**Java ConcurrentHashMap Example and Iterator**

JULY 29, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [36 COMMENTS](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comments)

Today we will look into Java ConcurrentHashMap Example. If you are a Java Developer, I am sure that you must be aware of ConcurrentModificationException that comes when you want to modify the Collection object while using iterator to go through with all its element. Actually Java Collection Framework iterator is great example of **[iterator design pattern](http://www.journaldev.com/1716/iterator-design-pattern-java" \o "Iterator Design Pattern in Java – Example Tutorial)** implementation.

**Java ConcurrentHashMap**

Java 1.5 has introduced java.util.concurrent package with [Collection classes](http://www.journaldev.com/1260/collections-in-java-tutorial) implementations that allow you to modify your collection objects at runtime.

**ConcurrentHashMap Example**

ConcurrentHashMap is the class that is similar to HashMap but works fine when you try to modify your map at runtime.

Lets run a sample program to explore this:

ConcurrentHashMapExample.java

package com.journaldev.util;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

public class ConcurrentHashMapExample {

public static void main(String[] args) {

//ConcurrentHashMap

Map<String,String> myMap = new ConcurrentHashMap<String,String>();

myMap.put("1", "1");

myMap.put("2", "1");

myMap.put("3", "1");

myMap.put("4", "1");

myMap.put("5", "1");

myMap.put("6", "1");

System.out.println("ConcurrentHashMap before iterator: "+myMap);

Iterator<String> it = myMap.keySet().iterator();

while(it.hasNext()){

String key = it.next();

if(key.equals("3")) myMap.put(key+"new", "new3");

}

System.out.println("ConcurrentHashMap after iterator: "+myMap);

//HashMap

myMap = new HashMap<String,String>();

myMap.put("1", "1");

myMap.put("2", "1");

myMap.put("3", "1");

myMap.put("4", "1");

myMap.put("5", "1");

myMap.put("6", "1");

System.out.println("HashMap before iterator: "+myMap);

Iterator<String> it1 = myMap.keySet().iterator();

while(it1.hasNext()){

String key = it1.next();

if(key.equals("3")) myMap.put(key+"new", "new3");

}

System.out.println("HashMap after iterator: "+myMap);

}

}

When we try to run the above class, output is

ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}

ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 4=1, 2=1}

HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}

Exception in thread "main" [java.util.ConcurrentModificationException](http://www.journaldev.com/378/java-util-concurrentmodificationexception)

at java.util.HashMap$HashIterator.nextEntry(HashMap.java:793)

at java.util.HashMap$KeyIterator.next(HashMap.java:828)

at com.test.ConcurrentHashMapExample.main(ConcurrentHashMapExample.java:44)

Looking at the output, its clear that **ConcurrentHashMap** takes care of any new entry in the map whereas HashMap throws ConcurrentModificationException.

Lets look at the exception stack trace closely. The statement that has thrown Exception is:

String key = it1.next();

It means that the new entry got inserted in the HashMap but Iterator is failing. Actually Iterator on Collection objects are **fail-fast** i.e any modification in the structure or the number of entry in the collection object will trigger this exception thrown by iterator.

So How does iterator knows that there has been some modification in the HashMap. We have taken the set of keys from HashMap once and then iterating over it.

HashMap contains a variable to count the number of modifications and iterator use it when you call its next() function to get the next entry.

HashMap.java

/\*\*

\* The number of times this HashMap has been structurally modified

\* Structural modifications are those that change the number of mappings in

\* the HashMap or otherwise modify its internal structure (e.g.,

\* rehash). This field is used to make iterators on Collection-views of

\* the HashMap fail-fast. (See ConcurrentModificationException).

\*/

transient volatile int modCount;

Now to prove above point, lets change the code a little bit to come out of the iterator loop when we insert the new entry. All we need to do is add a break statement after the put call.

if(key.equals("3")){

myMap.put(key+"new", "new3");

break;

}

Now execute the modified code and the output will be:

ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}

ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 4=1, 2=1}

HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}

HashMap after iterator: {3=1, 2=1, 1=1, 3new=new3, 6=1, 5=1, 4=1}

Finally, what if we won’t add a new entry but update the existing key-value pair. Will it throw exception?

Change the code in the original program and check yourself.

//myMap.put(key+"new", "new3");

myMap.put(key, "new3");

If you get confused (or shocked) with the output, comment below and I will be happy to explain it further.

