TreeSet

A **TreeSet in Java** is another important implementation of the Set interface that is similar to the HashSet class, with one added improvement.

It sorts elements in ascending order while HashSet does not maintain any order.

Java TreeSet implements SortedSet interface. It is a collection for storing a set of unique elements (objects) according to their natural ordering.

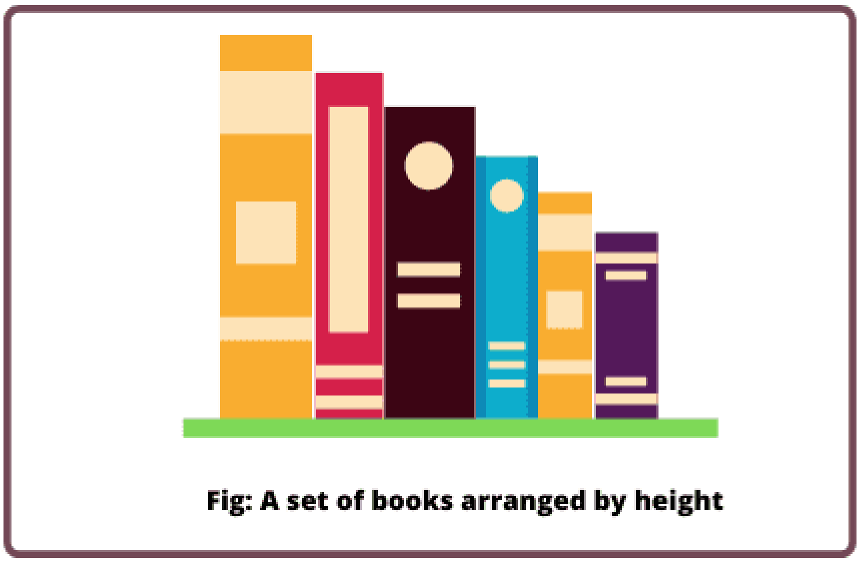
Fig: A set of books arranged by height

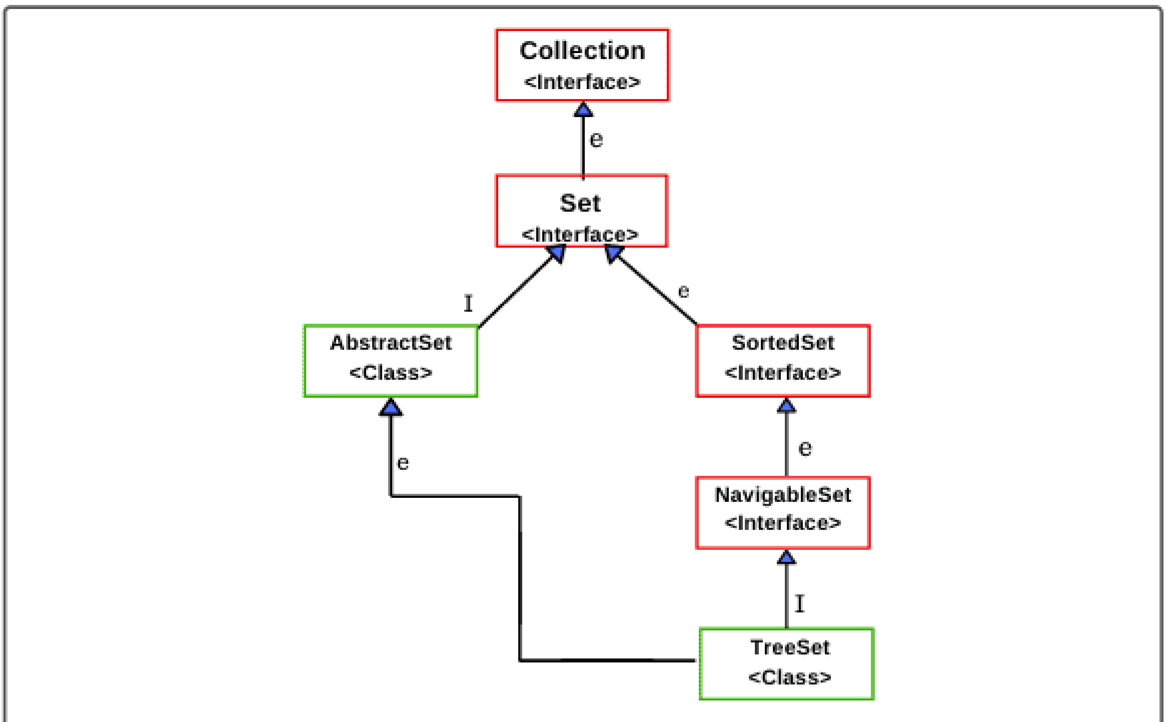

Fig: Java TreeSet Hierarchy Diagram

```
import java.util.Set;
import java.util.TreeSet;
public class TreeSetEx1 {
public static void main(String[] args)
{
// Create a tree set.
   Set<String> ts = new TreeSet<>();

// Check Set is empty or not.
   boolean empty = ts.isEmpty();
   System.out.println("Is TreeSet empty: " +empty);

// Checking the size of TreeSet before adding elements into it.
   int size = ts.size();
   System.out.println("Size of TreeSet: " +size);

// Adding elements into TreeSet.
   ts.add("India"); // ts.size() is 1.
   ts.add("USA"); // ts.size() is 2.
   ts.add("Australia"); // ts.size() is 3.
   ts.add("New zealand"); // ts.size() is 4.
   ts.add("Switzerland"); // ts.size() is 5.

   System.out.println("Sorted TreeSet: " +ts);
   int size2 = ts.size();
   System.out.println("Size of TreeSet after adding elements: " +size2);
  }
}
```

```
Output:
      Is TreeSet empty: true
      Size of TreeSet: 0
      Sorted TreeSet: [Australia, India, New zealand, Switzerland, USA]
      Size of TreeSet after adding elements: 5
```

```
import java.util.TreeSet;
public class TreeSetEx2 {
public static void main(String[] args)
{
// Creating an TreeSet object of type String.
   TreeSet<String> ts = new TreeSet<>();

// Adding string objects to tree set.
   ts.add("India");
   ts.add("USA");
   ts.add("Australia");
   ts.add("New zealand");
   ts.add("Switzerland");

// Checking for a specific element in set.
   boolean element = ts.contains("USA");
   System.out.println("Is USA in TreeSet: " +element);

// Removing element from the tree set.
   ts.remove("New zealand");
   System.out.println("Sorted tree set: " +ts);
   ts.clear();
   System.out.println("Elements in tree set: " +ts);
  }
}
```

```
Output:
       Is USA in TreeSet: true
       Elements in tree set: [Australia, India, Switzerland, USA]
       Elements in tree set: []
```

```
import java.util.HashSet;
import java.util.Set;
import java.util.SortedSet;
import java.util.TreeSet;
public class TreeSetEx3 {
public static void main(String[] args)
{
// Creating a set object and adding elements to it.
   Set<String> s = new HashSet<>();
   s.add("Delhi");
   s.add("New York");
   s.add("Paris");
   s.add("London");
   s.add("Delhi"); // Adding duplicate elements.

   TreeSet<String> ts = new TreeSet<>(s);
   System.out.println("Sorted TreeSet: " +ts);

