JAVA.UTIL.TREESET.SUBSET METHOD

Description

The **subSet***EfromElement, booleanfromInclusive, EtoElement, booleantoInclusive* method is used to return a view of the portion of this set whose elements range from fromElement to toElement. If fromElement and toElement are equal, the returned set is empty unless fromExclusive and toExclusive are both true.

Declaration

Following is the declaration for **java.util.TreeSet.subSet** method.

```
public NavigableSet<E> subSet(E fromElement,boolean fromInclusive,E toElement,boolean
toInclusive)
```

Parameters

- **fromElement** -- This is the low endpoint of the returned set.
- **fromInclusive** -- This is true if the low endpoint is to be included in the returned view.
- **toElement** -- This is the high endpoint of the returned set.
- **toInclusive** -- This is true if the high endpoint is to be included in the returned view.

Return Value

The method call returns a view of the portion of this set whose elements range from fromElement, inclusive, to toElement, exclusive.

Exception

- **ClassCastException** -- This is thrown if fromElement and toElement cannot be compared to one another using this set's comparator.
- **NullPointerException** -- This is thrown if fromElement or toElement is null and this set uses natural ordering, or its comparator does not permit null elements.
- **IllegalArgumentException** -- This is thrown if fromElement is greater than toElement; or if this set itself has a restricted range, and fromElement or toElement lies outside the bounds of the range.

Example

The following example shows the usage of java.util.TreeSet.subSet method.

```
package com.tutorialspoint;

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

public class TreeSetDemo {
   public static void main(String[] args) {
   // creating a TreeSet
   TreeSet <Integer>treeadd = new TreeSet<Integer>();
   TreeSet <Integer>treesubset = new TreeSet<Integer>();

   // adding in the tree set
   treeadd.add(1);
   treeadd.add(2);
   treeadd.add(3);
```

```
   treeadd.add(4);
   treeadd.add(5);
   treeadd.add(6);
   treeadd.add(7);
   treeadd.add(8);

   // creating subset
   treesubset=(TreeSet)treeadd.subSet(3, true, 7, true);

   // create iterator
   Iterator iterator;
   iterator = treesubset.iterator();

   // displaying the Tree set data
   System.out.println("Tree subset data: ");
   while (iterator.hasNext()){
   System.out.println(iterator.next() + " ");
   }
   }
}
```

Let us compile and run the above program, this will produce the following result.

```
Tree subset data:
3
4
5
6
7
```