Did you noticed those angle brackets while creating our collection object and Iterator, it’s called generics in java and it’s very powerful when it comes to type-checking at compile time to remove ClassCastException at runtime, learn more about generics in [**Java Generics Example**](http://www.journaldev.com/1663/java-generics-example-method-class-interface).

Further Reading: [Java Collections Interview Questions](http://www.journaldev.com/1330/java-collections-interview-questions-and-answers) and [Iterator Design Pattern Java](http://www.journaldev.com/1716/iterator-design-pattern-java" \o "Iterator Design Pattern in Java – Example Tutorial).

FILED UNDER: [JAVA](http://www.journaldev.com/dev/java)

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [**Google Plus**](https://plus.google.com/118104420597648001532/posts?rel=author), **[Facebook](https://www.facebook.com/journaldev)** or [**Twitter**](https://twitter.com/JournalDev). I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **[Youtube](https://www.youtube.com/user/JournalDev)**.

[« Shell Script to change extension of multiple files in Unix](http://www.journaldev.com/118/shell-script-to-change-extension-of-multiple-files-in-unix)

[How to Create immutable Class in java? »](http://www.journaldev.com/129/how-to-create-immutable-class-in-java)

**Comments**

1. **Vikas Kumar says**

[JULY 12, 2017 AT 10:30 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-38627)

Hi ,

I have checked this example in our eclipse and it was working in HashMap class too .

public class ConcurrentHashMapExample {

/\*\*  
\* @param args  
\*/  
public static void main(String[] args) {  
// TODO Auto-generated method stub  
try{  
HashMap map = new HashMap();

map.put(“One”, “I am one”);  
map.put(“Two”,”I am two”);  
System.out.println(“Map is returned in first”+map);

Iterator itr= map.keySet().iterator();

while(itr.hasNext()){  
String key=itr.next();

System.out.println(“Traversing value”+key);  
if(“One”.equals(key))  
map.put(key+”OneOne”, “I am Eleven”);

System.out.println(“Map is returned in last”+map);  
}  
}  
catch(Exception e){

e.printStackTrace();  
}

Output is :

Map is returned in first{Two=I am two, One=I am one}  
Traversing valueTwo  
Map is returned in last{Two=I am two, One=I am one}  
Traversing valueOne  
Map is returned in last{OneOneOne=I am Eleven, Two=I am two, One=I am one}

In case of remove the element from hashmap ,It will showing the exception .Kindly correct the logic here

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-38627)

1. **Tejas says**

[APRIL 12, 2017 AT 10:18 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-37906)

Map hm= new ConcurrentHashMap();  
hm.put(“1”,1);  
hm.put(“2”,2);  
hm.put(“3”,3);  
System.out.println(hm);  
Iterator it = hm.keySet().iterator();

while(it.hasNext()){  
String key = it.next();  
if(key.equals(“3”))  
hm.put(“3”,4);  
}  
System.out.println(hm);  
System.out.println(“+++++++++++++++++++++++++++++++++++”);  
hm= new HashMap();  
hm.put(“1”,1);  
hm.put(“2”,2);  
hm.put(“3”,3);  
System.out.println(hm);  
Iterator it1 = hm.keySet().iterator();

while(it.hasNext()){  
String key = it1.next();  
if(key.equals(“3”))  
hm.put(“3”,4);  
}  
System.out.println(hm);

O/P is  
//concurrent hash map  
{1=1, 2=2, 3=3}  
{1=1, 2=2, 3=4}  
+++++++++++++++++++++++++++++++++++  
//hashmap  
{1=1, 2=2, 3=3}  
{1=1, 2=2, 3=3}

\*Note :- I am using java 8 . is cocncurrent HashpMap example is handeled

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-37906)

1. **narayana says**

[FEBRUARY 9, 2017 AT 7:58 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-37446)

Excellent example

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-37446)

1. **Rupa says**

[OCTOBER 14, 2016 AT 1:38 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-36576)

Great example ,nice explaination,Thanks for sharing.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-36576)

1. **Anil Gupta From Sidhi Mp says**

[OCTOBER 22, 2015 AT 12:03 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-33412)

really awesome

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-33412)

* + **Anil Gupta From Sidhi Mp says**

[OCTOBER 22, 2015 AT 1:17 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-33413)

thanks for giving this type of deeply knowledge ,

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-33413)

1. **mr\_rookie says**

[JUNE 18, 2015 AT 8:39 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32650)

every time when get a iterator .the operation underlying is :  
public Iterator iterator() {  
return new Itr();  
}  
private class Itr implements Iterator {

int expectedModCount = modCount;  
……  
}

then every mutative operation , thr modcount will be modCount++;  
that’s why when modCount changed ,but expectedModCount was initialized when iterator(),and not changed after.  
if (modCount != expectedModCount)  
throw new ConcurrentModificationException();

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32650)

1. **husenbadshah goundi says**

[MAY 29, 2015 AT 6:22 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32550)

greate example …..superb

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32550)

1. **Chandrasekhar G says**

[MAY 11, 2015 AT 4:06 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32413)

Really this tutorial is awesome.

