| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractCollection.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   [**NEXT CLASS**](http://docs.google.com/java/util/AbstractList.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/AbstractCollection.html)    [**NO FRAMES**](http://docs.google.com/AbstractCollection.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

## **java.util**

Class AbstractCollection<E>

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

**All Implemented Interfaces:** [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Collection](http://docs.google.com/java/util/Collection.html)<E> **Direct Known Subclasses:** [AbstractList](http://docs.google.com/java/util/AbstractList.html), [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html), [AbstractSet](http://docs.google.com/java/util/AbstractSet.html), [ArrayDeque](http://docs.google.com/java/util/ArrayDeque.html)

public abstract class **AbstractCollection<E>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Collection](http://docs.google.com/java/util/Collection.html)<E>

This class provides a skeletal implementation of the Collection interface, to minimize the effort required to implement this interface.

To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

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

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

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)

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

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/AbstractCollection.html#add(E))([E](http://docs.google.com/java/util/AbstractCollection.html) e)            Ensures that this collection contains the specified element (optional operation). |
| boolean | [**addAll**](http://docs.google.com/java/util/AbstractCollection.html#addAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/AbstractCollection.html)> c)            Adds all of the elements in the specified collection to this collection (optional operation). |
| void | [**clear**](http://docs.google.com/java/util/AbstractCollection.html#clear())()            Removes all of the elements from this collection (optional operation). |
| boolean | [**contains**](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this collection contains the specified element. |
| boolean | [**containsAll**](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Returns true if this collection contains all of the elements in the specified collection. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/AbstractCollection.html#isEmpty())()            Returns true if this collection contains no elements. |
| abstract  [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/AbstractCollection.html)> | [**iterator**](http://docs.google.com/java/util/AbstractCollection.html#iterator())()            Returns an iterator over the elements contained in this collection. |
| boolean | [**remove**](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes a single instance of the specified element from this collection, if it is present (optional operation). |
| boolean | [**removeAll**](http://docs.google.com/java/util/AbstractCollection.html#removeAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Removes all of this collection's elements that are also contained in the specified collection (optional operation). |
| boolean | [**retainAll**](http://docs.google.com/java/util/AbstractCollection.html#retainAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Retains only the elements in this collection that are contained in the specified collection (optional operation). |
| abstract  int | [**size**](http://docs.google.com/java/util/AbstractCollection.html#size())()            Returns the number of elements in this collection. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/AbstractCollection.html#toArray())()            Returns an array containing all of the elements in this collection. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/AbstractCollection.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/AbstractCollection.html#toString())()            Returns a string representation of this collection. |

| **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()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [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.**[**Collection**](http://docs.google.com/java/util/Collection.html) |
| --- |
| [equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Collection.html#hashCode()) |

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

### AbstractCollection

protected **AbstractCollection**()

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

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

### iterator

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

Returns an iterator over the elements contained in this collection.

**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/AbstractCollection.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/AbstractCollection.html)> **Returns:**an iterator over the elements contained in this collection

### size

public abstract int **size**()

**Description copied from interface:** [**Collection**](http://docs.google.com/java/util/Collection.html#size()) Returns the number of elements in this collection. If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**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/AbstractCollection.html)> **Returns:**the number of elements in this collection

### isEmpty

public boolean **isEmpty**()

Returns true if this collection contains no elements.

This implementation returns size() == 0.

**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/AbstractCollection.html)> **Returns:**true if this collection contains no elements

### contains

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

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

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

**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/AbstractCollection.html)> **Parameters:**o - element whose presence in this collection is to be tested **Returns:**true if this collection contains the specified element **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the type of the specified element is incompatible with this collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this collection does not permit null elements (optional)

### toArray

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

Returns an array containing all of the elements in this collection. If this collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

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

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

This implementation returns an array containing all the elements returned by this collection's iterator, in the same order, stored in consecutive elements of the array, starting with index 0. The length of the returned array is equal to the number of elements returned by the iterator, even if the size of this collection changes during iteration, as might happen if the collection permits concurrent modification during iteration. The size method is called only as an optimization hint; the correct result is returned even if the iterator returns a different number of elements.

