| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CopyOnWriteArrayList.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/ConcurrentSkipListSet.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/CopyOnWriteArraySet.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/CopyOnWriteArrayList.html)    [**NO FRAMES**](http://docs.google.com/CopyOnWriteArrayList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

## **java.util.concurrent**

Class CopyOnWriteArrayList<E>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.CopyOnWriteArrayList<E>**

**Type Parameters:**E - the type of elements held in this collection **All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Collection](http://docs.google.com/java/util/Collection.html)<E>, [List](http://docs.google.com/java/util/List.html)<E>, [RandomAccess](http://docs.google.com/java/util/RandomAccess.html)

public class **CopyOnWriteArrayList<E>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [List](http://docs.google.com/java/util/List.html)<E>, [RandomAccess](http://docs.google.com/java/util/RandomAccess.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

A thread-safe variant of [ArrayList](http://docs.google.com/java/util/ArrayList.html) in which all mutative operations (add, set, and so on) are implemented by making a fresh copy of the underlying array.

This is ordinarily too costly, but may be *more* efficient than alternatives when traversal operations vastly outnumber mutations, and is useful when you cannot or don't want to synchronize traversals, yet need to preclude interference among concurrent threads. The "snapshot" style iterator method uses a reference to the state of the array at the point that the iterator was created. This array never changes during the lifetime of the iterator, so interference is impossible and the iterator is guaranteed not to throw ConcurrentModificationException. The iterator will not reflect additions, removals, or changes to the list since the iterator was created. Element-changing operations on iterators themselves (remove, set, and add) are not supported. These methods throw UnsupportedOperationException.

All elements are permitted, including null.

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a CopyOnWriteArrayList [*happen-before*](http://docs.google.com/package-summary.html#MemoryVisibility) actions subsequent to the access or removal of that element from the CopyOnWriteArrayList in another thread.

This class is a member of the  [Java Collections Framework](http://docs.google.com/technotes/guides/collections/index.html).

**Since:** 1.5 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.util.concurrent.CopyOnWriteArrayList)

| **Constructor Summary** | |
| --- | --- |
| [**CopyOnWriteArrayList**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#CopyOnWriteArrayList())()            Creates an empty list. |
| [**CopyOnWriteArrayList**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#CopyOnWriteArrayList(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)            Creates a list containing the elements of the specified collection, in the order they are returned by the collection's iterator. |
| [**CopyOnWriteArrayList**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#CopyOnWriteArrayList(E%5B%5D))([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)[] toCopyIn)            Creates a list holding a copy of the given array. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#add(E))([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e)            Appends the specified element to the end of this list. |
| void | [**add**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#add(int,%20E))(int index, [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) element)            Inserts the specified element at the specified position in this list. |
| boolean | [**addAll**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#addAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)            Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. |
| boolean | [**addAll**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#addAll(int,%20java.util.Collection))(int index, [Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)            Inserts all of the elements in the specified collection into this list, starting at the specified position. |
| int | [**addAllAbsent**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#addAllAbsent(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)            Appends all of the elements in the specified collection that are not already contained in this list, to the end of this list, in the order that they are returned by the specified collection's iterator. |
| boolean | [**addIfAbsent**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#addIfAbsent(E))([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e)            Append the element if not present. |
| void | [**clear**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#clear())()            Removes all of the elements from this list. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#clone())()            Returns a shallow copy of this list. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this list contains the specified element. |
| boolean | [**containsAll**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#containsAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Returns true if this list contains all of the elements of the specified collection. |
| boolean | [**equals**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Compares the specified object with this list for equality. |
| [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) | [**get**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#get(int))(int index)            Returns the element at the specified position in this list. |
| int | [**hashCode**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#hashCode())()            Returns the hash code value for this list. |
| int | [**indexOf**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#indexOf(E,%20int))([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e, int index)            Returns the index of the first occurrence of the specified element in this list, searching forwards from index, or returns -1 if the element is not found. |
| int | [**indexOf**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#indexOf(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#isEmpty())()            Returns true if this list contains no elements. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#iterator())()            Returns an iterator over the elements in this list in proper sequence. |
| int | [**lastIndexOf**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#lastIndexOf(E,%20int))([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e, int index)            Returns the index of the last occurrence of the specified element in this list, searching backwards from index, or returns -1 if the element is not found. |
| int | [**lastIndexOf**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#lastIndexOf(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. |
| [ListIterator](http://docs.google.com/java/util/ListIterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> | [**listIterator**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#listIterator())()            Returns a list iterator over the elements in this list (in proper sequence). |
| [ListIterator](http://docs.google.com/java/util/ListIterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> | [**listIterator**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#listIterator(int))(int index)            Returns a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. |
| [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) | [**remove**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#remove(int))(int index)            Removes the element at the specified position in this list. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the first occurrence of the specified element from this list, if it is present. |
| boolean | [**removeAll**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#removeAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Removes from this list all of its elements that are contained in the specified collection. |
| boolean | [**retainAll**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#retainAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Retains only the elements in this list that are contained in the specified collection. |
| [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) | [**set**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#set(int,%20E))(int index, [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) element)            Replaces the element at the specified position in this list with the specified element. |
| int | [**size**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#size())()            Returns the number of elements in this list. |
| [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> | [**subList**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#subList(int,%20int))(int fromIndex, int toIndex)            Returns a view of the portion of this list between fromIndex, inclusive, and toIndex, exclusive. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#toArray())()            Returns an array containing all of the elements in this list in proper sequence (from first to last element). |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#toString())()            Returns a string representation of this list. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### CopyOnWriteArrayList

