| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ConcurrentLinkedQueue.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/concurrent/ConcurrentHashMap.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ConcurrentMap.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ConcurrentLinkedQueue.html)    [**NO FRAMES**](http://docs.google.com/ConcurrentLinkedQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **java.util.concurrent**

Class ConcurrentLinkedQueue<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.AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<E>  
 **java.util.concurrent.ConcurrentLinkedQueue<E>**

**Type Parameters:**E - the type of elements held in this collection **All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Collection](http://docs.google.com/java/util/Collection.html)<E>, [Queue](http://docs.google.com/java/util/Queue.html)<E>

public class **ConcurrentLinkedQueue<E>**extends [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<E>implements [Queue](http://docs.google.com/java/util/Queue.html)<E>, [Serializable](http://docs.google.com/java/io/Serializable.html)

An unbounded thread-safe [queue](http://docs.google.com/java/util/Queue.html) based on linked nodes. This queue orders elements FIFO (first-in-first-out). The *head* of the queue is that element that has been on the queue the longest time. The *tail* of the queue is that element that has been on the queue the shortest time. New elements are inserted at the tail of the queue, and the queue retrieval operations obtain elements at the head of the queue. A ConcurrentLinkedQueue is an appropriate choice when many threads will share access to a common collection. This queue does not permit null elements.

This implementation employs an efficient "wait-free" algorithm based on one described in  [Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms](http://www.cs.rochester.edu/u/michael/PODC96.html) by Maged M. Michael and Michael L. Scott.

Beware that, unlike in most collections, the size method is *NOT* a constant-time operation. Because of the asynchronous nature of these queues, determining the current number of elements requires a traversal of the elements.

This class and its iterator implement all of the *optional* methods of the [Collection](http://docs.google.com/java/util/Collection.html) and [Iterator](http://docs.google.com/java/util/Iterator.html) interfaces.

Memory consistency effects: As with other concurrent collections, actions in a thread prior to placing an object into a ConcurrentLinkedQueue [*happen-before*](http://docs.google.com/package-summary.html#MemoryVisibility) actions subsequent to the access or removal of that element from the ConcurrentLinkedQueue in another thread.

This class is a member of the  [Java Collections Framework](http://docs.google.com/technotes/guides/collections/index.html).

**Since:** 1.5 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.util.concurrent.ConcurrentLinkedQueue)

| **Constructor Summary** | |
| --- | --- |
| [**ConcurrentLinkedQueue**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#ConcurrentLinkedQueue())()            Creates a ConcurrentLinkedQueue that is initially empty. |
| [**ConcurrentLinkedQueue**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#ConcurrentLinkedQueue(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> c)            Creates a ConcurrentLinkedQueue initially containing the elements of the given collection, added in traversal order of the collection's iterator. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#add(E))([E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) e)            Inserts the specified element at the tail of this queue. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this queue contains the specified element. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#isEmpty())()            Returns true if this queue contains no elements. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#iterator())()            Returns an iterator over the elements in this queue in proper sequence. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#offer(E))([E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) e)            Inserts the specified element at the tail of this queue. |
| [E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) | [**peek**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#peek())()            Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty. |
| [E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#poll())()            Retrieves and removes the head of this queue, or returns null if this queue is empty. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.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 queue, if it is present. |
| int | [**size**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#size())()            Returns the number of elements in this queue. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#toArray())()            Returns an array containing all of the elements in this queue, in proper sequence. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this queue, in proper sequence; the runtime type of the returned array is that of the specified array. |

| **Methods inherited from class java.util.**[**AbstractQueue**](http://docs.google.com/java/util/AbstractQueue.html) |
| --- |
| [addAll](http://docs.google.com/java/util/AbstractQueue.html#addAll(java.util.Collection)), [clear](http://docs.google.com/java/util/AbstractQueue.html#clear()), [element](http://docs.google.com/java/util/AbstractQueue.html#element()), [remove](http://docs.google.com/java/util/AbstractQueue.html#remove()) |

| **Methods inherited from class java.util.**[**AbstractCollection**](http://docs.google.com/java/util/AbstractCollection.html) |
| --- |
| [containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)), [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)), [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()), [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.**[**Queue**](http://docs.google.com/java/util/Queue.html) |
| --- |
| [element](http://docs.google.com/java/util/Queue.html#element()), [remove](http://docs.google.com/java/util/Queue.html#remove()) |

| **Methods inherited from interface java.util.**[**Collection**](http://docs.google.com/java/util/Collection.html) |
| --- |
| [addAll](http://docs.google.com/java/util/Collection.html#addAll(java.util.Collection)), [clear](http://docs.google.com/java/util/Collection.html#clear()), [containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)), [equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Collection.html#hashCode()), [removeAll](http://docs.google.com/java/util/Collection.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/Collection.html#retainAll(java.util.Collection)) |

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

### ConcurrentLinkedQueue

public **ConcurrentLinkedQueue**()

Creates a ConcurrentLinkedQueue that is initially empty.

