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

## **java.util**

Interface Set<E>

**Type Parameters:**E - the type of elements maintained by this set **All Superinterfaces:** [Collection](http://docs.google.com/java/util/Collection.html)<E>, [Iterable](http://docs.google.com/java/lang/Iterable.html)<E> **All Known Subinterfaces:** [NavigableSet](http://docs.google.com/java/util/NavigableSet.html)<E>, [SortedSet](http://docs.google.com/java/util/SortedSet.html)<E> **All Known Implementing Classes:** [AbstractSet](http://docs.google.com/java/util/AbstractSet.html), [ConcurrentSkipListSet](http://docs.google.com/java/util/concurrent/ConcurrentSkipListSet.html), [CopyOnWriteArraySet](http://docs.google.com/java/util/concurrent/CopyOnWriteArraySet.html), [EnumSet](http://docs.google.com/java/util/EnumSet.html), [HashSet](http://docs.google.com/java/util/HashSet.html), [JobStateReasons](http://docs.google.com/javax/print/attribute/standard/JobStateReasons.html), [LinkedHashSet](http://docs.google.com/java/util/LinkedHashSet.html), [TreeSet](http://docs.google.com/java/util/TreeSet.html)

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

A collection that contains no duplicate elements. More formally, sets contain no pair of elements e1 and e2 such that e1.equals(e2), and at most one null element. As implied by its name, this interface models the mathematical *set* abstraction.

The Set interface places additional stipulations, beyond those inherited from the Collection interface, on the contracts of all constructors and on the contracts of the add, equals and hashCode methods. Declarations for other inherited methods are also included here for convenience. (The specifications accompanying these declarations have been tailored to the Set interface, but they do not contain any additional stipulations.)

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set. A special case of this prohibition is that it is not permissible for a set to contain itself as an element.

Some set implementations have restrictions on the elements that they may contain. For example, some implementations prohibit null elements, and some have restrictions on the types of their elements. Attempting to add an ineligible element throws an unchecked exception, typically NullPointerException or ClassCastException. Attempting to query the presence of an ineligible element may throw an exception, or it may simply return false; some implementations will exhibit the former behavior and some will exhibit the latter. More generally, attempting an operation on an ineligible element whose completion would not result in the insertion of an ineligible element into the set may throw an exception or it may succeed, at the option of the implementation. Such exceptions are marked as "optional" in the specification for this interface.

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), [SortedSet](http://docs.google.com/java/util/SortedSet.html), [HashSet](http://docs.google.com/java/util/HashSet.html), [TreeSet](http://docs.google.com/java/util/TreeSet.html), [AbstractSet](http://docs.google.com/java/util/AbstractSet.html), [Collections.singleton(java.lang.Object)](http://docs.google.com/java/util/Collections.html#singleton(T)), [Collections.EMPTY\_SET](http://docs.google.com/java/util/Collections.html#EMPTY_SET)

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/Set.html#add(E))([E](http://docs.google.com/java/util/Set.html) e)            Adds the specified element to this set if it is not already present (optional operation). |
| boolean | [**addAll**](http://docs.google.com/java/util/Set.html#addAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/Set.html)> c)            Adds all of the elements in the specified collection to this set if they're not already present (optional operation). |
| void | [**clear**](http://docs.google.com/java/util/Set.html#clear())()            Removes all of the elements from this set (optional operation). |
| boolean | [**contains**](http://docs.google.com/java/util/Set.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this set contains the specified element. |
| boolean | [**containsAll**](http://docs.google.com/java/util/Set.html#containsAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Returns true if this set contains all of the elements of the specified collection. |
| boolean | [**equals**](http://docs.google.com/java/util/Set.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Compares the specified object with this set for equality. |
| int | [**hashCode**](http://docs.google.com/java/util/Set.html#hashCode())()            Returns the hash code value for this set. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/Set.html#isEmpty())()            Returns true if this set contains no elements. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/Set.html)> | [**iterator**](http://docs.google.com/java/util/Set.html#iterator())()            Returns an iterator over the elements in this set. |
| boolean | [**remove**](http://docs.google.com/java/util/Set.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the specified element from this set if it is present (optional operation). |
| boolean | [**removeAll**](http://docs.google.com/java/util/Set.html#removeAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Removes from this set all of its elements that are contained in the specified collection (optional operation). |
| boolean | [**retainAll**](http://docs.google.com/java/util/Set.html#retainAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Retains only the elements in this set that are contained in the specified collection (optional operation). |
| int | [**size**](http://docs.google.com/java/util/Set.html#size())()            Returns the number of elements in this set (its cardinality). |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/Set.html#toArray())()            Returns an array containing all of the elements in this set. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/Set.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this set; the runtime type of the returned array is that of the specified array. |

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

### size

int **size**()

Returns the number of elements in this set (its cardinality). If this set 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/Set.html)> **Returns:**the number of elements in this set (its cardinality)

### isEmpty

boolean **isEmpty**()

Returns true if this set contains no elements.

