Difference between HashMap and Hashtable

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

What is the Difference between HashMap and Hashtable? This is one of the frequently asked interview questions for Java/J2EE professionals. **HashMap** and **Hashtable** both classes implements **java.util.Map** interface, however there are differences in the way they work and their usage. Here we will discuss the differences between these classes.

HashMap vs Hashtable

1) HashMap is non-synchronized. This means if it's used in multithread environment then more than one thread can access and process the HashMap simultaneously.

Hashtable is synchronized. It ensures that no more than one thread can access the Hashtable at a given moment of time. The thread which works on Hashtable acquires a lock on it to make the other threads wait till its work gets completed.

2) HashMap allows one null key and any number of null values.

Hashtable doesn't allow null keys and null values.

3) HashMap implementation LinkedHashMap maintains the insertion order and TreeMap sorts the mappings based on the ascending order of keys.

Hashtable doesn't guarantee any kind of order. It doesn't maintain the mappings in any particular order.

4) Initially Hashtable was not the part of collection framework it has been made a collection framework member later after being retrofitted to implement the Map interface.

HashMap implements Map interface and is a part of collection framework since the beginning.

5) Another difference between these classes is that the Iterator of the HashMap is a fail-fast and it throws ConcurrentModificationException if any

other Thread modifies the map structurally by adding or removing any element except iterator's own remove() method. In Simple words fail-fast means: When calling iterator.next(), if any modification has been made between the moment the iterator was created and the moment next() is called, a ConcurrentModificationException is immediately thrown.

Enumerator for the Hashtable is not fail-fast.

For e.g.

HashMap:

```
HashMap hm= new HashMap();
....
....
Set keys = hm.keySet();
for (Object key : keys) {
    //it will throw the ConcurrentModificationException here
    hm.put(object & value pair here);
}
```

Hashtable:

```
Hashtable ht= new Hashtable();
....
.....
Enumeration keys = ht.keys();
 for (Enumeration en = ht.elements() ; en.hasMoreElements() ; en.nextElement()) {
      //No exception would be thrown here
      ht.put(key & value pair here);
 }
```

When to use HashMap and Hashtable?

1) As stated above the main difference between HashMap & Hashtable is synchronization. If there is a need of thread-safe operation then Hashtable can be used as all its methods are synchronized but it's a legacy class and should be avoided as there is nothing about it, which cannot be done by HashMap. For multi-thread environment I would recommend you to use ConcurrentHashMap (Almost similar to Hashtable) or even you can make the HashMap synchronized explicitly (Read here..).

2) Synchronized operation gives poor performance so it should be avoided until unless required. Hence for non-thread environment HashMap should be used without any doubt.