to store Integers in a tree set in descending order

o2-o1

⓪

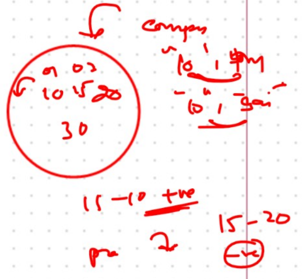
return 1 → insertion order will be in asc

return -1 → insertion order will be in desc

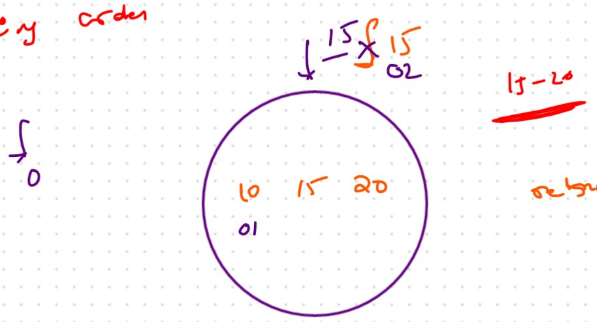
return 0 → no sort [Remove duplicate]

[5-5] =0
[10-10]
'5'-'5' =0

15

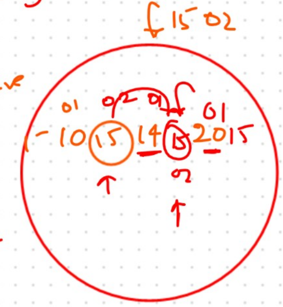


② WAP to store elements in a TreeSet while inserting [including duplicates] in ascending order



Iter1
o2-o1
→ 15-10 = +ve

Iter2 :
15-14
+ve



Iter 3
15-20 [stop]
-ve

③ WAP to store elements in TreeSet while insertion [include duplicates] & follow descending order

④ WAP to store the elements in a tree set in descending [reverse] of alphabetic order [input should string]

["apple", "banana", "cat"]
↓
["cat", "banana", "apple"]