**CopyOnWriteArrayList in Java**

AUGUST 1, 2016 BY [PANKAJ](http://www.journaldev.com/author/pankaj) [21 COMMENTS](http://www.journaldev.com/1289/copyonwritearraylist-java#comments)

CopyOnWriteArrayList in Java is a thread safe implementation of List interface. CopyOnWriteArrayList was added in Java 1.5 and part of Collections framework.

**Java ArrayList and ConcurrentModificationException**

ArrayList is one of the basic implementations of List interface and it’s part of **Java Collections Framework**. We can use **iterator** to traverse through ArrayList elements.

Let’s check a sample program of ArrayList.

ConcurrentListExample.java

package com.journaldev.collections;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.concurrent.CopyOnWriteArrayList;

public class ConcurrentListExample {

public static void main(String[] args) {

List<String> list = new ArrayList<>();

list.add("1");

list.add("2");

list.add("3");

list.add("4");

list.add("5");

// get the iterator

Iterator<String> it = list.iterator();

//manipulate list while iterating

while(it.hasNext()){

System.out.println("list is:"+list);

String str = it.next();

System.out.println(str);

if(str.equals("2"))list.remove("5");

if(str.equals("3"))list.add("3 found");

//below code don't throw ConcurrentModificationException

//because it doesn't change modCount variable of list

if(str.equals("4")) list.set(1, "4");

}

}

}

When we run above program, we get [java.util.ConcurrentModificationException](http://www.journaldev.com/378/java-util-concurrentmodificationexception) as soon as the ArrayList is modified.

It happens because ArrayList iterator is **fail-fast** by design. What it means is that once the iterator is created, if the ArrayList is modified, it throws **ConcurrentModificationException**.

If you check the console log, you will notice that exception is thrown by Iterator next() method. If you will look into the ArrayList source code, following method is called everytime we invoke next() on iterator that throws exception.

final void checkForComodification() {

if (modCount != expectedModCount)

throw new ConcurrentModificationException();

}

Here modCount is the ArrayList variable that holds the modification count and every time we use add, remove or trimToSize method, it increments. expectedModCount is the iterator variable that is initialized when we create iterator with same value as modCount. This explains why we don’t get exception if we use set method to replace any existing element.

So basically iterator throws ConcurrentModificationException if list size is changed.

**CopyOnWriteArrayList in Java**

[](http://cdn.journaldev.com/wp-content/uploads/2013/01/copyonwritearraylist-java.jpg)

Sometimes we want to add or remove elements from the list if we find some specific element, in that case we should use concurrent collection class – CopyOnWriteArrayList. This is a thread-safe variant of java.util.ArrayList in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

**CopyOnWriteArrayList** introduces extra overload to the processing but it’s very effective when number of modifications are minimal compared to number of traversal operations.

If we change the implementation to CopyOnWriteArrayList, then we don’t get any exception and below is the output produced.

list is:[1, 2, 3, 4, 5]

1

list is:[1, 2, 3, 4, 5]

2

list is:[1, 2, 3, 4]

3

list is:[1, 2, 3, 4, 3 found]

4

list is:[1, 4, 3, 4, 3 found]

5

Notice that it allows the modification of list, but it doesn’t change the iterator and we get same elements as it was on original list.

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)**.

[« Java Variable Arguments or Java Varargs Explained](http://www.journaldev.com/1257/java-variable-arguments-or-java-varargs-explained)

[Collections in Java – Tutorial »](http://www.journaldev.com/1260/collections-in-java-tutorial)

**Comments**

1. **Vikas Kumar says**

[JULY 12, 2017 AT 11:35 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-38629)

Hi Pankaj ,

As I checked the concept for concurrentmodification exception for ArrayList Object .

If we perform remove operation during Iteration ,It is working and throwing any exception .

And In case of add element in ArrayList during Iteration .It is throwing

Exception in thread “main” java.util.ConcurrentModificationException  
at java.util.AbstractList$Itr.checkForComodification(Unknown Source)  
at java.util.AbstractList$Itr.next(Unknown Source)  
at com.collections.ArrayListExample.main(ArrayListExample.java:29)

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-38629)

1. [**infoj**](http://netjs.blogspot.com/2016/01/copyonwritearraylist-in-java.html)**says**

[MARCH 6, 2016 AT 8:07 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-34237)

One diff is that underlying array in ArrayList is not declared volatile where as in [CopyOnWriteArrayList](http://netjs.blogspot.com/2016/01/copyonwritearraylist-in-java.html) underlying object array is declared volatile.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-34237)

1. **Pankaj Roy says**

[SEPTEMBER 9, 2015 AT 3:52 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-33118)

Am not getting concurrent modification exception and am able to change the element and remove the same from the map/concurrentHashMap while inside Iterator?? Can you have a look?

public class ConcurrentHM {  
public static void main(String[] args) {  
Map map= new HashMap();  
map.put(“1”, “1”);  
map.put(“2”, “1”);  
map.put(“3”, “1”);  
map.put(“4”, “1”);  
map.put(“5”, “1”);  
map.put(“6”, “1”);  
Iterator it = map.keySet().iterator();  
System.out.println(“HashMap befor iterator: “+map);  
while(it.hasNext()){  
String key = it.next();  
if(key.equals(“4”))  
map.remove(key);  
}  
System.out.println(“HashMap after iterator: “+map);

map= new ConcurrentHashMap();  
map.put(“1”, “1”);  
map.put(“2”, “1”);  
map.put(“3”, “1”);  
map.put(“4”, “1”);  
map.put(“5”, “1”);  
map.put(“6”, “1”);  
Iterator it1 = map.keySet().iterator();  
System.out.println(“ConcurremntHashMap befor iterator: “+map);  
while(it1.hasNext()){  
String key = it1.next();  
if(key.equals(“4”)){  
map.remove(key);  
}

