WAP to Insert String and StringBuffer objects into TreeSet where Sorting order is increasing length order.
If 2 objects having same length then concides there Alphabetical order.

```
public class Test
{
    p s v m (String[] args)
{
  TreeSet t=new TreeSet(new MyComparator());
t.add("A");   t.add(new StringBuffer("ABC");
t.add(new StringBuffer("AA");   t.add("xx");
  t.add("ABCD");   t.add("A");
      S.O.p(t);
}}
class MyComparator Implements Comparator
{
   public int compare(Object obj1, Object obj2)
   {
    String s1 = obj1.toString();
    String s2 = obj2.toString();
     int l1 = obj1.length();
     int l2 = obj2.length();
     if (l1<l2)
     {   return -1;
     }
  else if (l1>l2)
      {
         return 1;
    }
    else {return s1.compareTo(s2);
}   }
```

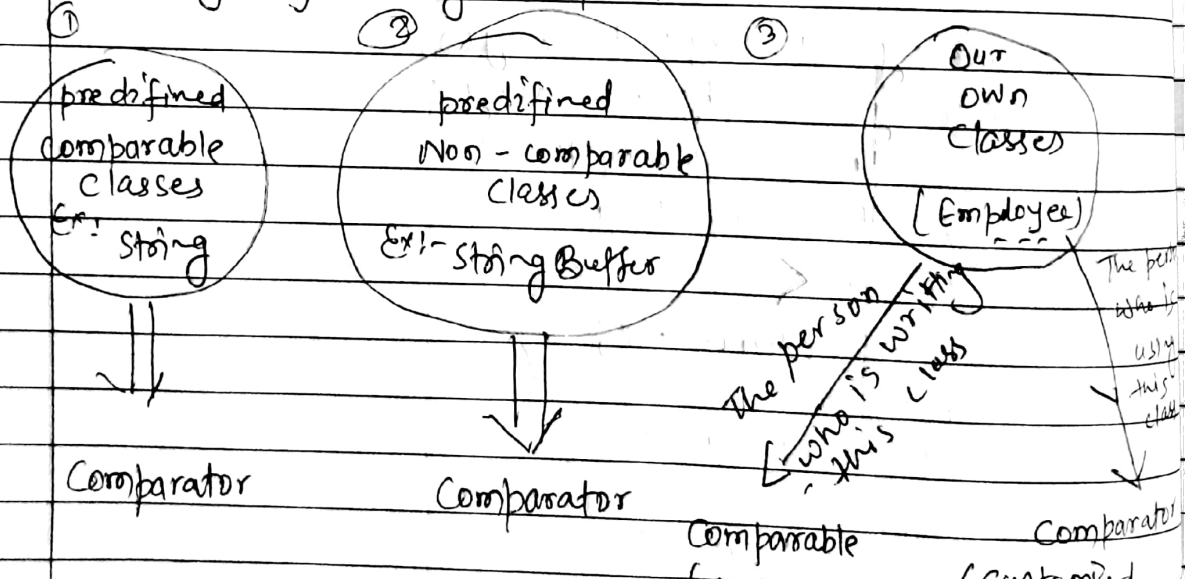
o/p ⇒ [A, AA, xx, ABC, ABCD]

⇒ Comparable vs comparator ..→

① For predefined comparable classes, default Natural sorting order already available. If we are not satisfied with that default natural sorting order then we can difine our own sorting by using comparator.

② For predifined non comparable classes (like StringBuffer) default natural sorting order not already available. We can define our own sorting by using comparator.

③ For our own classes, like Empdoyee, the person who is writting the class is responsible to define default natural sorting order by implementing comparable interface. The person who is using our class, if he is not satisfied with default natural sorting order then he can difine his own sorting by using comparator.



Ex:-

```
import java.util.*;
public class Test
{
    p s v m (String[] args)
    { Employee e1 = new Employee("nag", 100);
```