Collection

The root interface in the *collection hierarchy*. A collection represents a group of objects, known as its *elements*. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. The JDK does not provide any *direct* implementations of this interface: it provides implementations of more specific subinterfaces like Set and List. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

*Bags* or *multisets* (unordered collections that may contain duplicate elements) should implement this interface directly.

All general-purpose Collection implementation classes (which typically implement Collection indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type Collection, which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose Collection implementations in the Java platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw UnsupportedOperationException if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an UnsupportedOperationException if the invocation would have no effect on the collection. For example, invoking the [addAll(Collection)](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html" \l "addAll-java.util.Collection-) method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Some collection 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 collection 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.

It is up to each collection to determine its own synchronization policy. In the absence of a stronger guarantee by the implementation, undefined behavior may result from the invocation of any method on a collection that is being mutated by another thread; this includes direct invocations, passing the collection to a method that might perform invocations, and using an existing iterator to examine the collection.

Some collection operations which perform recursive traversal of the collection may fail with an exception for self-referential instances where the collection directly or indirectly contains itself. This includes the clone(), equals(), hashCode() and toString() methods. Implementations may optionally handle the self-referential scenario, however most current implementations do not do so.

This interface is a member of the [Java Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/index.html).

**Implementation Requirements:**

The default method implementations (inherited or otherwise) do not apply any synchronization protocol. If a Collection implementation has a specific synchronization protocol, then it must override default implementations to apply that protocol.

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Abstract Methods**](javascript:show(4);)[**Default Methods**](javascript:show(16);) | |
| **Modifier and Type** | **Method and Description** |
| boolean | [**add**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#add-E-)([**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html) e)  Ensures that this collection contains the specified element (optional operation). |
| boolean | [**addAll**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#addAll-java.util.Collection-)([**Collection**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)<? extends [**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> c)  Adds all of the elements in the specified collection to this collection (optional operation). |
| void | [**clear**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#clear--)()  Removes all of the elements from this collection (optional operation). |
| boolean | [**contains**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#contains-java.lang.Object-)([**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) o)  Returns true if this collection contains the specified element. |
| boolean | [**containsAll**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#containsAll-java.util.Collection-)([**Collection**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)<?> c)  Returns true if this collection contains all of the elements in the specified collection. |
| boolean | [**equals**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#equals-java.lang.Object-)([**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) o)  Compares the specified object with this collection for equality. |
| int | [**hashCode**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#hashCode--)()  Returns the hash code value for this collection. |
| boolean | [**isEmpty**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#isEmpty--)()  Returns true if this collection contains no elements. |
| [**Iterator**](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html)<[**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> | [**iterator**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#iterator--)()  Returns an iterator over the elements in this collection. |
| default [**Stream**](https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html)<[**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> | [**parallelStream**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#parallelStream--)()  Returns a possibly parallel Stream with this collection as its source. |
| boolean | [**remove**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#remove-java.lang.Object-)([**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) o)  Removes a single instance of the specified element from this collection, if it is present (optional operation). |
| boolean | [**removeAll**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#removeAll-java.util.Collection-)([**Collection**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)<?> c)  Removes all of this collection's elements that are also contained in the specified collection (optional operation). |
| default boolean | [**removeIf**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#removeIf-java.util.function.Predicate-)([**Predicate**](https://docs.oracle.com/javase/8/docs/api/java/util/function/Predicate.html)<? super [**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> filter)  Removes all of the elements of this collection that satisfy the given predicate. |
| boolean | [**retainAll**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#retainAll-java.util.Collection-)([**Collection**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)<?> c)  Retains only the elements in this collection that are contained in the specified collection (optional operation). |
| int | [**size**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#size--)()  Returns the number of elements in this collection. |
| default **[Spliterator](https://docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html" \o "interface in java.util)**<[**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> | [**spliterator**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#spliterator--)()  Creates a **[Spliterator](https://docs.oracle.com/javase/8/docs/api/java/util/Spliterator.html" \o "interface in java.util)** over the elements in this collection. |
| default [**Stream**](https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html)<[**E**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> | [**stream**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#stream--)()  Returns a sequential Stream with this collection as its source. |
| [**Object**](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)[] | [**toArray**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#toArray--)()  Returns an array containing all of the elements in this collection. |
| <T> T[] | [**toArray**](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#toArray-T:A-)(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. |

### Methods inherited from interface java.lang.[Iterable](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html)

### *Method Detail*

#### size

int size()

Returns the number of elements in this collection. If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**Returns:**

the number of elements in this collection

#### isEmpty

boolean isEmpty()

Returns true if this collection contains no elements.

**Returns:**

true if this collection contains no elements

#### contains

boolean contains([Object](https://docs.oracle.com/javase/8/docs/api/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)).

**Parameters:**

o - element whose presence in this collection is to be tested

**Returns:**

true if this collection contains the specified element

**Throws:**

[ClassCastException](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCastException.html) - if the type of the specified element is incompatible with this collection ([optional](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#optional-restrictions))

[NullPointerException](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html) - if the specified element is null and this collection does not permit null elements ([optional](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html#optional-restrictions))

#### iterator

[Iterator](https://docs.oracle.com/javase/8/docs/api/java/util/Iterator.html)<[E](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)> iterator()

Returns an iterator over the elements in this collection. There are no guarantees concerning the order in which the elements are returned (unless this collection is an instance of some class that provides a guarantee).

**Specified by:**

[iterator](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html#iterator--) in interface [Iterable](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html" \o "interface in java.lang)<[E](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html)>

**Returns:**

an Iterator over the elements in this collection

#### toArray

[Object](https://docs.oracle.com/javase/8/docs/api/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.

**Returns:**

an array containing all of the elements in this collection

#### toArray

<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 [toArray()](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.html" \l "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().

**Type Parameters:**

T - the runtime type of the array to contain the collection

**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](https://docs.oracle.com/javase/8/docs/api/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](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html) - if the specified array is null

#### add

boolean add([E](https://docs.oracle.com/javase/8/docs/api/java/util/Collection.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.

**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](https://docs.oracle.com/javase/8/docs/api/java/lang/UnsupportedOperationException.html) - if the add operation is not supported by this collection

[ClassCastException](https://docs.oracle.com/javase/8/docs/api/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this collection

[NullPointerException](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html) - if the specified element is null and this collection does not permit null elements

[IllegalArgumentException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalArgumentException.html) - if some property of the element prevents it from being added to this collection

[IllegalStateException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to insertion restrictions