**How to iterate TreeMap in reverse order in Java**

[**JAVA COLLECTIONS**](http://beginnersbook.com/category/java-collections/)

By default TreeMap elements are sorted in ascending order of keys. We can iterate the TreeMap in reverse order to display the elements in descending order of keys.

**Display TreeMap elements in reverse order:**

import java.util.\*;

class TreeMapDemo {

public static void main(String args[]) {

Map<String, String> treemap =

new TreeMap<String, String>(Collections.reverseOrder());

// Put elements to the map

treemap.put("Key1", "Jack");

treemap.put("Key2", "Rick");

treemap.put("Key3", "Kate");

treemap.put("Key4", "Tom");

treemap.put("Key5", "Steve");

Set set = treemap.entrySet();

Iterator i = set.iterator();

// Display elements

while(i.hasNext()) {

Map.Entry me = (Map.Entry)i.next();

System.out.print(me.getKey() + ": ");

System.out.println(me.getValue());

}

}

}

**Output:**

Key5: Steve

Key4: Tom

Key3: Kate

Key2: Rick

Key1: Jack

As you can see elements are displayed in the reverse order of keys.

**More about Collections.reverseOrder() from**[**javadoc**](http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#reverseOrder())**:**  
public static Comparator reverseOrder(): Returns a comparator that imposes the reverse of the natural ordering on a collection of objects that implement the Comparable interface. (The natural ordering is the ordering imposed by the objects’ own compareTo method.) This enables a simple idiom for sorting (or maintaining) collections (or arrays) of objects that implement the Comparable interface in reverse-natural-order. For example, suppose a is an array of strings. Then: Arrays.sort(a, Collections.reverseOrder());

sorts the array in reverse-lexicographic (alphabetical) order. The returned comparator is serializable.

**Returns:**  
A comparator that imposes the reverse of the natural ordering on a collection of objects that implement the Comparable interface.