JAVA - THE TREESET CLASS

TreeSet provides an implementation of the Set interface that uses a tree for storage. Objects are stored in sorted, ascending order.

Access and retrieval times are quite fast, which makes TreeSet an excellent choice when storing large amounts of sorted information that must be found quickly.

The TreeSet class supports four constructors. The first form constructs an empty tree set that will be sorted in ascending order according to the natural order of its elements:

```
TreeSet( )
```

The second form builds a tree set that contains the elements of c.

```
TreeSet(Collection c)
```

The third form constructs an empty tree set that will be sorted according to the comparator specified by comp.

```
TreeSet(Comparator comp)
```

The fourth form builds a tree set that contains the elements of ss:

```
TreeSet(SortedSet ss)
```

Apart from the methods inherited from its parent classes, TreeSet defines the following methods:

SN	Methods with Description
1	**void add(Object o)** Adds the specified element to this set if it is not already present.
2	**boolean addAll(Collection c)** Adds all of the elements in the specified collection to this set.
3	**void clear()** Removes all of the elements from this set.
4	**Object clone()** Returns a shallow copy of this TreeSet instance.
5	**Comparator comparator()** Returns the comparator used to order this sorted set, or null if this tree set uses its elements natural ordering.
6	**boolean contains(Object o)** Returns true if this set contains the specified element.
7	**Object first()**

Returns the first (lowest) element currently in this sorted set.

8 **SortedSet headSet(Object toElement)**

Returns a view of the portion of this set whose elements are strictly less than toElement.

9 **boolean isEmpty()**

Returns true if this set contains no elements.

10 **Iterator iterator()**

Returns an iterator over the elements in this set.

11 **Object last()**

Returns the last (highest) element currently in this sorted set.

12 **boolean remove(Object o)**

Removes the specified element from this set if it is present.

13 **int size()**

Returns the number of elements in this set (its cardinality).

14 **SortedSet subSet(Object fromElement, Object toElement)**

Returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.

15 **SortedSet tailSet(Object fromElement)**

Returns a view of the portion of this set whose elements are greater than or equal to fromElement.

Example:

The following program illustrates several of the methods supported by this collection:

```
import java.util.*;

public class TreeSetDemo {

   public static void main(String args[]) {
      // Create a tree set
      TreeSet ts = new TreeSet();
      // Add elements to the tree set
      ts.add("C");
      ts.add("A");
      ts.add("B");
      ts.add("E");
      ts.add("F");
      ts.add("D");
      System.out.println(ts);
   }
}
```

This would produce the following result:

```
[A, B, C, D, E, F]
```