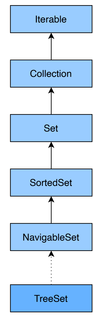
Java **TreeSet** class implements the **Set** interface that uses a tree for storage. It inherits the **AbstractSet** class and implements the **NavigableSet** interface.

Some of the features of **TreeSet** are:

1. **TreeSet** does not allow duplicate elements.
2. **TreeSet** class doesn't allow null elements.
3. Since elements are stored in a tree, the access and retrieval times are quite fast in a **TreeSet**.
4. The elements are stored in ascending order in a **TreeSet**.



TreeSet hierarchy

Difference between a HashSet and TreeSet

1. The HashSet allows one null element, whereas a TreeSet does not allow a null element.
2. The elements are stored in random order in a HashSet, whereas it is stored in sorted order in TreeSet.
3. HashSet is faster than Treeset for the operations like add, remove, contains, size, etc.

Creating a TreeSet

Before we look at the different methods to create a **TreeSet**, we will discuss one very important prerequisite to store the elements in a **TreeSet**. Since all the elements are stored in sorted order in a **TreeSet**, storing elements should either implement the **Comparable** interface or a custom **Comparator** while creating the **TreeSet**.

Let's discuss the different methods to create **TreeSet**.

Using the no-arg constructor

A **TreeSet** internally uses **TreeMap** which we will be discussing in a later section. When an instance of **TreeSet** is created using the no-arg constructor it internally creates an empty instance of **TreeMap**.

Below is the code syntax to create a **TreeSet**.

```
Set<Integer> set= new TreeSet<>();
```

Using a constructor with Comparator as an argument

If the objects that we are storing in a **TreeSet** do not implement the **Comparable** interface or if we need to store the elements in descending order, then we can provide a custom **Comparator** while creating the **TreeSet**. Now when the elements are stored in the **TreeSet**, they are sorted as per the logic provided by the **Comparator**.

Using a constructor with a Collection type argument

A **TreeSet** can be created from another Collection as well. The elements are stored in ascending order irrespective of the order in which the elements are stored in the Collection.

Using a constructor with the argument of type SortedSet

This constructor behaves as a copy constructor and creates a new sorted set with the same elements and the same ordering of the provided sorted set.

```
import java.util.LinkedList;
import java.util.List;
import java.util.TreeSet;

public class TreeSetDemo {

    public static void main(String args[]) {

        List<Integer> list = new LinkedList<>();
        list.add(21);
        list.add(32);
        list.add(44);
        list.add(11);
        list.add(54);

        TreeSet<Integer> set = new TreeSet<>(list);
        System.out.println("TreeSet elements in ascending order " + set);

    }
}
```

Run Save Reset

Inserting an element into a TreeSet

There are two methods to insert an element in TreeSet:

Inserting a single element

To insert a single element, we can use the `add(E e)` method. This method returns `true` if the element is inserted, and it returns `false` if the element is already present.

Inserting multiple elements

We can insert multiple elements in a TreeSet using the `addAll(Collection<> c)` method.

```
import java.util.Comparator;
import java.util.TreeSet;

public class TreeSetDemo {

    public static void main(String args[]) {
        TreeSet<Integer> set = new TreeSet<>();
        set.add(21);
        set.add(32);
        set.add(44);
        set.add(11);
        set.add(54);
        System.out.println("TreeSet elements in ascending order " + set);


        // This TreeSet will store the elements in reverse order.
        TreeSet<Integer> reverseSet = new TreeSet<>(Comparator.reverseOrder());
        reverseSet.add(21);
        reverseSet.add(32);
        reverseSet.add(44);
        reverseSet.add(11);
        reverseSet.add(54);
        System.out.println("TreeSet elements in descending order " + reverseSet);
    }

}
```

Run Save Reset