| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LinkedList.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/LinkedHashSet.html)   [**NEXT CLASS**](http://docs.google.com/java/util/List.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/LinkedList.html)    [**NO FRAMES**](http://docs.google.com/LinkedList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#lnxbz9) | [METHOD](#44sinio) |

## **java.util**

Class LinkedList<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.AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<E>  
 **java.util.LinkedList<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>, [Deque](http://docs.google.com/java/util/Deque.html)<E>, [List](http://docs.google.com/java/util/List.html)<E>, [Queue](http://docs.google.com/java/util/Queue.html)<E>

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

Linked list implementation of the List interface. Implements all optional list operations, and permits all elements (including null). In addition to implementing the List interface, the LinkedList class provides uniformly named methods to get, remove and insert an element at the beginning and end of the list. These operations allow linked lists to be used as a stack, [queue](http://docs.google.com/java/util/Queue.html), or [double-ended queue](http://docs.google.com/java/util/Deque.html).

The class implements the Deque interface, providing first-in-first-out queue operations for add, poll, along with other stack and deque operations.

All of the operations perform as could be expected for a doubly-linked list. Operations that index into the list will traverse the list from the beginning or the end, whichever is closer to the specified index.

**Note that this implementation is not synchronized.** If multiple threads access a linked list 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; 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 LinkedList(...));

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

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

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/LinkedList.html#add(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Appends the specified element to the end of this list. |
| void | [**add**](http://docs.google.com/java/util/LinkedList.html#add(int,%20E))(int index, [E](http://docs.google.com/java/util/LinkedList.html) element)            Inserts the specified element at the specified position in this list. |
| boolean | [**addAll**](http://docs.google.com/java/util/LinkedList.html#addAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/LinkedList.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/LinkedList.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/LinkedList.html)> c)            Inserts all of the elements in the specified collection into this list, starting at the specified position. |
| void | [**addFirst**](http://docs.google.com/java/util/LinkedList.html#addFirst(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Inserts the specified element at the beginning of this list. |
| void | [**addLast**](http://docs.google.com/java/util/LinkedList.html#addLast(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Appends the specified element to the end of this list. |
| void | [**clear**](http://docs.google.com/java/util/LinkedList.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/LinkedList.html#clone())()            Returns a shallow copy of this LinkedList. |
| boolean | [**contains**](http://docs.google.com/java/util/LinkedList.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this list contains the specified element. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/LinkedList.html)> | [**descendingIterator**](http://docs.google.com/java/util/LinkedList.html#descendingIterator())()            Returns an iterator over the elements in this deque in reverse sequential order. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**element**](http://docs.google.com/java/util/LinkedList.html#element())()            Retrieves, but does not remove, the head (first element) of this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**get**](http://docs.google.com/java/util/LinkedList.html#get(int))(int index)            Returns the element at the specified position in this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**getFirst**](http://docs.google.com/java/util/LinkedList.html#getFirst())()            Returns the first element in this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**getLast**](http://docs.google.com/java/util/LinkedList.html#getLast())()            Returns the last element in this list. |
| int | [**indexOf**](http://docs.google.com/java/util/LinkedList.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. |
| int | [**lastIndexOf**](http://docs.google.com/java/util/LinkedList.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/LinkedList.html)> | [**listIterator**](http://docs.google.com/java/util/LinkedList.html#listIterator(int))(int index)            Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list. |
| boolean | [**offer**](http://docs.google.com/java/util/LinkedList.html#offer(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Adds the specified element as the tail (last element) of this list. |
| boolean | [**offerFirst**](http://docs.google.com/java/util/LinkedList.html#offerFirst(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Inserts the specified element at the front of this list. |
| boolean | [**offerLast**](http://docs.google.com/java/util/LinkedList.html#offerLast(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Inserts the specified element at the end of this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**peek**](http://docs.google.com/java/util/LinkedList.html#peek())()            Retrieves, but does not remove, the head (first element) of this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**peekFirst**](http://docs.google.com/java/util/LinkedList.html#peekFirst())()            Retrieves, but does not remove, the first element of this list, or returns null if this list is empty. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**peekLast**](http://docs.google.com/java/util/LinkedList.html#peekLast())()            Retrieves, but does not remove, the last element of this list, or returns null if this list is empty. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**poll**](http://docs.google.com/java/util/LinkedList.html#poll())()            Retrieves and removes the head (first element) of this list |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**pollFirst**](http://docs.google.com/java/util/LinkedList.html#pollFirst())()            Retrieves and removes the first element of this list, or returns null if this list is empty. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**pollLast**](http://docs.google.com/java/util/LinkedList.html#pollLast())()            Retrieves and removes the last element of this list, or returns null if this list is empty. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**pop**](http://docs.google.com/java/util/LinkedList.html#pop())()            Pops an element from the stack represented by this list. |
| void | [**push**](http://docs.google.com/java/util/LinkedList.html#push(E))([E](http://docs.google.com/java/util/LinkedList.html) e)            Pushes an element onto the stack represented by this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**remove**](http://docs.google.com/java/util/LinkedList.html#remove())()            Retrieves and removes the head (first element) of this list. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**remove**](http://docs.google.com/java/util/LinkedList.html#remove(int))(int index)            Removes the element at the specified position in this list. |
| boolean | [**remove**](http://docs.google.com/java/util/LinkedList.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. |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**removeFirst**](http://docs.google.com/java/util/LinkedList.html#removeFirst())()            Removes and returns the first element from this list. |
| boolean | [**removeFirstOccurrence**](http://docs.google.com/java/util/LinkedList.html#removeFirstOccurrence(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the first occurrence of the specified element in this list (when traversing the list from head to tail). |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**removeLast**](http://docs.google.com/java/util/LinkedList.html#removeLast())()            Removes and returns the last element from this list. |
| boolean | [**removeLastOccurrence**](http://docs.google.com/java/util/LinkedList.html#removeLastOccurrence(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the last occurrence of the specified element in this list (when traversing the list from head to tail). |
| [E](http://docs.google.com/java/util/LinkedList.html) | [**set**](http://docs.google.com/java/util/LinkedList.html#set(int,%20E))(int index, [E](http://docs.google.com/java/util/LinkedList.html) element)            Replaces the element at the specified position in this list with the specified element. |
| int | [**size**](http://docs.google.com/java/util/LinkedList.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/LinkedList.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/LinkedList.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. |