public **CopyOnWriteArrayList**()

Creates an empty list.

### CopyOnWriteArrayList

public **CopyOnWriteArrayList**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)

Creates a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

**Parameters:**c - the collection of initially held elements **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null

### CopyOnWriteArrayList

public **CopyOnWriteArrayList**([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)[] toCopyIn)

Creates a list holding a copy of the given array.

**Parameters:**toCopyIn - the array (a copy of this array is used as the internal array) **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

| **Method Detail** |
| --- |

### size

public int **size**()

Returns the number of elements in this list.

**Specified by:**[size](http://docs.google.com/java/util/Collection.html#size()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[size](http://docs.google.com/java/util/List.html#size()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Returns:**the number of elements in this list

### isEmpty

public boolean **isEmpty**()

Returns true if this list contains no elements.

**Specified by:**[isEmpty](http://docs.google.com/java/util/Collection.html#isEmpty()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[isEmpty](http://docs.google.com/java/util/List.html#isEmpty()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Returns:**true if this list contains no elements

### contains

public boolean **contains**([Object](http://docs.google.com/java/lang/Object.html) o)

Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that (o==null ? e==null : o.equals(e)).

**Specified by:**[contains](http://docs.google.com/java/util/Collection.html#contains(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[contains](http://docs.google.com/java/util/List.html#contains(java.lang.Object)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**o - element whose presence in this list is to be tested **Returns:**true if this list contains the specified element

### indexOf

public int **indexOf**([Object](http://docs.google.com/java/lang/Object.html) o)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the lowest index i such that (o==null ? get(i)==null : o.equals(get(i))), or -1 if there is no such index.

**Specified by:**[indexOf](http://docs.google.com/java/util/List.html#indexOf(java.lang.Object)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**o - element to search for **Returns:**the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element

### indexOf

public int **indexOf**([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e,  
 int index)

Returns the index of the first occurrence of the specified element in this list, searching forwards from index, or returns -1 if the element is not found. More formally, returns the lowest index i such that (i >= index && (e==null ? get(i)==null : e.equals(get(i)))), or -1 if there is no such index.

**Parameters:**e - element to search forindex - index to start searching from **Returns:**the index of the first occurrence of the element in this list at position index or later in the list; -1 if the element is not found. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative

### lastIndexOf

public int **lastIndexOf**([Object](http://docs.google.com/java/lang/Object.html) o)

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the highest index i such that (o==null ? get(i)==null : o.equals(get(i))), or -1 if there is no such index.

