comparable. Otherwise, we will get R.T.E saying ClassCastException.

→ If we are difining our own sorting by comparator, then keys need not be homogenious and comparable we can take heterogenious non comparable Objects also.

→ Whether we are depending on default natural sorting order or customized sorting order, there are no restriction for values. We can take heterogenious non comparable Object also.

⇒ Null acceptence :-

① For non empty Treemap, If we are trying to insert an entry with null key then we will get R.T.E saying N.P.E.

② For Empty Tree Map, As the first entry with null key is allowed but after inserting that entry, If we are trying to insert any other entry then we will get R.T.E saying NPE.

NOTE :- The above null acceptence rule applicable untill 1.6V only. From 1.7V onwards, null is not allowed for Key.

→ But for values, we can use null any no. of times. there is no restriction whether it is 1.6 or 1.7 versions.

Constructors :→

→ ① TreeMap t = new TreeMap();
creates an empty treemap where the elements will be inserted based on default sorted keys

② TreeMap t = new TreeMap(Comparator c);
↳ C.S.O of Keys

③ TreeMap t = new TreeMap(Map m);

④ TreeMap t = new TreeMap(SortedMap m)

Q:

```
import java.util.*;
public class Test
{
    p s v m(String[] args)
{
    TreeMap t = new TreeMap();
    m.put(100, "ZZZ");
    m.put(103, "YYY");
    m.put(101, "XXX");
    m.put(104, 106);
// m.put("FFFF", "XXX"); // → CCE
// m.put(null, "XXX"); → NPE
    S.O.p(m);
```

→ {100=ZZZ, 101=XXX, 103=YYY, 104=106}