Thanks for sharing this.

Regards,  
Chandu.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32413)

1. **Ram says**

[APRIL 23, 2015 AT 4:27 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32278)

Excellent example about concurenthashmap and concurentmodified exception

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32278)

1. **NARAYANA says**

[JANUARY 27, 2015 AT 2:04 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-31553)

Really helpfull,while attending the inrerviews…It is good !

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-31553)

1. **sunil says**

[JANUARY 7, 2015 AT 3:43 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-31349)

Its really good explanation……Thanks…

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-31349)

1. **Ramakrisha says**

[NOVEMBER 5, 2014 AT 12:04 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-30673)

HI

Very good explantion,its very helpfull to me.

Thanks

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-30673)

1. **nawaz says**

[AUGUST 28, 2014 AT 11:07 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29940)

i had gone through so many web sites, but no one has given clear definition, but finally i got complete clarity on this point. thanks brother.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29940)

1. **Ganesh Rashinker says**

[AUGUST 10, 2014 AT 11:53 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29769)

The above example is good. I have tried to explain the same using multiple threads trying to add data to HashMap.

<http://www.javavisited.com/2014/04/internal-working-of-hashmap_15.html>

and

<http://www.javavisited.com/2014/04/internal-working-of-concurrenthashmap.html>

Please check this website for a demo of what I have mentioned above. Please correct me if my understanding is wrong somewhere.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29769)

1. **nikhil says**

[MAY 26, 2014 AT 5:36 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29002)

if u are breaking from while when key is 3 then how could it print the remaining values in while loop

if(key.equals(“3″)){  
myMap.put(key+”new”, “new3”);  
break;  
}

ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}  
ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 4=1, 2=1}  
HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}  
HashMap after iterator: {3=1, 2=1, 1=1, 3new=new3, 6=1, 5=1, 4=1}

The output should be

ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}  
ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 4=1, 2=1}  
HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}  
HashMap after iterator: {3=1, 2=1, 1=1, 3new=new3}

Please clarify

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-29002)

* + **prakash says**

[MARCH 27, 2015 AT 1:55 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32064)

If you observe carefully he is not displaying value in while loop…  
print is last statement….

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32064)

1. **Ankit Tripathi says**

[APRIL 14, 2014 AT 10:58 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28580)

Nicely compiled article. Understood the concept quickly.  
Thank you.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28580)

1. **Binh Thanh Nguyen says**

[APRIL 2, 2014 AT 9:15 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28411)

Thanks, nice post

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28411)

1. **Devendra says**

[MARCH 3, 2014 AT 4:13 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28223)

Nicely explained. can you please tell why there is no exception when we are try to change the value but not key.

//myMap.put(key+”new”, “new3”);  
myMap.put(key, “new3”);

do we have seperate “modCount” for key and value in map.

but how it is getting a updated value?

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28223)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[MARCH 3, 2014 AT 9:22 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28225)

modCount is for key only, since we are not changing the structure of the map the exception is not thrown.

You are seeing updates value because we are printing the HashMap and not the iterator values.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28225)

* + - [**arif khan**](http://www.journaldev.com/)**says**

[JUNE 1, 2015 AT 12:38 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32566)

This is the nice tutorial..100/100

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-32566)

1. **Nazeer Khan says**

[DECEMBER 17, 2013 AT 5:55 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-27017)

For the key “4” :  
while(it1.hasNext()){  
String key = it1.next();  
if(key.equals(“4″)){  
myMap.put(key+”new”, “new4”);  
}  
}

Output:  
Got no ConcurrentModificationException occured…..  
HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}  
HashMap after iterator: {3=1, 2=1, 1=1, 4new=new4, 6=1, 5=1, 4=1}

How???

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-27017)

* + [**Pankaj Chopra**](http://suresolution.blogspot.com/)**says**

[FEBRUARY 23, 2014 AT 7:44 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28180)

Hi, I am getting the same issue. We are getting error with 3, but not with 4.  
Could you please explain? Why?