**Specified by:**[lastIndexOf](http://docs.google.com/java/util/List.html#lastIndexOf(java.lang.Object)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**o - element to search for **Returns:**the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element

### lastIndexOf

public int **lastIndexOf**([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e,  
 int index)

Returns the index of the last occurrence of the specified element in this list, searching backwards from index, or returns -1 if the element is not found. More formally, returns the highest index i such that (i <= index && (e==null ? get(i)==null : e.equals(get(i)))), or -1 if there is no such index.

**Parameters:**e - element to search forindex - index to start searching backwards from **Returns:**the index of the last occurrence of the element at position less than or equal to index in this list; -1 if the element is not found. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is greater than or equal to the current size of this list

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

Returns a shallow copy of this list. (The elements themselves are not copied.)

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a clone of this list**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### toArray

public [Object](http://docs.google.com/java/lang/Object.html)[] **toArray**()

Returns an array containing all of the elements in this list in proper sequence (from first to last element).

The returned array will be "safe" in that no references to it are maintained by this list. (In other words, this method must allocate a new array). The caller is thus free to modify the returned array.

This method acts as bridge between array-based and collection-based APIs.

**Specified by:**[toArray](http://docs.google.com/java/util/Collection.html#toArray()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[toArray](http://docs.google.com/java/util/List.html#toArray()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Returns:**an array containing all the elements in this list**See Also:**[Arrays.asList(Object[])](http://docs.google.com/java/util/Arrays.html#asList(T...))

### toArray

public <T> T[] **toArray**(T[] a)

Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. If the list fits in the specified array, it is returned therein. Otherwise, a new array is allocated with the runtime type of the specified array and the size of this list.

If this list fits in the specified array with room to spare (i.e., the array has more elements than this list), the element in the array immediately following the end of the list is set to null. (This is useful in determining the length of this list *only* if the caller knows that this list does not contain any null elements.)

Like the [toArray()](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#toArray()) method, this method acts as bridge between array-based and collection-based APIs. Further, this method allows precise control over the runtime type of the output array, and may, under certain circumstances, be used to save allocation costs.

Suppose x is a list known to contain only strings. The following code can be used to dump the list into a newly allocated array of String:

String[] y = x.toArray(new String[0]);

Note that toArray(new Object[0]) is identical in function to toArray().

**Specified by:**[toArray](http://docs.google.com/java/util/Collection.html#toArray(T%5B%5D)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[toArray](http://docs.google.com/java/util/List.html#toArray(T%5B%5D)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**a - the array into which the elements of the list are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose. **Returns:**an array containing all the elements in this list **Throws:** [ArrayStoreException](http://docs.google.com/java/lang/ArrayStoreException.html) - if the runtime type of the specified array is not a supertype of the runtime type of every element in this list [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### get

public [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) **get**(int index)

Returns the element at the specified position in this list.

**Specified by:**[get](http://docs.google.com/java/util/List.html#get(int)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - index of the element to return **Returns:**the element at the specified position in this list **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index >= size())

### set

public [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) **set**(int index,  
 [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) element)

Replaces the element at the specified position in this list with the specified element.

**Specified by:**[set](http://docs.google.com/java/util/List.html#set(int,%20E)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - index of the element to replaceelement - element to be stored at the specified position **Returns:**the element previously at the specified position **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index >= size())

### add

public boolean **add**([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e)

Appends the specified element to the end of this list.

**Specified by:**[add](http://docs.google.com/java/util/Collection.html#add(E)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[add](http://docs.google.com/java/util/List.html#add(E)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**e - element to be appended to this list **Returns:**true (as specified by [Collection.add(E)](http://docs.google.com/java/util/Collection.html#add(E)))

### add

public void **add**(int index,  
 [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) element)

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

**Specified by:**[add](http://docs.google.com/java/util/List.html#add(int,%20E)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - index at which the specified element is to be insertedelement - element to be inserted **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index > size())

### remove

public [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) **remove**(int index)

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

**Specified by:**[remove](http://docs.google.com/java/util/List.html#remove(int)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - the index of the element to be removed **Returns:**the element previously at the specified position **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index >= size())

### remove

public boolean **remove**([Object](http://docs.google.com/java/lang/Object.html) o)

Removes the first occurrence of the specified element from this list, if it is present. If this list does not contain the element, it is unchanged. More formally, removes the element with the lowest index i such that (o==null ? get(i)==null : o.equals(get(i))) (if such an element exists). Returns true if this list contained the specified element (or equivalently, if this list changed as a result of the call).