}  
System.out.println(“ConcurremntHashMap after iterator: “+map);  
}

}

Output :  
HashMap befor iterator: {3=1, 2=1, 1=1, 6=1, 5=1, 4=1}  
HashMap after iterator: {3=1, 2=1, 1=1, 6=1, 5=1}  
ConcurremntHashMap befor iterator: {1=1, 5=1, 6=1, 3=1, 4=1, 2=1}  
ConcurremntHashMap after iterator: {1=1, 5=1, 6=1, 3=1, 2=1}

Thanks

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-33118)

* + **Srinivas vootla says**

[SEPTEMBER 21, 2016 AT 1:18 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-36364)

there could be chance because what ever the key you are removing from map is the last element ,so there is chance get call map.next()

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-36364)

1. **naresh says**

[JUNE 14, 2015 AT 7:57 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-32627)

Hai Pankaj

I cannot understand internal implementaion of below code

final void checkForComodification() {  
if (modCount != expectedModCount)  
throw new ConcurrentModificationException();  
}  
As you said modCount is the ArrayList variable that holds the modification count and every time we use add, remove or trimToSize method, it increments. expectedModCount is the iterator variable that is initialized when we create iterator with same value as modCount.

My Question is what is the initial value of at first time of modcount and expectedModcounttime and at when level the exception is thrown. i cannot understand can u please explain briefly.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-32627)

1. [**xinwendashibaike**](http://xinwendashibaike.github.io/)**says**

[MARCH 26, 2015 AT 11:40 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-32063)

List list = new ArrayList();  
list.add(“1”);  
list.add(“2”);

Iterator it2=list.iterator();  
while(it2.hasNext()){  
String hh=(String)it2.next();  
if(hh.equals(“1”)){  
list.remove(0);  
}  
}  
System.out.println(list);

after list.remove(0) size of list is 1,so loop will stop

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-32063)

1. **Shambhu Shah says**

[APRIL 4, 2014 AT 10:25 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28450)

Nice explained..well done pankaj.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28450)

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

[APRIL 4, 2014 AT 5:58 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28456)

Thanks Sambhu

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28456)

1. **Bawse says**

[MARCH 28, 2014 AT 1:14 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28368)

Shouldn’t you pass “5” into the arraylist constructor to avoid java defaulting creating arraylist of size 10?

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28368)

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

[MARCH 28, 2014 AT 2:51 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28371)

Yes we can provide that, but it’s an example so I prefer to keep it simple. In real life scenarios, where we want optimization and performance specifying list capacity is beneficial.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-28371)

1. **Anoj Singh says**

[SEPTEMBER 17, 2013 AT 3:33 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23513)

Hi,

I want to know if employee class same example instead of String class how can get exception  
ConcurrentModificationException.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23513)

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

[SEPTEMBER 17, 2013 AT 5:44 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23516)

change the object type in the list and add some values while iterating.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23516)

* + - **Anoj Singh says**

[SEPTEMBER 20, 2013 AT 11:56 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23619)

public class Test {  
public static void main(String[] args) {  
List al=new ArrayList();  
Employee ee=new Employee(1,”anoj”);  
Employee ee1=new Employee(2,”hai”);  
al.add(ee);  
al.add(ee1);  
Iterator it=al.iterator();  
while(it.hasNext()){  
Employee hh=(Employee)it.next();  
if(hh.getName().equals(“anoj”)){  
al.remove(0);  
System.out.println(al);  
}

}  
}}

but i didn’t get ConcurrentModificationException.

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-23619)

* + - * **sasidhar says**

[OCTOBER 8, 2013 AT 8:38 PM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-24244)

u dint get error for this bcoz,  
ur deleting 1 object from content of 2 objects thats y there is no more objects to read 2 nd time in hasNext()..  
try to add another object after removing object..  
u ll get error..

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-24244)

* + - * + **ANEK SINGH says**

[FEBRUARY 28, 2016 AT 3:04 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-34193)

Explanation given is wrong …  
package com.Ikeacommon.generics;

import java.util.ArrayList;  
import java.util.\*;  
import java.io.\*;

public class ConcurrentModification {  
public static void main(String[] args) {

List l = new ArrayList();

l.add(12);  
l.add(13);  
l.add(15);  
l.add(130);  
l.add(483);  
Iterator it=l.iterator();

System.out.println(“before remove method :”+l);  
while (it.hasNext()){

int i=it.next();

if(i==130){l.remove(3);}  
System.out.println(i);  
//it.remove();

}  
System.out.println(“After remove method:”+l);

}}  
See here I am removing the 4th element ..after loop continues and read 5th element ..still no Concurrent Modification Exception….

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

[FEBRUARY 28, 2016 AT 9:16 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-34194)

Do following:  
use generics  
think what’s happening in l.remove(3) statement  
print the list size after the while loop. You should have your answer by this time…

* + - **Rani Maddala says**

[JUNE 18, 2017 AT 4:01 AM](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-38406)

HI ANEK SINGH ,

Instead of if(i==130){l.remove(3);} , if I have l.remove(i) , then it’s throwing an exception.

But I didn;t understand why it’s not throwing an exception if we gave index.

Regards  
Rani

[Reply](http://www.journaldev.com/1289/copyonwritearraylist-java#comment-38406)