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

## **java.util**

Class AbstractSequentialList<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<E>**

**All Implemented Interfaces:** [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> **Direct Known Subclasses:** [LinkedList](http://docs.google.com/java/util/LinkedList.html)

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

This class provides a skeletal implementation of the List interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), AbstractList should be used in preference to this class.

This class is the opposite of the AbstractList class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the listIterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's hasNext, next, hasPrevious, previous and index methods.

For a modifiable list the programmer should additionally implement the list iterator's set method. For a variable-size list the programmer should additionally implement the list iterator's remove and add methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the Collection interface specification.

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), [AbstractList](http://docs.google.com/java/util/AbstractList.html), [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)

| **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** | |
| --- | --- |
| protected | [**AbstractSequentialList**](http://docs.google.com/java/util/AbstractSequentialList.html#AbstractSequentialList())()            Sole constructor. |

| **Method Summary** | |
| --- | --- |
| void | [**add**](http://docs.google.com/java/util/AbstractSequentialList.html#add(int,%20E))(int index, [E](http://docs.google.com/java/util/AbstractSequentialList.html) element)            Inserts the specified element at the specified position in this list (optional operation). |
| boolean | [**addAll**](http://docs.google.com/java/util/AbstractSequentialList.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/AbstractSequentialList.html)> c)            Inserts all of the elements in the specified collection into this list at the specified position (optional operation). |
| [E](http://docs.google.com/java/util/AbstractSequentialList.html) | [**get**](http://docs.google.com/java/util/AbstractSequentialList.html#get(int))(int index)            Returns the element at the specified position in this list. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/AbstractSequentialList.html)> | [**iterator**](http://docs.google.com/java/util/AbstractSequentialList.html#iterator())()            Returns an iterator over the elements in this list (in proper sequence). |
| abstract  [ListIterator](http://docs.google.com/java/util/ListIterator.html)<[E](http://docs.google.com/java/util/AbstractSequentialList.html)> | [**listIterator**](http://docs.google.com/java/util/AbstractSequentialList.html#listIterator(int))(int index)            Returns a list iterator over the elements in this list (in proper sequence). |
| [E](http://docs.google.com/java/util/AbstractSequentialList.html) | [**remove**](http://docs.google.com/java/util/AbstractSequentialList.html#remove(int))(int index)            Removes the element at the specified position in this list (optional operation). |
| [E](http://docs.google.com/java/util/AbstractSequentialList.html) | [**set**](http://docs.google.com/java/util/AbstractSequentialList.html#set(int,%20E))(int index, [E](http://docs.google.com/java/util/AbstractSequentialList.html) element)            Replaces the element at the specified position in this list with the specified element (optional operation). |

| **Methods inherited from class java.util.**[**AbstractList**](http://docs.google.com/java/util/AbstractList.html) |
| --- |
| [add](http://docs.google.com/java/util/AbstractList.html#add(E)), [clear](http://docs.google.com/java/util/AbstractList.html#clear()), [equals](http://docs.google.com/java/util/AbstractList.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/AbstractList.html#hashCode()), [indexOf](http://docs.google.com/java/util/AbstractList.html#indexOf(java.lang.Object)), [lastIndexOf](http://docs.google.com/java/util/AbstractList.html#lastIndexOf(java.lang.Object)), [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) |
| --- |
| [addAll](http://docs.google.com/java/util/AbstractCollection.html#addAll(java.util.Collection)), [contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)), [containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)), [isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()), [remove](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)), [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)), [size](http://docs.google.com/java/util/AbstractCollection.html#size()), [toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray()), [toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray(T%5B%5D)), [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) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [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) |
| --- |
| [addAll](http://docs.google.com/java/util/List.html#addAll(java.util.Collection)), [contains](http://docs.google.com/java/util/List.html#contains(java.lang.Object)), [containsAll](http://docs.google.com/java/util/List.html#containsAll(java.util.Collection)), [isEmpty](http://docs.google.com/java/util/List.html#isEmpty()), [remove](http://docs.google.com/java/util/List.html#remove(java.lang.Object)), [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)), [size](http://docs.google.com/java/util/List.html#size()), [toArray](http://docs.google.com/java/util/List.html#toArray()), [toArray](http://docs.google.com/java/util/List.html#toArray(T%5B%5D)) |

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

### AbstractSequentialList

protected **AbstractSequentialList**()

Sole constructor. (For invocation by subclass constructors, typically implicit.)

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

### get

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

Returns the element at the specified position in this list.

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the element using ListIterator.next and returns it.

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

Replaces the element at the specified position in this list with the specified element (optional operation).

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it gets the current element using ListIterator.next and replaces it with ListIterator.set.

Note that this implementation will throw an UnsupportedOperationException if the list iterator does not implement the set operation.

**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/AbstractSequentialList.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/AbstractSequentialList.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:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the set operation is not supported by this list [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this list [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this list does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this list [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/AbstractSequentialList.html) element)

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

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it inserts the specified element with ListIterator.add.

Note that this implementation will throw an UnsupportedOperationException if the list iterator does not implement the add operation.

**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/AbstractSequentialList.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/AbstractSequentialList.html)> **Parameters:**index - index at which the specified element is to be insertedelement - element to be inserted **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the add operation is not supported by this list [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this list [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this list does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this list [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/AbstractSequentialList.html) **remove**(int index)

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

This implementation first gets a list iterator pointing to the indexed element (with listIterator(index)). Then, it removes the element with ListIterator.remove.

Note that this implementation will throw an UnsupportedOperationException if the list iterator does not implement the remove operation.

**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/AbstractSequentialList.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/AbstractSequentialList.html)> **Parameters:**index - the index of the element to be removed **Returns:**the element previously at the specified position **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the remove operation is not supported by this list [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index >= size())

### addAll

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

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

This implementation gets an iterator over the specified collection and a list iterator over this list pointing to the indexed element (with listIterator(index)). Then, it iterates over the specified collection, inserting the elements obtained from the iterator into this list, one at a time, using ListIterator.add followed by ListIterator.next (to skip over the added element).

Note that this implementation will throw an UnsupportedOperationException if the list iterator returned by the listIterator method does not implement the add operation.

**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/AbstractSequentialList.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/AbstractSequentialList.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:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the addAll operation is not supported by this list [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of the specified collection prevents it from being added to this list [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection contains one or more null elements and this list does not permit null elements, or if the specified collection is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of an element of the specified collection prevents it from being added to this list [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the index is out of range (index < 0 || index > size())

### iterator

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

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

This implementation merely returns a list iterator over the list.

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

### listIterator

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

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

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

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

[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).