**Specified by:**[remove](http://docs.google.com/java/util/Collection.html#remove(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[remove](http://docs.google.com/java/util/List.html#remove(java.lang.Object)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**o - element to be removed from this list, if present **Returns:**true if this list contained the specified element

### addIfAbsent

public boolean **addIfAbsent**([E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html) e)

Append the element if not present.

**Parameters:**e - element to be added to this list, if absent **Returns:**true if the element was added

### containsAll

public boolean **containsAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Returns true if this list contains all of the elements of the specified collection.

**Specified by:**[containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[containsAll](http://docs.google.com/java/util/List.html#containsAll(java.util.Collection)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**c - collection to be checked for containment in this list **Returns:**true if this list contains all of the elements of the specified collection **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null**See Also:**[contains(Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#contains(java.lang.Object))

### removeAll

public boolean **removeAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Removes from this list all of its elements that are contained in the specified collection. This is a particularly expensive operation in this class because of the need for an internal temporary array.

**Specified by:**[removeAll](http://docs.google.com/java/util/Collection.html#removeAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[removeAll](http://docs.google.com/java/util/List.html#removeAll(java.util.Collection)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**c - collection containing elements to be removed from this list **Returns:**true if this list changed as a result of the call **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this list is incompatible with the specified collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if this list contains a null element and the specified collection does not permit null elements (optional), or if the specified collection is null**See Also:**[remove(Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#remove(java.lang.Object))

### retainAll

public boolean **retainAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Retains only the elements in this list that are contained in the specified collection. In other words, removes from this list all of its elements that are not contained in the specified collection.

**Specified by:**[retainAll](http://docs.google.com/java/util/Collection.html#retainAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[retainAll](http://docs.google.com/java/util/List.html#retainAll(java.util.Collection)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**c - collection containing elements to be retained in this list **Returns:**true if this list changed as a result of the call **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this list is incompatible with the specified collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if this list contains a null element and the specified collection does not permit null elements (optional), or if the specified collection is null**See Also:**[remove(Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#remove(java.lang.Object))

### addAllAbsent

public int **addAllAbsent**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)

Appends all of the elements in the specified collection that are not already contained in this list, to the end of this list, in the order that they are returned by the specified collection's iterator.

**Parameters:**c - collection containing elements to be added to this list **Returns:**the number of elements added **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null**See Also:**[addIfAbsent(Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#addIfAbsent(E))

### clear

public void **clear**()

Removes all of the elements from this list. The list will be empty after this call returns.

**Specified by:**[clear](http://docs.google.com/java/util/Collection.html#clear()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[clear](http://docs.google.com/java/util/List.html#clear()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>

### addAll

public boolean **addAll**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)

Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator.

**Specified by:**[addAll](http://docs.google.com/java/util/Collection.html#addAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[addAll](http://docs.google.com/java/util/List.html#addAll(java.util.Collection)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**c - collection containing elements to be added to this list **Returns:**true if this list changed as a result of the call **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null**See Also:**[add(Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#add(E))

### addAll

public boolean **addAll**(int index,  
 [Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> c)

Inserts all of the elements in the specified collection into this list, starting at the specified position. Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator.

**Specified by:**[addAll](http://docs.google.com/java/util/List.html#addAll(int,%20java.util.Collection)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - index at which to insert the first element from the specified collectionc - collection containing elements to be added to this list **Returns:**true if this list changed as a result of the call **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index > size()) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null**See Also:**[add(int,Object)](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html#add(int,%20E))

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns a string representation of this list. The string representation consists of the string representations of the list's elements in the order they are returned by its iterator, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (comma and space). Elements are converted to strings as by [String.valueOf(Object)](http://docs.google.com/java/lang/String.html#valueOf(java.lang.Object)).

