| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ArrayList.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/ArrayDeque.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Arrays.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/ArrayList.html)    [**NO FRAMES**](http://docs.google.com/ArrayList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

## **java.util**

Class ArrayList<E>

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

**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) **Direct Known Subclasses:** [AttributeList](http://docs.google.com/javax/management/AttributeList.html), [RoleList](http://docs.google.com/javax/management/relation/RoleList.html), [RoleUnresolvedList](http://docs.google.com/javax/management/relation/RoleUnresolvedList.html)

public class **ArrayList<E>**extends [AbstractList](http://docs.google.com/java/util/AbstractList.html)<E>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)

Resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)

The size, isEmpty, get, set, iterator, and listIterator operations run in constant time. The add operation runs in *amortized constant time*, that is, adding n elements requires O(n) time. All of the other operations run in linear time (roughly speaking). The constant factor is low compared to that for the LinkedList implementation.

Each ArrayList instance has a *capacity*. The capacity is the size of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an ArrayList, its capacity grows automatically. The details of the growth policy are not specified beyond the fact that adding an element has constant amortized time cost.

An application can increase the capacity of an ArrayList instance before adding a large number of elements using the ensureCapacity operation. This may reduce the amount of incremental reallocation.

**Note that this implementation is not synchronized.** If multiple threads access an ArrayList instance concurrently, and at least one of the threads modifies the list structurally, it *must* be synchronized externally. (A structural modification is any operation that adds or deletes one or more elements, or explicitly resizes the backing array; merely setting the value of an element is not a structural modification.) This is typically accomplished by synchronizing on some object that naturally encapsulates the list. If no such object exists, the list should be "wrapped" using the [Collections.synchronizedList](http://docs.google.com/java/util/Collections.html#synchronizedList(java.util.List)) method. This is best done at creation time, to prevent accidental unsynchronized access to the list:

List list = Collections.synchronizedList(new ArrayList(...));

The iterators returned by this class's iterator and listIterator methods are *fail-fast*: if the list is structurally modified at any time after the iterator is created, in any way except through the iterator's own remove or add methods, the iterator will throw a [ConcurrentModificationException](http://docs.google.com/java/util/ConcurrentModificationException.html). Thus, in the face of concurrent modification, the iterator fails quickly and cleanly, rather than risking arbitrary, non-deterministic behavior at an undetermined time in the future.

Note that the fail-fast behavior of an iterator cannot be guaranteed as it is, generally speaking, impossible to make any hard guarantees in the presence of unsynchronized concurrent modification. Fail-fast iterators throw ConcurrentModificationException on a best-effort basis. Therefore, it would be wrong to write a program that depended on this exception for its correctness: *the fail-fast behavior of iterators should be used only to detect bugs.*

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

**Since:** 1.2 **See Also:**[Collection](http://docs.google.com/java/util/Collection.html), [List](http://docs.google.com/java/util/List.html), [LinkedList](http://docs.google.com/java/util/LinkedList.html), [Vector](http://docs.google.com/java/util/Vector.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.util.ArrayList)

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.util.**[**AbstractList**](http://docs.google.com/java/util/AbstractList.html) |
| --- |
| [modCount](http://docs.google.com/java/util/AbstractList.html#modCount) |

