**Hashtable in java with example**

BY CHAITANYA SINGH | FILED UNDER: [JAVA COLLECTIONS](https://beginnersbook.com/category/java-collections/)

This class implements a hash table, which maps keys to values. Any non-null object can be used as a key or as a value. Hashtable is similar to [HashMap](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/) except it is synchronized. There are few more differences between HashMap and Hashtable class, you can read them in detail at: [Difference between HashMap and Hashtable](https://beginnersbook.com/2014/06/difference-between-hashmap-and-hashtable/).

# How to synchronize HashMap in Java with example

By Chaitanya Singh | Filed Under: [Java Collections](https://beginnersbook.com/category/java-collections/)

[HashMap is a non-synchronized collection class](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/). If we need to perform thread-safe operations on it then we must need to synchronize it explicitly. In this tutorial we will see how to synchronize HashMap.

#### Example:

In this example we have a HashMap<Integer, String> it is having integer keys and String type values. In order to synchronize it we are using [Collections.synchronizedMap(hashmap)](https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html#synchronizedMap(java.util.Map))  it returns a thread-safe map backed up by the specified HashMap.

**Important point to note in the below example:**  
Iterator should be used in a synchronized block even if we have synchronized the HashMap explicitly (As we did in the below code).

**Syntax:**

Map map = Collections.synchronizedMap(new HashMap());

...

//This doesn't need to be in synchronized block

Set set = map.keySet();

// Synchronizing on map, not on set

synchronized (map) {

// Iterator must be in synchronized block

Iterator iterator = set.iterator();

while (iterator.hasNext()){

...

}

}

**Complete Code:**

package beginnersbook.com;

import java.util.Collections;

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

import java.util.Iterator;

public class HashMapSyncExample {

public static void main(String args[]) {

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

hmap.put(2, "Anil");

hmap.put(44, "Ajit");

hmap.put(1, "Brad");

hmap.put(4, "Sachin");

hmap.put(88, "XYZ");

Map map= Collections.synchronizedMap(hmap);

Set set = map.entrySet();

synchronized(map){

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:

1: Brad

2: Anil

4: Sachin

88: XYZ

44: Ajit

In this tutorial we will see how to create a Hashtable, how to populate its entries and then we will learn how to display its key-value pairs using Enumeration. At the end of this article we will see Hashtable tutorials and methods of Hashtable class.

**Example**

import java.util.Hashtable;

import java.util.Enumeration;

public class HashtableExample {

public static void main(String[] args) {

Enumeration names;

String key;

// Creating a Hashtable

Hashtable<String, String> hashtable =

new Hashtable<String, String>();

// Adding Key and Value pairs to Hashtable

hashtable.put("Key1","Chaitanya");

hashtable.put("Key2","Ajeet");

hashtable.put("Key3","Peter");

hashtable.put("Key4","Ricky");

hashtable.put("Key5","Mona");

names = hashtable.keys();

while(names.hasMoreElements()) {

key = (String) names.nextElement();

System.out.println("Key: " +key+ " & Value: " +

hashtable.get(key));

}

}

}

**Output:**

Key: Key4 & Value: Ricky

Key: Key3 & Value: Peter

Key: Key2 & Value: Ajeet

Key: Key1 & Value: Chaitanya

Key: Key5 & Value: Mona

# Difference between HashMap and Hashtable

By Chaitanya Singh | Filed Under: [Java Collections](https://beginnersbook.com/category/java-collections/)

What is the Difference between HashMap and Hashtable? This is one of the frequently asked [interview questions](https://beginnersbook.com/2013/05/java-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](https://beginnersbook.com/2013/12/hashmap-in-java-with-example/) 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](https://docs.oracle.com/javase/6/docs/api/java/util/Hashtable.html) 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](https://beginnersbook.com/2013/12/linkedhashmap-in-java/) maintains the insertion order and [TreeMap](https://beginnersbook.com/2013/12/treemap-in-java-with-example/) 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](https://beginnersbook.com/java-collections-tutorials/) 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](https://docs.oracle.com/javase/6/docs/api/java/util/ConcurrentModificationException.html) 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](https://beginnersbook.com/2013/03/multithreading-in-java/) environment I would recommend you to use ConcurrentHashMap (Almost similar to Hashtable) or even you can make the HashMap synchronized explicitly ([Read here](https://beginnersbook.com/2013/12/how-to-synchronize-hashmap-in-java-with-example/)..).

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.

**Hashtable tutorials**

* [Hashtable example](https://beginnersbook.com/2014/07/hashtable-in-java-with-example/)
* [Sort Hashtable](https://beginnersbook.com/2014/06/how-to-sort-hashtable-in-java/)
* [Hashtable Iterator example](https://beginnersbook.com/2014/07/hashtable-iterator-example-java/)
* [Check key-value existence in Hashtable](https://beginnersbook.com/2014/07/check-key-value-existence-in-hashtable-example-java/)
* [Remove mapping from Hashtable](https://beginnersbook.com/2014/07/remove-mapping-from-hashtable-example-java/)
* [Remove all mappings from Hashtable](https://beginnersbook.com/2014/07/remove-all-mappings-from-hashtable-example-java/)
* [Get size of Hashtable](https://beginnersbook.com/2014/07/get-size-of-hashtable-example-in-java/)
* [Hashtable vs HashMap](https://beginnersbook.com/2014/06/difference-between-hashmap-and-hashtable/)

**Methods of Hashtable class:**

1) void clear(): Removes all the key-value mappings from Hashtable and makes it empty. Clears this hashtable so that it contains no keys..

2) Object clone(): Creates a shallow copy of this hashtable. All the structure of the hashtable itself is copied, but the keys and values are not cloned. This is a relatively expensive operation.

3) boolean contains(Object value): Tests if some key maps into the specified value in this hashtable. This operation is more expensive than the containsKey method.  
Note that this method is identical in functionality to containsValue, (which is part of the Map interface in the collections framework).

4) boolean isEmpty(): Tests if this hashtable maps no keys to values.

5) Enumeration keys(): Returns an enumeration of the keys contained in the hash table.

6) Object put(Object key, Object value): Maps the specified key to the specified value in this hashtable.

7) void rehash(): Increases the size of the hash table and rehashes all of its keys.

8) Object remove(Object key): Removes the key (and its corresponding value) from this hashtable.

9) int size(): Returns the number of key-value mappings present in Hashtable.

10) String toString(): Returns the string equivalent of a hash table.

11) boolean containsKey(Object key): Tests if the specified object is a key in this hashtable.

12) boolean containsValue(Object value): Tests if the specified object is a value in this hashtable. Returns true if some value equal to value exists within the hash table. Returns false if the value isn’t found.

13) Enumeration elements(): Returns an enumeration of the values contained in the hash table.

14) Object get(Object key): Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

**References:**

1. [Hashtable javadoc](https://docs.oracle.com/javase/7/docs/api/java/util/Hashtable.html)
2. [Enumeration javadoc](https://docs.oracle.com/javase/7/docs/api/java/util/Enumeration.html)