// Using methods of SortedSet interface.
   System.out.println("First Element: " +ts.first());
   System.out.println("Last Element: " +ts.last());
   System.out.println("HeadSet Elements: " +ts.headSet("London"));
   System.out.println("TailSet Elements: " +ts.tailSet("London"));

   SortedSet<String> subSet = ts.subSet("Delhi", "Paris");
   System.out.println("SubSet Elements: " +subSet);
   System.out.println("Sorted Set: " +ts.comparator()); // It will return null because
natural ordering is used.
  }
}
```

```
Output:
      Sorted TreeSet: [Delhi, London, New York, Paris]
      First Element: Delhi
      Last Element: Paris
      HeadSet Elements: [Delhi]
      TailSet Elements: [London, New York, Paris]
      SubSet Elements: [Delhi, London, New York]
      Sorted Set: null
```

```
import java.util.TreeSet;
public class TreeSetEx4 {
public static void main(String[] args)
{
// Creating TreeSet object and adding elements to it.
   TreeSet<Integer> ts = new TreeSet<>();

   ts.add(25);
   ts.add(80);
   ts.add(05);
   ts.add(100);
   ts.add(90);
   ts.add(200);
   ts.add(300);
   System.out.println("Sorted TreeSet: " +ts);

// Using methods of NavigableSet interface.
   System.out.println("Largest element less than 100: " +ts.lower(100));
   System.out.println("Smallest element greater than 100: " +ts.higher(100));
   System.out.println("Floor: " +ts.floor(85));
   System.out.println("Ceiling: " +ts.ceiling(10));

   System.out.println(ts.pollFirst()); // Remove and retrieve the first element from t
he set.
   System.out.println(ts.pollLast()); // Remove and retrieve the last element from the
set.
   System.out.println("New Treeset: " +ts);

   System.out.println("HeadSet: " +ts.headSet(90,true));
   System.out.println("SubSet: " +ts.subSet(90, true, 200, true));
  }
}
```

```
Output:
      Sorted TreeSet: [5, 25, 80, 90, 100, 200, 300]
      Largest element less than 100: 90
      Smallest element greater than 100: 200
      Floor: 80
      Ceiling: 25
      5
      300
      New Treeset: [25, 80, 90, 100, 200]
      HeadSet: [25, 80, 90]
      SubSet: [90, 100, 200]
```

```
import java.util.Iterator;
import java.util.TreeSet;
public class KeySetDemo {
public static void main(String[] args)
{
 TreeSet<Integer> ts = new TreeSet<>();
  ts.add(25);
  ts.add(80);
  ts.add(05);
  ts.add(100);
  ts.add(90);
 System.out.println("Sorted TreeSet:");

// Traversing elements.
   Iterator<Integer> itr = ts.iterator();
   while(itr.hasNext())
   {
      System.out.println(itr.next());
   }
   System.out.println("Iterating elements through Iterator in descending order");
   Iterator<Integer> it = ts.descendingIterator();
   while(it.hasNext())
   {
     System.out.println(it.next());
   }
  }
}
```

```
Output:
      Sorted TreeSet:
```

```
5
25
80
90
100
Iterating elements through Iterator in descending order
100
90
80
25
5
```