| **Methods inherited from class java.util.**[**AbstractSequentialList**](http://docs.google.com/java/util/AbstractSequentialList.html) |
| --- |
| [iterator](http://docs.google.com/java/util/AbstractSequentialList.html#iterator()) |

| **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()), [listIterator](http://docs.google.com/java/util/AbstractList.html#listIterator()), [removeRange](http://docs.google.com/java/util/AbstractList.html#removeRange(int,%20int)), [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)), [isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()), [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()), [isEmpty](http://docs.google.com/java/util/List.html#isEmpty()), [iterator](http://docs.google.com/java/util/List.html#iterator()), [listIterator](http://docs.google.com/java/util/List.html#listIterator()), [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)) |

| **Methods inherited from interface java.util.**[**Deque**](http://docs.google.com/java/util/Deque.html) |
| --- |
| [iterator](http://docs.google.com/java/util/Deque.html#iterator()) |

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

### LinkedList

public **LinkedList**()

Constructs an empty list.

### LinkedList

public **LinkedList**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/LinkedList.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** |
| --- |

### getFirst

public [E](http://docs.google.com/java/util/LinkedList.html) **getFirst**()

Returns the first element in this list.

**Specified by:**[getFirst](http://docs.google.com/java/util/Deque.html#getFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the first element in this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty

### getLast

public [E](http://docs.google.com/java/util/LinkedList.html) **getLast**()

Returns the last element in this list.

**Specified by:**[getLast](http://docs.google.com/java/util/Deque.html#getLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the last element in this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty

### removeFirst

public [E](http://docs.google.com/java/util/LinkedList.html) **removeFirst**()

Removes and returns the first element from this list.

**Specified by:**[removeFirst](http://docs.google.com/java/util/Deque.html#removeFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the first element from this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty

### removeLast

public [E](http://docs.google.com/java/util/LinkedList.html) **removeLast**()

Removes and returns the last element from this list.

**Specified by:**[removeLast](http://docs.google.com/java/util/Deque.html#removeLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the last element from this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty

### addFirst

public void **addFirst**([E](http://docs.google.com/java/util/LinkedList.html) e)

Inserts the specified element at the beginning of this list.

**Specified by:**[addFirst](http://docs.google.com/java/util/Deque.html#addFirst(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to add

### addLast

public void **addLast**([E](http://docs.google.com/java/util/LinkedList.html) e)

Appends the specified element to the end of this list.

This method is equivalent to [add(E)](http://docs.google.com/java/util/LinkedList.html#add(E)).

**Specified by:**[addLast](http://docs.google.com/java/util/Deque.html#addLast(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to add

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

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

### add

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

Appends the specified element to the end of this list.

This method is equivalent to [addLast(E)](http://docs.google.com/java/util/LinkedList.html#addLast(E)).

