**How to sort HashMap in Java by Keys and Values**

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

As we know that HashMap doesn’t preserve any order by default. If there is a need we need to sort it explicitly based on the requirement. In this tutorial we will learn how to sort HashMap **by keys** using TreeMap and **by values**using **Comparator**.

**HashMap Sorting by Keys**

In this example we are sorting the HashMap based on the keys using the TreeMap collection class.

package beginnersbook.com;

import java.util.HashMap;

import java.util.Map;

import java.util.TreeMap;

import java.util.Set;

import java.util.Iterator;

public class Details {

public static void main(String[] args) {

HashMap<Integer, String> hmap = new HashMap<Integer, String>();

hmap.put(5, "A");

hmap.put(11, "C");

hmap.put(4, "Z");

hmap.put(77, "Y");

hmap.put(9, "P");

hmap.put(66, "Q");

hmap.put(0, "R");

System.out.println("Before Sorting:");

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

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

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

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

}

Map<Integer, String> map = new TreeMap<Integer, String>(hmap);

System.out.println("After Sorting:");

Set set2 = map.entrySet();

Iterator iterator2 = set2.iterator();

while(iterator2.hasNext()) {

Map.Entry me2 = (Map.Entry)iterator2.next();

System.out.print(me2.getKey() + ": ");

System.out.println(me2.getValue());

}

}

}

Output:

Before Sorting:

0: R

4: Z

5: A

66: Q

9: P

77: Y

11: C

After Sorting:

0: R

4: Z

5: A

9: P

11: C

66: Q

77: Y

**HashMap Sorting by Values**

In this example we are sorting HashMap by values using Comparator.

package beginnersbook.com;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.LinkedList;

import java.util.List;

import java.util.Map;

import java.util.Set;

public class HMapSortingByvalues {

public static void main(String[] args) {

HashMap<Integer, String> hmap = new HashMap<Integer, String>();

hmap.put(5, "A");

hmap.put(11, "C");

hmap.put(4, "Z");

hmap.put(77, "Y");

hmap.put(9, "P");

hmap.put(66, "Q");

hmap.put(0, "R");

System.out.println("Before Sorting:");

Set set = hmap.entrySet();

Iterator iterator = set.iterator();

while(iterator.hasNext()) {

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

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

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

}

Map<Integer, String> map = sortByValues(hmap);

System.out.println("After Sorting:");

Set set2 = map.entrySet();

Iterator iterator2 = set2.iterator();

while(iterator2.hasNext()) {

Map.Entry me2 = (Map.Entry)iterator2.next();

System.out.print(me2.getKey() + ": ");

System.out.println(me2.getValue());

}

}

private static HashMap sortByValues(HashMap map) {

List list = new LinkedList(map.entrySet());

// Defined Custom Comparator here

Collections.sort(list, new Comparator() {

public int compare(Object o1, Object o2) {

return ((Comparable) ((Map.Entry) (o1)).getValue())

.compareTo(((Map.Entry) (o2)).getValue());

}

});

// Here I am copying the sorted list in HashMap

// using LinkedHashMap to preserve the insertion order

HashMap sortedHashMap = new LinkedHashMap();

for (Iterator it = list.iterator(); it.hasNext();) {

Map.Entry entry = (Map.Entry) it.next();

sortedHashMap.put(entry.getKey(), entry.getValue());

}

return sortedHashMap;

}

}

Output:

Before Sorting:

0: R

4: Z

5: A

66: Q

9: P

77: Y

11: C

After Sorting:

5: A

11: C

9: P

66: Q

0: R

77: Y

4: Z