| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ListIterator.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/List.html)   [**NEXT CLASS**](http://docs.google.com/java/util/ListResourceBundle.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/ListIterator.html)    [**NO FRAMES**](http://docs.google.com/ListIterator.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **java.util**

Interface ListIterator<E>

**All Superinterfaces:** [Iterator](http://docs.google.com/java/util/Iterator.html)<E>

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

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. A ListIterator has no current element; its *cursor position* always lies between the element that would be returned by a call to previous() and the element that would be returned by a call to next(). An iterator for a list of length n has n+1 possible cursor positions, as illustrated by the carets (^) below:

Element(0) Element(1) Element(2) ... Element(n-1)  
 cursor positions: ^ ^ ^ ^ ^

Note that the [remove()](http://docs.google.com/java/util/ListIterator.html#remove()) and [set(Object)](http://docs.google.com/java/util/ListIterator.html#set(E)) methods are *not* defined in terms of the cursor position; they are defined to operate on the last element returned by a call to [next()](http://docs.google.com/java/util/ListIterator.html#next()) or [previous()](http://docs.google.com/java/util/ListIterator.html#previous()).

This interface 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), [Iterator](http://docs.google.com/java/util/Iterator.html), [Enumeration](http://docs.google.com/java/util/Enumeration.html), [List.listIterator()](http://docs.google.com/java/util/List.html#listIterator())

| **Method Summary** | |
| --- | --- |
| void | [**add**](http://docs.google.com/java/util/ListIterator.html#add(E))([E](http://docs.google.com/java/util/ListIterator.html) e)            Inserts the specified element into the list (optional operation). |
| boolean | [**hasNext**](http://docs.google.com/java/util/ListIterator.html#hasNext())()            Returns true if this list iterator has more elements when traversing the list in the forward direction. |
| boolean | [**hasPrevious**](http://docs.google.com/java/util/ListIterator.html#hasPrevious())()            Returns true if this list iterator has more elements when traversing the list in the reverse direction. |
| [E](http://docs.google.com/java/util/ListIterator.html) | [**next**](http://docs.google.com/java/util/ListIterator.html#next())()            Returns the next element in the list. |
| int | [**nextIndex**](http://docs.google.com/java/util/ListIterator.html#nextIndex())()            Returns the index of the element that would be returned by a subsequent call to next. |
| [E](http://docs.google.com/java/util/ListIterator.html) | [**previous**](http://docs.google.com/java/util/ListIterator.html#previous())()            Returns the previous element in the list. |
| int | [**previousIndex**](http://docs.google.com/java/util/ListIterator.html#previousIndex())()            Returns the index of the element that would be returned by a subsequent call to previous. |
| void | [**remove**](http://docs.google.com/java/util/ListIterator.html#remove())()            Removes from the list the last element that was returned by next or previous (optional operation). |
| void | [**set**](http://docs.google.com/java/util/ListIterator.html#set(E))([E](http://docs.google.com/java/util/ListIterator.html) e)            Replaces the last element returned by next or previous with the specified element (optional operation). |

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

### hasNext

boolean **hasNext**()

Returns true if this list iterator has more elements when traversing the list in the forward direction. (In other words, returns true if next would return an element rather than throwing an exception.)

**Specified by:**[hasNext](http://docs.google.com/java/util/Iterator.html#hasNext()) in interface [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/ListIterator.html)> **Returns:**true if the list iterator has more elements when traversing the list in the forward direction.

### next

[E](http://docs.google.com/java/util/ListIterator.html) **next**()

Returns the next element in the list. This method may be called repeatedly to iterate through the list, or intermixed with calls to previous to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

**Specified by:**[next](http://docs.google.com/java/util/Iterator.html#next()) in interface [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/ListIterator.html)> **Returns:**the next element in the list. **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if the iteration has no next element.

### hasPrevious

boolean **hasPrevious**()

Returns true if this list iterator has more elements when traversing the list in the reverse direction. (In other words, returns true if previous would return an element rather than throwing an exception.)

**Returns:**true if the list iterator has more elements when traversing the list in the reverse direction.

### previous

[E](http://docs.google.com/java/util/ListIterator.html) **previous**()

Returns the previous element in the list. This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

**Returns:**the previous element in the list. **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if the iteration has no previous element.

### nextIndex

int **nextIndex**()

Returns the index of the element that would be returned by a subsequent call to next. (Returns list size if the list iterator is at the end of the list.)

**Returns:**the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

### previousIndex

int **previousIndex**()

Returns the index of the element that would be returned by a subsequent call to previous. (Returns -1 if the list iterator is at the beginning of the list.)

**Returns:**the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

### remove

void **remove**()

Removes from the list the last element that was returned by next or previous (optional operation). This call can only be made once per call to next or previous. It can be made only if ListIterator.add has not been called after the last call to next or previous.

**Specified by:**[remove](http://docs.google.com/java/util/Iterator.html#remove()) in interface [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/ListIterator.html)> **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the remove operation is not supported by this list iterator. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

### set

void **set**([E](http://docs.google.com/java/util/ListIterator.html) e)

Replaces the last element returned by next or previous with the specified element (optional operation). This call can be made only if neither ListIterator.remove nor ListIterator.add have been called after the last call to next or previous.

**Parameters:**e - the element with which to replace the last element returned by next or previous. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the set operation is not supported by this list iterator. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this list. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some aspect of the specified element prevents it from being added to this list. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

### add

void **add**([E](http://docs.google.com/java/util/ListIterator.html) e)

Inserts the specified element into the list (optional operation). The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

**Parameters:**e - the element to insert. **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the add method is not supported by this list iterator. [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this list. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some aspect of this element prevents it from being added to this list.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ListIterator.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/List.html)   [**NEXT CLASS**](http://docs.google.com/java/util/ListResourceBundle.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/ListIterator.html)    [**NO FRAMES**](http://docs.google.com/ListIterator.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[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).