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this list

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) o)

Compares the specified object with this list for equality. Returns true if the specified object is the same object as this object, or if it is also a [List](http://docs.google.com/java/util/List.html) and the sequence of elements returned by an [iterator](http://docs.google.com/java/util/List.html#iterator()) over the specified list is the same as the sequence returned by an iterator over this list. The two sequences are considered to be the same if they have the same length and corresponding elements at the same position in the sequence are *equal*. Two elements e1 and e2 are considered *equal* if (e1==null ? e2==null : e1.equals(e2)).

**Specified by:**[equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[equals](http://docs.google.com/java/util/List.html#equals(java.lang.Object)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**o - the object to be compared for equality with this list **Returns:**true if the specified object is equal to this list**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Returns the hash code value for this list.

This implementation uses the definition in [List.hashCode()](http://docs.google.com/java/util/List.html#hashCode()).

**Specified by:**[hashCode](http://docs.google.com/java/util/Collection.html#hashCode()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[hashCode](http://docs.google.com/java/util/List.html#hashCode()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the hash code value for this list**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### iterator

public [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **iterator**()

Returns an iterator over the elements in this list in proper sequence.

The returned iterator provides a snapshot of the state of the list when the iterator was constructed. No synchronization is needed while traversing the iterator. The iterator does *NOT* support the remove method.

**Specified by:**[iterator](http://docs.google.com/java/lang/Iterable.html#iterator()) in interface [Iterable](http://docs.google.com/java/lang/Iterable.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[iterator](http://docs.google.com/java/util/Collection.html#iterator()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)>**Specified by:**[iterator](http://docs.google.com/java/util/List.html#iterator()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Returns:**an iterator over the elements in this list in proper sequence

### listIterator

public [ListIterator](http://docs.google.com/java/util/ListIterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **listIterator**()

Returns a list iterator over the elements in this list (in proper sequence).

The returned iterator provides a snapshot of the state of the list when the iterator was constructed. No synchronization is needed while traversing the iterator. The iterator does *NOT* support the remove, set or add methods.

**Specified by:**[listIterator](http://docs.google.com/java/util/List.html#listIterator()) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Returns:**a list iterator over the elements in this list (in proper sequence)

### listIterator

public [ListIterator](http://docs.google.com/java/util/ListIterator.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **listIterator**(int index)

Returns a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to [next](http://docs.google.com/java/util/ListIterator.html#next()). An initial call to [previous](http://docs.google.com/java/util/ListIterator.html#previous()) would return the element with the specified index minus one.

The returned iterator provides a snapshot of the state of the list when the iterator was constructed. No synchronization is needed while traversing the iterator. The iterator does *NOT* support the remove, set or add methods.

**Specified by:**[listIterator](http://docs.google.com/java/util/List.html#listIterator(int)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**index - index of first element to be returned from the list iterator (by a call to the next method) **Returns:**a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index > size())

### subList

public [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **subList**(int fromIndex,  
 int toIndex)

Returns a view of the portion of this list between fromIndex, inclusive, and toIndex, exclusive. The returned list is backed by this list, so changes in the returned list are reflected in this list, and vice-versa. While mutative operations are supported, they are probably not very useful for CopyOnWriteArrayLists.

The semantics of the list returned by this method become undefined if the backing list (i.e., this list) is *structurally modified* in any way other than via the returned list. (Structural modifications are those that change the size of the list, or otherwise perturb it in such a fashion that iterations in progress may yield incorrect results.)

**Specified by:**[subList](http://docs.google.com/java/util/List.html#subList(int,%20int)) in interface [List](http://docs.google.com/java/util/List.html)<[E](http://docs.google.com/java/util/concurrent/CopyOnWriteArrayList.html)> **Parameters:**fromIndex - low endpoint (inclusive) of the subListtoIndex - high endpoint (exclusive) of the subList **Returns:**a view of the specified range within this list **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - for an illegal endpoint index value (fromIndex < 0 || toIndex > size || fromIndex > toIndex)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CopyOnWriteArrayList.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/ConcurrentSkipListSet.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/CopyOnWriteArraySet.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/CopyOnWriteArrayList.html)    [**NO FRAMES**](http://docs.google.com/CopyOnWriteArrayList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).