**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/LinkedList.html)>**Specified by:**[add](http://docs.google.com/java/util/Deque.html#add(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.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/LinkedList.html)>**Specified by:**[add](http://docs.google.com/java/util/Queue.html#add(E)) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.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/LinkedList.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)))

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

### addAll

public boolean **addAll**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/LinkedList.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. (Note that this will occur if the specified collection is this list, and it's 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/LinkedList.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/LinkedList.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/LinkedList.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/LinkedList.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/LinkedList.html)>**Overrides:**[addAll](http://docs.google.com/java/util/AbstractSequentialList.html#addAll(int,%20java.util.Collection)) in class [AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<[E](http://docs.google.com/java/util/LinkedList.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

### clear

public void **clear**()

Removes all of the elements from this list.

**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/LinkedList.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/LinkedList.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/LinkedList.html)>

### get

public [E](http://docs.google.com/java/util/LinkedList.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/LinkedList.html)>**Overrides:**[get](http://docs.google.com/java/util/AbstractSequentialList.html#get(int)) in class [AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<[E](http://docs.google.com/java/util/LinkedList.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/LinkedList.html) **set**(int index,  
 [E](http://docs.google.com/java/util/LinkedList.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/LinkedList.html)>**Overrides:**[set](http://docs.google.com/java/util/AbstractSequentialList.html#set(int,%20E)) in class [AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<[E](http://docs.google.com/java/util/LinkedList.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 void **add**(int index,  
 [E](http://docs.google.com/java/util/LinkedList.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/LinkedList.html)>**Overrides:**[add](http://docs.google.com/java/util/AbstractSequentialList.html#add(int,%20E)) in class [AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<[E](http://docs.google.com/java/util/LinkedList.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/LinkedList.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/LinkedList.html)>**Overrides:**[remove](http://docs.google.com/java/util/AbstractSequentialList.html#remove(int)) in class [AbstractSequentialList](http://docs.google.com/java/util/AbstractSequentialList.html)<[E](http://docs.google.com/java/util/LinkedList.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())

### 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/LinkedList.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/LinkedList.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/LinkedList.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/LinkedList.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

### peek

public [E](http://docs.google.com/java/util/LinkedList.html) **peek**()

Retrieves, but does not remove, the head (first element) of this list.

**Specified by:**[peek](http://docs.google.com/java/util/Deque.html#peek()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)>**Specified by:**[peek](http://docs.google.com/java/util/Queue.html#peek()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the head of this list, or null if this list is empty**Since:** 1.5

### element

public [E](http://docs.google.com/java/util/LinkedList.html) **element**()

Retrieves, but does not remove, the head (first element) of this list.

**Specified by:**[element](http://docs.google.com/java/util/Deque.html#element()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)>**Specified by:**[element](http://docs.google.com/java/util/Queue.html#element()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the head of this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty**Since:** 1.5

### poll

public [E](http://docs.google.com/java/util/LinkedList.html) **poll**()

Retrieves and removes the head (first element) of this list

**Specified by:**[poll](http://docs.google.com/java/util/Deque.html#poll()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)>**Specified by:**[poll](http://docs.google.com/java/util/Queue.html#poll()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the head of this list, or null if this list is empty**Since:** 1.5

### remove

public [E](http://docs.google.com/java/util/LinkedList.html) **remove**()

Retrieves and removes the head (first element) of this list.

**Specified by:**[remove](http://docs.google.com/java/util/Deque.html#remove()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)>**Specified by:**[remove](http://docs.google.com/java/util/Queue.html#remove()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the head of this list **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty**Since:** 1.5

### offer

public boolean **offer**([E](http://docs.google.com/java/util/LinkedList.html) e)

Adds the specified element as the tail (last element) of this list.

**Specified by:**[offer](http://docs.google.com/java/util/Deque.html#offer(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)>**Specified by:**[offer](http://docs.google.com/java/util/Queue.html#offer(E)) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to add **Returns:**true (as specified by [Queue.offer(E)](http://docs.google.com/java/util/Queue.html#offer(E)))**Since:** 1.5

### offerFirst

public boolean **offerFirst**([E](http://docs.google.com/java/util/LinkedList.html) e)

Inserts the specified element at the front of this list.

**Specified by:**[offerFirst](http://docs.google.com/java/util/Deque.html#offerFirst(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to insert **Returns:**true (as specified by [Deque.offerFirst(E)](http://docs.google.com/java/util/Deque.html#offerFirst(E)))**Since:** 1.6

### offerLast

public boolean **offerLast**([E](http://docs.google.com/java/util/LinkedList.html) e)

Inserts the specified element at the end of this list.

