**CopyOnWriteArrayList in Java**

**CopyOnWriteArrayList class** is introduced in JDK 1.5, which implements the [**List interface**](https://www.geeksforgeeks.org/list-interface-java-examples/). It is an enhanced version of **[ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/)**in which all modifications (add, set, remove, etc) are implemented by making a fresh copy. It is found in **java.util.concurrent** package. It is a data structure created to be used in a concurrent environment.

**Here are few points about CopyOnWriteArrayList:**

* As the name indicates, CopyOnWriteArrayList creates a Cloned copy of underlying ArrayList, for every update operation at a certain point both will be synchronized automatically, which is taken care of by JVM. Therefore, there is no effect for threads that are performing read operation.
* It is costly to use because for every update operation a cloned copy will be created. Hence, CopyOnWriteArrayList is the best choice if our frequent operation is read operation.
* The underlined data structure is a grow-able array.
* It is a thread-safe version of ArrayList.
* Insertion is preserved, duplicates, null, and heterogeneous Objects are allowed.
* The main important point about CopyOnWriteArrayList is the [Iterator](https://www.geeksforgeeks.org/iterators-in-java/) of CopyOnWriteArrayList can not perform remove operation otherwise we get Run-time exception saying **UnsupportedOperationException.**add() and set() methods on CopyOnWriteArrayList iterator also throws **UnsupportedOperationException.**Also Iterator of CopyOnWriteArrayList will never throw **ConcurrentModificationException**.

// Java program to illustrate

// CopyOnWriteArrayList class

import java.util.\*;

import java.util.concurrent.CopyOnWriteArrayList;

public class ConcurrentDemo extends Thread {

static CopyOnWriteArrayList<String> l

= new CopyOnWriteArrayList<String>();

public void run()

{

// Child thread trying to

// add new element in the

// Collection object

l.add("D");

}

public static void main(String[] args)

throws InterruptedException

{

l.add("A");

l.add("B");

l.add("c");

// We create a child thread

// that is going to modify

// ArrayList l.

ConcurrentDemo t = new ConcurrentDemo();

t.start();

Thread.sleep(1000);

// Now we iterate through

// the ArrayList and get

// exception.

Iterator itr = l.iterator();

while (itr.hasNext()) {

String s = (String)itr.next();

System.out.println(s);

Thread.sleep(1000);

}

System.out.println(l);

}

}

**Iterating over CopyOnWriteArrayList:**We can iterate over CopyOnWriteArrayList using [iterator()](https://www.geeksforgeeks.org/copyonwritearraylist-iterator-method-in-java/) method. The important point to be noted is that the iterator we create is an immutable snapshot of the original list. Because of this property, we can see that **GfG**is not printed at the first iteration.

// Java program to illustrate

// CopyOnWriteArrayList class

import java.io.\*;

import java.util.\*;

import java.util.concurrent.\*;

class Demo {

public static void main(String[] args)

{

CopyOnWriteArrayList<String> list

= new CopyOnWriteArrayList<>();

// Initial Iterator

Iterator itr = list.iterator();

list.add("GfG");

System.out.println("List contains: ");

while (itr.hasNext())

System.out.println(itr.next());

// iterator after adding an element

itr = list.iterator();

System.out.println("List contains:");

while (itr.hasNext())

System.out.println(itr.next());

}

}