JAVA.UTIL.TREEMAP.HEADMAP METHOD

http://www.tutorialspoint.com/java/util/treemap_headmap_inclusive.htm

Description

The**headMap*KtoKey, booleaninclusive*** method is used to return a view of the portion of this map whose keys are less than *orequalto, ifinclusiveistrue*toKey.

Declaration

Following is the declaration for **java.util.TreeMap.headMap** method.

```
public NavigableMap<K,V> headMap(K toKey,boolean inclusive)
```

Parameters

- **toKey** -- This is the high endpoint of the keys in the returned map.
- **inclusive** -- 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 map whose keys are less than *orequalto, ifinclusiveistrue* toKey.

Exception

- **ClassCastException** -- This exception is thrown if toKey is not compatible with this map's comparator.
- **NullPointerException** -- This exception is thrown if toKey is null and this map uses natural ordering, or its comparator does not permit null keys.
- **IllegalArgumentException** -- This exception is thrown if this map itself has a restricted range, and toKey lies outside the bounds of the range.

Example

The following example shows the usage of java.util.TreeMap.headMap method.

```
package com.tutorialspoint;

import java.util.*;

public class TreeMapDemo {
    public static void main(String[] args) {
    // creating maps
    TreeMap<Integer, String> treemap = new TreeMap<Integer, String>();
    NavigableMap<Integer, String> treemapheadincl = new TreeMap<Integer, String>();

    // populating tree map
    treemap.put(2, "two");
    treemap.put(1, "one");
    treemap.put(3, "three");
    treemap.put(6, "six");
    treemap.put(5, "five");

    // getting head map inclusive 3
    treemapheadincl=treemap.headMap(3,true);

    System.out.println("Checking values of the map");
    System.out.println("Value is: "+ treemapheadincl);
    }
}
```

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

```
Checking values of the map
Value is: {1=one, 2=two, 3=three}
```