**Specified by:**[offerLast](http://docs.google.com/java/util/Deque.html#offerLast(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to insert **Returns:**true (as specified by [Deque.offerLast(E)](http://docs.google.com/java/util/Deque.html#offerLast(E)))**Since:** 1.6

### peekFirst

public [E](http://docs.google.com/java/util/LinkedList.html) **peekFirst**()

Retrieves, but does not remove, the first element of this list, or returns null if this list is empty.

**Specified by:**[peekFirst](http://docs.google.com/java/util/Deque.html#peekFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the first element of this list, or null if this list is empty**Since:** 1.6

### peekLast

public [E](http://docs.google.com/java/util/LinkedList.html) **peekLast**()

Retrieves, but does not remove, the last element of this list, or returns null if this list is empty.

**Specified by:**[peekLast](http://docs.google.com/java/util/Deque.html#peekLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the last element of this list, or null if this list is empty**Since:** 1.6

### pollFirst

public [E](http://docs.google.com/java/util/LinkedList.html) **pollFirst**()

Retrieves and removes the first element of this list, or returns null if this list is empty.

**Specified by:**[pollFirst](http://docs.google.com/java/util/Deque.html#pollFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the first element of this list, or null if this list is empty**Since:** 1.6

### pollLast

public [E](http://docs.google.com/java/util/LinkedList.html) **pollLast**()

Retrieves and removes the last element of this list, or returns null if this list is empty.

**Specified by:**[pollLast](http://docs.google.com/java/util/Deque.html#pollLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the last element of this list, or null if this list is empty**Since:** 1.6

### push

public void **push**([E](http://docs.google.com/java/util/LinkedList.html) e)

Pushes an element onto the stack represented by this list. In other words, inserts the element at the front of this list.

This method is equivalent to [addFirst(E)](http://docs.google.com/java/util/LinkedList.html#addFirst(E)).

**Specified by:**[push](http://docs.google.com/java/util/Deque.html#push(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**e - the element to push**Since:** 1.6

### pop

public [E](http://docs.google.com/java/util/LinkedList.html) **pop**()

Pops an element from the stack represented by this list. In other words, removes and returns the first element of this list.

This method is equivalent to [removeFirst()](http://docs.google.com/java/util/LinkedList.html#removeFirst()).

**Specified by:**[pop](http://docs.google.com/java/util/Deque.html#pop()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**the element at the front of this list (which is the top of the stack represented by this list) **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this list is empty**Since:** 1.6

### removeFirstOccurrence

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

Removes the first occurrence of the specified element in this list (when traversing the list from head to tail). If the list does not contain the element, it is unchanged.

**Specified by:**[removeFirstOccurrence](http://docs.google.com/java/util/Deque.html#removeFirstOccurrence(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**o - element to be removed from this list, if present **Returns:**true if the list contained the specified element**Since:** 1.6

### removeLastOccurrence

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

Removes the last occurrence of the specified element in this list (when traversing the list from head to tail). If the list does not contain the element, it is unchanged.

**Specified by:**[removeLastOccurrence](http://docs.google.com/java/util/Deque.html#removeLastOccurrence(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Parameters:**o - element to be removed from this list, if present **Returns:**true if the list contained the specified element**Since:** 1.6

### listIterator

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

Returns a list-iterator of the elements in this list (in proper sequence), starting at the specified position in the list. Obeys the general contract of List.listIterator(int).

The list-iterator is *fail-fast*: if the list is structurally modified at any time after the Iterator is created, in any way except through the list-iterator's own remove or add methods, the list-iterator will throw a ConcurrentModificationException. 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.

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

### descendingIterator

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#descendingIterator()) Returns an iterator over the elements in this deque in reverse sequential order. The elements will be returned in order from last (tail) to first (head).

**Specified by:**[descendingIterator](http://docs.google.com/java/util/Deque.html#descendingIterator()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/LinkedList.html)> **Returns:**an iterator over the elements in this deque in reverse sequence**Since:** 1.6

### clone

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

Returns a shallow copy of this LinkedList. (The elements themselves are not cloned.)

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

Like the [toArray()](http://docs.google.com/java/util/LinkedList.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/LinkedList.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/LinkedList.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/LinkedList.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

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LinkedList.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/LinkedHashSet.html)   [**NEXT CLASS**](http://docs.google.com/java/util/List.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/LinkedList.html)    [**NO FRAMES**](http://docs.google.com/LinkedList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#lnxbz9) | [METHOD](#44sinio) |

[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).