Sort HashMap by Keys

In this post we will see how to sort HashMap using keys. As we know that keys are unique in HashMap. No keys are same.

We will use 3 different ways to do this.

1. **Using keySet() method and ArrayList<E> class**
2. **Using keySet() method, TreeSet<E> class and LinkedHashMap<K,V> class**
3. **Using TreeMap<K, V> class**

Let us discuss all of the methods in details.

1. **Using keySet() method and ArrayList<E> class**
   1. First we will take keySet() from HashMap<K,V> and use addAll() method of ArrayList<E> to insert all keys into the ArrayList<E>.
   2. Then we will use Collections.sort(..) method to sort the keys.
   3. Now, keys are sorted. Define a new LinkedHashMap<K, V> that will hold new Key – Value mappings.
   4. Iterate through the ArrayList<E> that contains keys and put them in LinkedHashMap with specified value from HashMap<K, V>.

Below is the code.

**public** **static** Map<String, String> sortMap(Map<String, String> map) {

/\*\*

\* Define new ArrayList<String>

\* \*/

List<String> keys = **new** ArrayList<String>();

/\*\*

\* use addAll(..) method to insert entire keySet in ArrayList.

\* \*/

keys.addAll(map.keySet());

/\*\*

\* Sort the List.

\* \*/

Collections.*sort*(keys);

/\*\*

\* Define new LinkedHashMap.

\* \*/

Map<String, String> newMap = **new** LinkedHashMap<String, String>();

/\*\*

\* Insert data in LinkedHashMap. The data will be in sorted order because

\* LinkedHashMap maintains the order of insertion.

\* \*/

**for** (String str : keys) {

newMap.put(str, map.get(str));

}

System.***out***.println("Sorted Map " + newMap);

**return** newMap;

}

1. **Using keySet() method, TreeSet<E> class and LinkedHashMap<K,V> class**
   1. Insert keySet() into the constructor of TreeSet<E> class.
   2. We don’t need to sort the keys because it will be handled by TreeSet<E> implicitly.
   3. Then define a new LinkedHashMap. Now iterate through the TreeSet and insert key as element of TreeSet and value from old map.

Below is the code.

**public** **static** **void** sortMapUsingTreeSet(Map<String, String> map) {

/\*\*

\* Define a new TreeSet.

\* Insert entire keyset in TreeSet.

\* Remember TreeSet stores data in Sorted Order.

\* We don't need to sort the keySet externally.

\* It will be handled by TreeSet itself.

\* \*/

TreeSet<String> treeSet = **new** TreeSet<String>(map.keySet());

/\*\*

\* Define new LinkedHashMap.

\* \*/

Map<String, String> newMap = **new** LinkedHashMap<String, String>();

/\*\*

\* Insert data in LinkedHashMap. The data will be in sorted order because

\* LinkedHashMap maintains the order of insertion.

\* \*/

**for** (String element : treeSet) {

newMap.put(element, map.get(element));

}

System.***out***.println("Set method " + newMap);

}

1. **Using TreeMap<K, V> class**
   1. Define a new TreeMap<K, V> class and pass the map in constructor of TreeMap<K, V> class
   2. It’s done.

Below is the code.

**public** **static** **void** sortMapUsingTreeMapConstructor(Map<String, String> map) {

TreeMap<String, String> treeMap = **new** TreeMap<String, String>(map);

System.***out***.println("Map method " + treeMap);

}

Below is the code for entire program.

**package** org.collections;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.HashMap;

**import** java.util.LinkedHashMap;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.TreeMap;

**import** java.util.TreeSet;

**public** **class** HashMapExample {

**public** **static** **void** main(String[] args) {

Map<String, String> map = **new** HashMap<String, String>();

*populateMap*(map);

*displayMap*(map);

*sortMapUsingTreeMapConstructor*(map);

}

/\*\*

\* This method is used to populate the Map.

\* \*/

**public** **static** **void** populateMap(Map<String, String> map) {

map.put("Robert", "Baratheon");

map.put("Roose", "Bolton");

map.put("Tyrion", "Lannister");

map.put("Eddard", "Stark");

map.put("Benjen", "Stark");

map.put("Aegon", "Targaryen");

map.put("Edmurd", "Tully");

}

/\*\*

\* This method is used to display the map.

\* \*/

**public** **static** **void** displayMap(Map<String, String> map) {

System.***out***.println("--------------------------------------------------------");

**for** (Entry<String, String> entry : map.entrySet()) {

System.***out***.println(entry.getKey() + " " + entry.getValue());

}

System.***out***.println("--------------------------------------------------------");

System.***out***.println("\n");

}

/\*\*

\* Method 1

\* Uses sorting to sort the keys and then insert in LinkedHashMap.

\* \*/

**public** **static** Map<String, String> sortMap(Map<String, String> map) {

/\*\*

\* Define new ArrayList<String>

\* \*/

List<String> keys = **new** ArrayList<String>();

/\*\*

\* use addAll(..) method to insert entire keySet in ArrayList.

\* \*/

keys.addAll(map.keySet());

/\*\*

\* Sort the List.

\* \*/

Collections.*sort*(keys);

/\*\*

\* Define new LinkedHashMap.

\* \*/

Map<String, String> newMap = **new** LinkedHashMap<String, String>();

/\*\*

\* Insert data in LinkedHashMap. The data will be in sorted order because

\* LinkedHashMap maintains the order of insertion.

\* \*/

**for** (String str : keys) {

newMap.put(str, map.get(str));

}

*displayMap*(newMap);

**return** newMap;

}

/\*\*

\* Method 2

\* Uses TreeSet<E> constructor to insert entire

\* keySet in TreeSet<E>.

\* \*/

**public** **static** Map<String, String> sortMapUsingTreeSet(Map<String, String> map) {

/\*\*

\* Define a new TreeSet.

\* Insert entire keySet in TreeSet.

\* Remember TreeSet stores data in Sorted Order.

\* We don't need to sort the keySet externally.

\* It will be handled by TreeSet itself.

\* \*/

TreeSet<String> treeSet = **new** TreeSet<String>(map.keySet());

/\*\*

\* Define new LinkedHashMap.

\* \*/

Map<String, String> newMap = **new** LinkedHashMap<String, String>();

/\*\*

\* Insert data in LinkedHashMap. The data will be in sorted order because

\* LinkedHashMap maintains the order of insertion.

\* \*/

**for** (String element : treeSet) {

newMap.put(element, map.get(element));

}

*displayMap*(newMap);

**return** newMap;

}

/\*\*

\* Method 3

\* Uses TreeMap<K, V> class and it constructor to

\* insert entire Map into TreeMap<K,V>

\* \*/

**public** **static** Map<String, String> sortMapUsingTreeMapConstructor(Map<String, String> map) {

TreeMap<String, String> treeMap = **new** TreeMap<String, String>(map);

*displayMap*(treeMap);

**return** treeMap;

}

}

Click [here to learn how to iterate over Map in 5 different ways](http://data-structure-learning.blogspot.com/2015/05/different-ways-to-traverse-map.html).