Sorting

① Arrays.Sort in Java

i) Java
- → Collections.Sort → Allowed for list implementing classes (Eg:- ArrayList, LinkedList, vector)
- → Arrays.Sort
  - → Arrays of primitive (int, char, etc)
  - → Arrays of non-primitive (Integer, String, User defined objects)

ii)
* In objects sorting is to be stable.
  not important in case of primitives.

iii) Arrays.Sort by → java.util.Arrays;
Sorts in place i.e changes the array.

→ Only sorts in ascending (increasing) order.

for decreasing order, iterator can be used to reverse after sorting
Collections.reverse() can also be used.

iv) Arrays.sort(arr);

Arrays.Sort(arr, 1, 4); → Sorting a subarray.

Ⓐ In user defined class for using Arrays.sort(), comparable class/interface is to be used.

v) Wrapper class eg:- Integer arr[] = {5, 20, 10, 12}; (↗ replaces int, to implement collections.)

Arrays.Sort(arr, Collections.reverseOrder());

P.T.O

vi) Collections.sort → import java.util.*

Eg:- →

```
List<Integer> list1 = new ArrayList<Integer>();
list.add(20);
list.add(5);
Collections.sort(list1);
System.out.println(list);
```

vii)

```
Arrays.sort(arr, new Mycomp);
Collections.sort(arr);
```

} while using comparator interface.

viii)

Stability in Sorting.

Original Array → ('CSE', 10), ('ECE', 15), ('CSE', 5), ('ECE', 20)

Stable sorted → ('CSE', 5), ('CSE', 10), ('ECE', 15), ('ECE', 20)

Unstable sorted → Any other combination.

∴ The point in (In above ~~string~~ obj) if sorted by numbers then the original order in which names appeared must be preserved. If numbers are same.