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28180)

* + - [**Pankaj Chopra**](http://suresolution.blogspot.com/)**says**

[FEBRUARY 23, 2014 AT 7:55 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28181)

I think I got the answer. The issue here is, the key=”4″ is in the last in Hash Map  
HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}

So, after comparing with “4”, and adding a new key-Value pair “4new”=”new4”, its done with the processing of iterator. As the error is thrown in next call to iterator.next(), so in this case, we are not getting any error. You can check in all other cases you will get the error.

Please correct me, if my understanding is incorrect.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28181)

* + - * **Abhishek Sunandan says**

[MARCH 28, 2014 AT 2:07 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28370)

We can have a better understanding if we Use Integer as keys

//HashMap  
Map myMap1 = new HashMap();  
myMap1.put(1, “1”);  
myMap1.put(2, “1”);  
myMap1.put(3, “1”);  
myMap1.put(4, “1”);  
myMap1.put(5, “1”);  
myMap1.put(6, “1”);  
System.out.println(“HashMap before iterator: “+myMap1);  
Iterator it1 = myMap1.keySet().iterator();

while(it1.hasNext()){  
Integer key = it1.next();  
if(key.equals(6)) {myMap1.put(7, “new3”);

}  
}  
System.out.println(“HashMap after iterator: “+myMap1);

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28370)

* + - * + **raj says**

[DECEMBER 17, 2016 AT 4:01 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-37070)

awesome

1. **Sravanthi says**

[AUGUST 5, 2013 AT 7:39 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21609)

I have added code to remove one item from map. the item is removed from map. But when its iterating it is printing the key and with value as null. Does that mean iterator has that key even though map does not have it? Does iterator will have a copy of keys?  
Below is modified code:  
while(it.hasNext()){  
String key = it.next();  
System.out.println(“Key “+key+”value: “+myMap.get(key));  
if(key.equals(“3″)) {  
myMap.put(key+”new”, “new3”);  
myMap.remove(“4”);  
System.out.println(“while iterating: “+myMap);  
}  
}  
System.out.println(“ConcurrentHashMap after iterator: “+myMap);

ouput is:  
ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}  
Key 1 value: 1  
Key 5 value: 1  
Key 6 value: 1  
Key 3 value: 1  
while iterating: {1=1, 3new=new3, 5=1, 6=1, 3=1, 2=1}  
Key 4 value: null  
Key 2 value: 1  
ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 2=1}

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21609)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[AUGUST 5, 2013 AT 8:27 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21617)

This is the expected behavior, when you get the iterator from Map then it has all the keys including “4”, now when you remove it, its removed from the HashMap but not from the iterator that you have already got, so its returning NULL.

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21617)

* + - **Sravanthi says**

[AUGUST 5, 2013 AT 9:47 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21623)

Thanks for the quick answer.  
So Iterator has separate memory with key copies?

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-21623)

* + - **Ramana Reddy says**

[MARCH 5, 2014 AT 10:28 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28243)

i have one doubt in this ….  
when ever we remove one value it is removed from HashMap  
but not Iteratior k fine…  
but we are printing only HashMap object on console  
even though why that Null values came for removed object…………………………..

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-28243)

1. **Rajeev says**

[DECEMBER 1, 2012 AT 4:29 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-6636)

Excellent! Clears the concept and the usage of ConcurrentHashMap Thanks

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-6636)

1. **Ashok singh says**

[MAY 21, 2012 AT 9:29 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-896)

defnetly we get confused about Output. Can you Explain more reagarding the above Program…When we adding the element first time it provides an Exception..But When you have Written “Map.put(key, “new3″);” this line it doesnt provide any error.Explain it in better way….

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-896)

* + **Pankaj says**

[OCTOBER 17, 2012 AT 6:57 PM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-941)

If you use myMap.put(key, “new3”); its actually modifying key value, so there is no **structural modification** in the HashMap.  
Output will be:  
ConcurrentHashMap before iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}  
ConcurrentHashMap after iterator: {1=1, 3new=new3, 5=1, 6=1, 3=1, 4=1, 2=1}  
HashMap before iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}  
HashMap after iterator: {3=new3, 2=1, 1=1, 6=1, 5=1, 4=1}

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-941)

1. **Khalid says**

[DECEMBER 15, 2011 AT 11:54 AM](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-882)

Odd behaviour, the result of the last modification comes out as

HashMap after iterator: {3=new3, 2=1, 1=1, 6=1, 5=1, 4=1}

[Reply](http://www.journaldev.com/122/java-concurrenthashmap-example-iterator#comment-882)