

SortedSet<E> interface :

It is the sub interface of Set<E> interface available from JDK 1.2V
As we know we can't perform sorting operation on HashSet and LinkedHashSet object by using Collections.sort(List<E> list)
In order to provide automatic sorting facility, java software people has provided SortedSet<E> interface.
By default It will provide defult natural sorting order.

It is a Set that maintains its elements in a sorted (ascending) order, either by their natural ordering (Comparable) or using a specified Comparator.

TreeSet<E>

public class TreeSet<E> extends AbstractSet<E> implements NavigableSet<E>, Comparable, Serializable

It is a predefined class available in java.util package under Set interface available from JDK 1.2v.

TreeSet class uses Red Black tree data structure.

It will sort the elements in natural sorting order i.e ascending order in case of number , and alphabetical order or Dictionary order in the case of String. In order to sort the elements according to user choice, It uses Comparable/Comparator interface.

It does not accept duplicate and null value (for natural sorting order)
(java.lang.NullPointerException)

It does not accept non comparable type of Objects if we try to insert it will throw a runtime exception i.e java.lang.ClassCastException

TreeSet implements NavigableSet.

NavigableSet extends SortedSet.

It contains 4 types of constructors :

- 1) TreeSet t1 = new TreeSet();
create an empty TreeSet object, elements will be inserted in using Comparable using compareTo().
- 2) TreeSet t2 = new TreeSet(Comparator c);
Customized sorting order.
- 3) TreeSet t3 = new TreeSet(Collection c);
loose coupling.
- 4) TreeSet t4 = new TreeSet(SortedSet s);
We can merge two TreeSet object to copy the data.

```
package com.ravi.tree_set_demo;

import java.util.TreeSet;

public class TreeSetDemo1
{
    public static void main(String[] args)
    {
        TreeSet<Object> ts1 = new TreeSet<>();
        ts1.add(78); //compareTo() of Integer class
        ts1.add(15);
        ts1.add(34);
        ts1.add(67);
        ts1.add(90);
        //ts1.add("Nit"); //java.lang.ClassCastException
        //ts1.add(null); //java.lang.NullPointerException
        System.out.println(ts1);
    }
}

package com.ravi.tree_set_demo;

import java.util.Iterator;
import java.util.TreeSet;

public class TreeSetDemo2
{
    public static void main(String[] args)
    {
        TreeSet<String> ts2 = new TreeSet<>();
        ts2.add("Mango");
        ts2.add("Grapes");
        ts2.add("Apple");
        ts2.add("Orange");

        System.out.println("In Ascending order :");
        ts2.forEach(System.out::println);

        System.out.println("In Descending order :");
        Iterator<String> descItr = ts2.descendingIterator();
        descItr.forEachRemaining(System.out::println);
    }
}

package com.ravi.tree_set_demo;

import java.util.TreeSet;

record Employee(Integer id, String name) implements Comparable<Employee>
{
    @Override
    public int compareTo(Employee e2)
    {
        return Integer.compare(this.id(), e2.id());
    }
}
public class TreeSetDemo3
{
    public static void main(String[] args)
    {
        TreeSet<Employee> ts1 = new TreeSet<>();
        ts1.add(new Employee(333, "Scott"));
        ts1.add(new Employee(111, "Zuber"));
        ts1.add(new Employee(222, "Aryan"));

        ts1.forEach(System.out::println);
    }
}

package com.ravi.tree_set_demo;

import java.util.TreeSet;

record Student(Integer id, String name)
{
}

public class TreeSetDemo4
{
    public static void main(String[] args)
    {
        TreeSet<Student> ts1 = new TreeSet<>((s1,s2)-> s1.name().compareTo(s2.name()));

        ts1.add(new Student(333, "Scott"));
        ts1.add(new Student(111, "Zuber"));
        ts1.add(new Student(222, "Aryan"));

        ts1.forEach(System.out::println);
    }
}

package com.ravi.tree_set_demo;

import java.util.TreeSet;

public class TreeSetDemo5
{
    public static void main(String[] args)
    {
        TreeSet<StringBuffer> ts1 = new TreeSet<>();
        ts1.add(new StringBuffer("B"));
        ts1.add(new StringBuffer("A"));
        ts1.add(new StringBuffer("C"));

        System.out.println(ts1);

        TreeSet<StringBuilder> ts2 = new TreeSet<>();
        ts2.add(new StringBuilder("B"));
        ts2.add(new StringBuilder("A"));
        ts2.add(new StringBuilder("C"));

        System.out.println(ts2);
    }
}

package com.ravi.tree_set_demo;

import java.util.ArrayList;
import java.util.TreeSet;

public class TreeSetDemo6 {
    public static void main(String[] args)
    {
        TreeSet<Character> ts1 = new TreeSet<>();
        ts1.add('C');
        ts1.add('B');
        ts1.add('A');
        System.out.println(ts1);

        TreeSet<Character> ts2 = new TreeSet<Character>(ts1);
        System.out.println(ts2);

        System.out.println(".....");

        ArrayList<String> listofCity = new ArrayList<>();
        listofCity.add("Hyderabad");
        listofCity.add("Hyderabad");
        listofCity.add("Pune");
        listofCity.add("Mumbai");
        listofCity.add("Ajmer");
        listofCity.add("Surat");

        TreeSet<String> cities = new TreeSet<String>(listofCity);
        System.out.println(cities);
    }
}

package com.ravi.tree_set_demo;

import java.util.*;

record Product(Integer id, String name, Double price)
{
}
public class TreeSetDemo7
{
    public static void main(String[] args)
    {
        TreeSet<Product> ts1 = new TreeSet<>((p1,p2)-> Integer.compare(p1.id(), p2.id()));

        ts1.add(new Product(333, "Mobile", 56789D));
        ts1.add(new Product(111, "Laptop", 46789D));
        ts1.add(new Product(222, "Camera", 96789D));

        System.out.println("Sorting based on the Id :");
        ts1.forEach(System.out::println);

        TreeSet<Product> ts2 = new TreeSet<>((p1,p2)-> Double.compare(p1.price(), p2.price()));

        ts2.add(new Product(333, "Mobile", 56789D));
        ts2.add(new Product(111, "Laptop", 46789D));
        ts2.add(new Product(222, "Camera", 96789D));

        System.out.println("Sorting based on the Price :");
        ts2.forEach(System.out::println);
    }
}

package com.ravi.tree_set_demo;

import java.util.*;

public class TreeSetDemo8
{
    public static void main(String[] args)
    {
        // Comparator that allows nulls
        Comparator<String> nullFriendlyComparator = (s1, s2) ->
        {
            if (s1 == null && s2 == null) return 0; // both null => equal
            if (s1 == null) return -1; // null is considered "smaller"
            if (s2 == null) return 1; // non-null is "greater"
            return s1.compareTo(s2); // normal comparison
        };
        TreeSet<String> set = new TreeSet<>(nullFriendlyComparator);

        set.add("Banana");
        set.add(null);
        set.add("Apple");
        set.add("Mango");

        System.out.println(set);
    }
}
```