| **Constructor Summary** | |
| --- | --- |
| [**ArrayList**](http://docs.google.com/java/util/ArrayList.html#ArrayList())()            Constructs an empty list with an initial capacity of ten. |
| [**ArrayList**](http://docs.google.com/java/util/ArrayList.html#ArrayList(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/ArrayList.html)> c)            Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator. |
| [**ArrayList**](http://docs.google.com/java/util/ArrayList.html#ArrayList(int))(int initialCapacity)            Constructs an empty list with the specified initial capacity. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/ArrayList.html#add(E))([E](http://docs.google.com/java/util/ArrayList.html) e)            Appends the specified element to the end of this list. |
| void | [**add**](http://docs.google.com/java/util/ArrayList.html#add(int,%20E))(int index, [E](http://docs.google.com/java/util/ArrayList.html) element)            Inserts the specified element at the specified position in this list. |
| boolean | [**addAll**](http://docs.google.com/java/util/ArrayList.html#addAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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/ArrayList.html)> c)            Inserts all of the elements in the specified collection into this list, starting at the specified position. |
| void | [**clear**](http://docs.google.com/java/util/ArrayList.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/ArrayList.html#clone())()            Returns a shallow copy of this ArrayList instance. |
| boolean | [**contains**](http://docs.google.com/java/util/ArrayList.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this list contains the specified element. |
| void | [**ensureCapacity**](http://docs.google.com/java/util/ArrayList.html#ensureCapacity(int))(int minCapacity)            Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument. |
| [E](http://docs.google.com/java/util/ArrayList.html) | [**get**](http://docs.google.com/java/util/ArrayList.html#get(int))(int index)            Returns the element at the specified position in this list. |
| int | [**indexOf**](http://docs.google.com/java/util/ArrayList.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/ArrayList.html#isEmpty())()            Returns true if this list contains no elements. |
| int | [**lastIndexOf**](http://docs.google.com/java/util/ArrayList.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. |
| [E](http://docs.google.com/java/util/ArrayList.html) | [**remove**](http://docs.google.com/java/util/ArrayList.html#remove(int))(int index)            Removes the element at the specified position in this list. |
| boolean | [**remove**](http://docs.google.com/java/util/ArrayList.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. |
| protected  void | [**removeRange**](http://docs.google.com/java/util/ArrayList.html#removeRange(int,%20int))(int fromIndex, int toIndex)            Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. |
| [E](http://docs.google.com/java/util/ArrayList.html) | [**set**](http://docs.google.com/java/util/ArrayList.html#set(int,%20E))(int index, [E](http://docs.google.com/java/util/ArrayList.html) element)            Replaces the element at the specified position in this list with the specified element. |
| int | [**size**](http://docs.google.com/java/util/ArrayList.html#size())()            Returns the number of elements in this list. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/ArrayList.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/ArrayList.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. |
| void | [**trimToSize**](http://docs.google.com/java/util/ArrayList.html#trimToSize())()            Trims the capacity of this ArrayList instance to be the list's current size. |

| **Methods inherited from class java.util.**[**AbstractList**](http://docs.google.com/java/util/AbstractList.html) |
| --- |
| [equals](http://docs.google.com/java/util/AbstractList.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/AbstractList.html#hashCode()), [iterator](http://docs.google.com/java/util/AbstractList.html#iterator()), [listIterator](http://docs.google.com/java/util/AbstractList.html#listIterator()), [listIterator](http://docs.google.com/java/util/AbstractList.html#listIterator(int)), [subList](http://docs.google.com/java/util/AbstractList.html#subList(int,%20int)) |

| **Methods inherited from class java.util.**[**AbstractCollection**](http://docs.google.com/java/util/AbstractCollection.html) |
| --- |
| [containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)), [removeAll](http://docs.google.com/java/util/AbstractCollection.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/AbstractCollection.html#retainAll(java.util.Collection)), [toString](http://docs.google.com/java/util/AbstractCollection.html#toString()) |

| **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)) |

| **Methods inherited from interface java.util.**[**List**](http://docs.google.com/java/util/List.html) |
| --- |
| [containsAll](http://docs.google.com/java/util/List.html#containsAll(java.util.Collection)), [equals](http://docs.google.com/java/util/List.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/List.html#hashCode()), [iterator](http://docs.google.com/java/util/List.html#iterator()), [listIterator](http://docs.google.com/java/util/List.html#listIterator()), [listIterator](http://docs.google.com/java/util/List.html#listIterator(int)), [removeAll](http://docs.google.com/java/util/List.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/List.html#retainAll(java.util.Collection)), [subList](http://docs.google.com/java/util/List.html#subList(int,%20int)) |

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

### ArrayList

public **ArrayList**(int initialCapacity)

Constructs an empty list with the specified initial capacity.

**Parameters:**initialCapacity - the initial capacity of the list **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified initial capacity is negative

### ArrayList

public **ArrayList**()

Constructs an empty list with an initial capacity of ten.