### ConcurrentLinkedQueue

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

Creates a ConcurrentLinkedQueue initially containing the elements of the given collection, added in traversal order of the collection's iterator.

**Parameters:**c - the collection of elements to initially contain **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection or any of its elements are null

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

### add

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

Inserts the specified element at the tail of this queue.

**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/concurrent/ConcurrentLinkedQueue.html)>**Specified by:**[add](http://docs.google.com/java/util/Queue.html#add(E)) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[add](http://docs.google.com/java/util/AbstractQueue.html#add(E)) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Parameters:**e - the element to add **Returns:**true (as specified by [Collection.add(E)](http://docs.google.com/java/util/Collection.html#add(E))) **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### offer

public boolean **offer**([E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) e)

Inserts the specified element at the tail of this queue.

**Specified by:**[offer](http://docs.google.com/java/util/Queue.html#offer(E)) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Parameters:**e - the element to add **Returns:**true (as specified by [Queue.offer(E)](http://docs.google.com/java/util/Queue.html#offer(E))) **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### poll

public [E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) **poll**()

**Description copied from interface:** [**Queue**](http://docs.google.com/java/util/Queue.html#poll()) Retrieves and removes the head of this queue, or returns null if this queue is empty.

**Specified by:**[poll](http://docs.google.com/java/util/Queue.html#poll()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### peek

public [E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html) **peek**()

**Description copied from interface:** [**Queue**](http://docs.google.com/java/util/Queue.html#peek()) Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

**Specified by:**[peek](http://docs.google.com/java/util/Queue.html#peek()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### isEmpty

public boolean **isEmpty**()

Returns true if this queue 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/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**true if this queue contains no elements

### size

public int **size**()

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

Beware that, unlike in most collections, this method is *NOT* a constant-time operation. Because of the asynchronous nature of these queues, determining the current number of elements requires an O(n) traversal.

**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/concurrent/ConcurrentLinkedQueue.html)>**Specified by:**[size](http://docs.google.com/java/util/AbstractCollection.html#size()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**the number of elements in this queue

### contains

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

Returns true if this queue contains the specified element. More formally, returns true if and only if this queue contains at least one element e such that 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/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Parameters:**o - object to be checked for containment in this queue **Returns:**true if this queue contains the specified element

### remove

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

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

**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/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[remove](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Parameters:**o - element to be removed from this queue, if present **Returns:**true if this queue changed as a result of the call

### toArray

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

Returns an array containing all of the elements in this queue, in proper sequence.

The returned array will be "safe" in that no references to it are maintained by this queue. (In other words, this method must allocate a new 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/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**an array containing all of the elements in this queue

### toArray

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

Returns an array containing all of the elements in this queue, in proper sequence; the runtime type of the returned array is that of the specified array. If the queue 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 queue.

If this queue fits in the specified array with room to spare (i.e., the array has more elements than this queue), the element in the array immediately following the end of the queue is set to null.

Like the [toArray()](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.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 queue known to contain only strings. The following code can be used to dump the queue 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/concurrent/ConcurrentLinkedQueue.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray(T%5B%5D)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Parameters:**a - the array into which the elements of the queue 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 queue **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 queue [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### iterator

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

Returns an iterator over the elements in this queue in proper sequence. The returned iterator is a "weakly consistent" iterator that will never throw [ConcurrentModificationException](http://docs.google.com/java/util/ConcurrentModificationException.html), and guarantees to traverse elements as they existed upon construction of the iterator, and may (but is not guaranteed to) reflect any modifications subsequent to construction.

**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/concurrent/ConcurrentLinkedQueue.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/concurrent/ConcurrentLinkedQueue.html)>**Specified by:**[iterator](http://docs.google.com/java/util/AbstractCollection.html#iterator()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html)> **Returns:**an iterator over the elements in this queue in proper sequence

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ConcurrentLinkedQueue.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/concurrent/ConcurrentHashMap.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ConcurrentMap.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ConcurrentLinkedQueue.html)    [**NO FRAMES**](http://docs.google.com/ConcurrentLinkedQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [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).