This method is equivalent to:

List<E> list = new ArrayList<E>(size());  
 for (E e : this)  
 list.add(e);  
 return list.toArray();

**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/AbstractCollection.html)> **Returns:**an array containing all of the elements in this collection

### toArray

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

Returns an array containing all of the elements in this collection; the runtime type of the returned array is that of the specified array. If the collection 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 collection.

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

If this collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

Like the [Collection.toArray()](http://docs.google.com/java/util/Collection.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 collection known to contain only strings. The following code can be used to dump the collection 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().

This implementation returns an array containing all the elements returned by this collection's iterator in the same order, stored in consecutive elements of the array, starting with index 0. If the number of elements returned by the iterator is too large to fit into the specified array, then the elements are returned in a newly allocated array with length equal to the number of elements returned by the iterator, even if the size of this collection changes during iteration, as might happen if the collection permits concurrent modification during iteration. The size method is called only as an optimization hint; the correct result is returned even if the iterator returns a different number of elements.

This method is equivalent to:

List<E> list = new ArrayList<E>(size());  
 for (E e : this)  
 list.add(e);  
 return list.toArray(a);

**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/AbstractCollection.html)> **Parameters:**a - the array into which the elements of this collection are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose. **Returns:**an array containing all of the elements in this collection **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 collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### add

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

Ensures that this collection contains the specified element (optional operation). Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. Collection classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it *must* throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an UnsupportedOperationException.

**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/AbstractCollection.html)> **Parameters:**e - element whose presence in this collection is to be ensured **Returns:**true if this collection changed as a result of the call **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the add operation is not supported by this collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this collection does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the element prevents it from being added to this collection [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to insertion restrictions

### remove

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

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes an element e such that (o==null ? e==null : o.equals(e)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

**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/AbstractCollection.html)> **Parameters:**o - element to be removed from this collection, if present **Returns:**true if an element was removed as a result of this call **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the remove operation is not supported by this collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the type of the specified element is incompatible with this collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this collection does not permit null elements (optional)

### containsAll

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

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

This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

**Specified by:**[containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/AbstractCollection.html)> **Parameters:**c - collection to be checked for containment in this collection **Returns:**true if this collection contains all of the elements in the specified collection **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the types of one or more elements in the specified collection are incompatible with this collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection contains one or more null elements and this collection does not permit null elements (optional), or if the specified collection is null**See Also:**[contains(Object)](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))

### addAll

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

Adds all of the elements in the specified collection to this collection (optional operation). 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 collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

**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/AbstractCollection.html)> **Parameters:**c - collection containing elements to be added to this collection **Returns:**true if this collection 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 collection [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 collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection contains a null element and this collection 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 collection [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if not all the elements can be added at this time due to insertion restrictions**See Also:**[add(Object)](http://docs.google.com/java/util/AbstractCollection.html#add(E))

### removeAll

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

Removes all of this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

**Specified by:**[removeAll](http://docs.google.com/java/util/Collection.html#removeAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/AbstractCollection.html)> **Parameters:**c - collection containing elements to be removed from this collection **Returns:**true if this collection changed as a result of the call **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the removeAll method is not supported by this collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the types of one or more elements in this collection are incompatible with the specified collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if this collection contains one or more null elements and the specified collection does not support null elements (optional), or if the specified collection is null**See Also:**[remove(Object)](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)), [contains(Object)](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))

### retainAll

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

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

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

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

### clear

public void **clear**()

Removes all of the elements from this collection (optional operation). The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the Iterator.remove operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

**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/AbstractCollection.html)> **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the clear operation is not supported by this collection

### toString

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

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

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

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractCollection.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   [**NEXT CLASS**](http://docs.google.com/java/util/AbstractList.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/AbstractCollection.html)    [**NO FRAMES**](http://docs.google.com/AbstractCollection.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#1t3h5sf) | [METHOD](#2s8eyo1) |

[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).