**Specified by:**[isEmpty](http://docs.google.com/java/util/Collection.html#isEmpty()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/Set.html)> **Returns:**true if this set contains no elements

### contains

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

Returns true if this set contains the specified element. More formally, returns true if and only if this set contains an 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/Set.html)> **Parameters:**o - element whose presence in this set is to be tested **Returns:**true if this set 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 set (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this set does not permit null elements (optional)

### iterator

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

Returns an iterator over the elements in this set. The elements are returned in no particular order (unless this set is an instance of some class that provides a guarantee).

**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/Set.html)>**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/Set.html)> **Returns:**an iterator over the elements in this set

### toArray

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

Returns an array containing all of the elements in this set. If this set 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 set. (In other words, this method must allocate a new array even if this set 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.

**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/Set.html)> **Returns:**an array containing all the elements in this set

### toArray

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

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

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

If this set 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 [toArray()](http://docs.google.com/java/util/Set.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 set known to contain only strings. The following code can be used to dump the set 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/Set.html)> **Parameters:**a - the array into which the elements of this set 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 the elements in this set **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 set [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### add

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

Adds the specified element to this set if it is not already present (optional operation). More formally, adds the specified element e to this set if the set contains no element e2 such that (e==null ? e2==null : e.equals(e2)). If this set already contains the element, the call leaves the set unchanged and returns false. In combination with the restriction on constructors, this ensures that sets never contain duplicate elements.

The stipulation above does not imply that sets must accept all elements; sets may refuse to add any particular element, including null, and throw an exception, as described in the specification for [Collection.add](http://docs.google.com/java/util/Collection.html#add(E)). Individual set implementations should clearly document any restrictions on the elements that they may contain.

**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/Set.html)> **Parameters:**e - element to be added to this set **Returns:**true if this set did not already contain the specified element **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the add operation is not supported by this set [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this set [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this set 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 set

### remove

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

Removes the specified element from this set if it is present (optional operation). More formally, removes an element e such that (o==null ? e==null : o.equals(e)), if this set contains such an element. Returns true if this set contained the element (or equivalently, if this set changed as a result of the call). (This set will not contain the element once the call returns.)

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

### containsAll

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

Returns true if this set contains all of the elements of the specified collection. If the specified collection is also a set, this method returns true if it is a *subset* of this set.

**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/Set.html)> **Parameters:**c - collection to be checked for containment in this set **Returns:**true if this set contains all of the elements of 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 set (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection contains one or more null elements and this set does not permit null elements (optional), or if the specified collection is null**See Also:**[contains(Object)](http://docs.google.com/java/util/Set.html#contains(java.lang.Object))

### addAll

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

Adds all of the elements in the specified collection to this set if they're not already present (optional operation). If the specified collection is also a set, the addAll operation effectively modifies this set so that its value is the *union* of the two sets. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**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/Set.html)> **Parameters:**c - collection containing elements to be added to this set **Returns:**true if this set 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 set [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 set [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection contains one or more null elements and this set 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 set**See Also:**[add(Object)](http://docs.google.com/java/util/Set.html#add(E))

### retainAll

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

Retains only the elements in this set that are contained in the specified collection (optional operation). In other words, removes from this set all of its elements that are not contained in the specified collection. If the specified collection is also a set, this operation effectively modifies this set so that its value is the *intersection* of the two sets.

**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/Set.html)> **Parameters:**c - collection containing elements to be retained in this set **Returns:**true if this set 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 set [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this set is incompatible with the specified collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if this set contains a null element 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/Set.html#remove(java.lang.Object))

### removeAll

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

Removes from this set all of its elements that are contained in the specified collection (optional operation). If the specified collection is also a set, this operation effectively modifies this set so that its value is the *asymmetric set difference* of the two sets.

**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/Set.html)> **Parameters:**c - collection containing elements to be removed from this set **Returns:**true if this set changed as a result of the call **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the removeAll operation is not supported by this set [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this set is incompatible with the specified collection (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if this set contains a null element 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/Set.html#remove(java.lang.Object)), [contains(Object)](http://docs.google.com/java/util/Set.html#contains(java.lang.Object))

### clear

void **clear**()

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

**Specified by:**[clear](http://docs.google.com/java/util/Collection.html#clear()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/Set.html)> **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if the clear method is not supported by this set

### equals

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

Compares the specified object with this set for equality. Returns true if the specified object is also a set, the two sets have the same size, and every member of the specified set is contained in this set (or equivalently, every member of this set is contained in the specified set). This definition ensures that the equals method works properly across different implementations of the set interface.

**Specified by:**[equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/Set.html)>**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**o - object to be compared for equality with this set **Returns:**true if the specified object is equal to this set**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

int **hashCode**()

Returns the hash code value for this set. The hash code of a set is defined to be the sum of the hash codes of the elements in the set, where the hash code of a null element is defined to be zero. This ensures that s1.equals(s2) implies that s1.hashCode()==s2.hashCode() for any two sets s1 and s2, as required by the general contract of [Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()).

**Specified by:**[hashCode](http://docs.google.com/java/util/Collection.html#hashCode()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/Set.html)>**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**the hash code value for this set**See Also:**[Object.equals(Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [equals(Object)](http://docs.google.com/java/util/Set.html#equals(java.lang.Object))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Set.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/ServiceLoader.html)   [**NEXT CLASS**](http://docs.google.com/java/util/SimpleTimeZone.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Set.html)    [**NO FRAMES**](http://docs.google.com/Set.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).