### ArrayList

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

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

**Parameters:**c - the collection whose elements are to be placed into this list **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null

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

### trimToSize

public void **trimToSize**()

Trims the capacity of this ArrayList instance to be the list's current size. An application can use this operation to minimize the storage of an ArrayList instance.

### ensureCapacity

public void **ensureCapacity**(int minCapacity)

Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.

**Parameters:**minCapacity - the desired minimum capacity

### 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/ArrayList.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/ArrayList.html)>**Specified by:**[size](http://docs.google.com/java/util/AbstractCollection.html#size()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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/ArrayList.html)>**Overrides:**[isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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/ArrayList.html)>**Overrides:**[contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.html)>**Overrides:**[indexOf](http://docs.google.com/java/util/AbstractList.html#indexOf(java.lang.Object)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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

### 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/ArrayList.html)>**Overrides:**[lastIndexOf](http://docs.google.com/java/util/AbstractList.html#lastIndexOf(java.lang.Object)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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

### clone

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

Returns a shallow copy of this ArrayList instance. (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 ArrayList instance**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/ArrayList.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/ArrayList.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.html)> **Returns:**an array containing all of the elements in this list in proper sequence**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 the list fits in the specified array with room to spare (i.e., the array has more elements than the list), the element in the array immediately following the end of the collection is set to null. (This is useful in determining the length of the list *only* if the caller knows that the list does not contain any null elements.)

**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/ArrayList.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/ArrayList.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray(T%5B%5D)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.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 the elements of the 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/ArrayList.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/ArrayList.html)>**Specified by:**[get](http://docs.google.com/java/util/AbstractList.html#get(int)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.html) **set**(int index,  
 [E](http://docs.google.com/java/util/ArrayList.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/ArrayList.html)>**Overrides:**[set](http://docs.google.com/java/util/AbstractList.html#set(int,%20E)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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/ArrayList.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/ArrayList.html)>**Overrides:**[add](http://docs.google.com/java/util/AbstractList.html#add(E)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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/ArrayList.html)>**Overrides:**[add](http://docs.google.com/java/util/AbstractList.html#add(int,%20E)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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/ArrayList.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).

**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/ArrayList.html)>**Overrides:**[remove](http://docs.google.com/java/util/AbstractList.html#remove(int)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.html)> **Parameters:**index - the index of the element to be removed **Returns:**the element that was removed from the list **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 the 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/ArrayList.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/ArrayList.html)>**Overrides:**[remove](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.html)> **Parameters:**o - element to be removed from this list, if present **Returns:**true if this list contained the specified element

### 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/ArrayList.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/ArrayList.html)>**Overrides:**[clear](http://docs.google.com/java/util/AbstractList.html#clear()) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.html)>

### addAll

public boolean **addAll**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/ArrayList.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. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this list, and this list is nonempty.)

**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/ArrayList.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/ArrayList.html)>**Overrides:**[addAll](http://docs.google.com/java/util/AbstractCollection.html#addAll(java.util.Collection)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/ArrayList.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:**[AbstractCollection.add(Object)](http://docs.google.com/java/util/AbstractCollection.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/ArrayList.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 the 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/ArrayList.html)>**Overrides:**[addAll](http://docs.google.com/java/util/AbstractList.html#addAll(int,%20java.util.Collection)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.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

### removeRange

protected void **removeRange**(int fromIndex,  
 int toIndex)

Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. Shifts any succeeding elements to the left (reduces their index). This call shortens the list by (toIndex - fromIndex) elements. (If toIndex==fromIndex, this operation has no effect.)

**Overrides:**[removeRange](http://docs.google.com/java/util/AbstractList.html#removeRange(int,%20int)) in class [AbstractList](http://docs.google.com/java/util/AbstractList.html)<[E](http://docs.google.com/java/util/ArrayList.html)> **Parameters:**fromIndex - index of first element to be removedtoIndex - index after last element to be removed **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex or toIndex out of range (fromIndex < 0 || fromIndex >= size() || toIndex > size() || toIndex < fromIndex)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ArrayList.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/ArrayDeque.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Arrays.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/ArrayList.html)    [**NO FRAMES**](http://docs.google.com/ArrayList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

[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).