| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LinkedBlockingQueue.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/LinkedBlockingDeque.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/LinkedBlockingQueue.html)    [**NO FRAMES**](http://docs.google.com/LinkedBlockingQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

## **java.util.concurrent**

Class LinkedBlockingQueue<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.LinkedBlockingQueue<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>, [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>, [Queue](http://docs.google.com/java/util/Queue.html)<E>

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

An optionally-bounded [blocking queue](http://docs.google.com/java/util/concurrent/BlockingQueue.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. Linked queues typically have higher throughput than array-based queues but less predictable performance in most concurrent applications.

The optional capacity bound constructor argument serves as a way to prevent excessive queue expansion. The capacity, if unspecified, is equal to [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE). Linked nodes are dynamically created upon each insertion unless this would bring the queue above capacity.

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.

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.LinkedBlockingQueue)

| **Constructor Summary** | |
| --- | --- |
| [**LinkedBlockingQueue**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#LinkedBlockingQueue())()            Creates a LinkedBlockingQueue with a capacity of [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE). |
| [**LinkedBlockingQueue**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#LinkedBlockingQueue(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> c)            Creates a LinkedBlockingQueue with a capacity of [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE), initially containing the elements of the given collection, added in traversal order of the collection's iterator. |
| [**LinkedBlockingQueue**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#LinkedBlockingQueue(int))(int capacity)            Creates a LinkedBlockingQueue with the given (fixed) capacity. |

| **Method Summary** | |
| --- | --- |
| void | [**clear**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#clear())()            Atomically removes all of the elements from this queue. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#drainTo(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> c)            Removes all available elements from this queue and adds them to the given collection. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#drainTo(java.util.Collection,%20int))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> c, int maxElements)            Removes at most the given number of available elements from this queue and adds them to the given collection. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#iterator())()            Returns an iterator over the elements in this queue in proper sequence. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#offer(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) e)            Inserts the specified element at the tail of this queue if it is possible to do so immediately without exceeding the queue's capacity, returning true upon success and false if this queue is full. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element at the tail of this queue, waiting if necessary up to the specified wait time for space to become available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) | [**peek**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#poll())()            Retrieves and removes the head of this queue, or returns null if this queue is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#poll(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available. |
| void | [**put**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#put(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) e)            Inserts the specified element at the tail of this queue, waiting if necessary for space to become available. |
| int | [**remainingCapacity**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#remainingCapacity())()            Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.html#size())()            Returns the number of elements in this queue. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) | [**take**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#take())()            Retrieves and removes the head of this queue, waiting if necessary until an element becomes available. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.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. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html#toString())()            Returns a string representation of this collection. |

| **Methods inherited from class java.util.**[**AbstractQueue**](http://docs.google.com/java/util/AbstractQueue.html) |
| --- |
| [add](http://docs.google.com/java/util/AbstractQueue.html#add(E)), [addAll](http://docs.google.com/java/util/AbstractQueue.html#addAll(java.util.Collection)), [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) |
| --- |
| [contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)), [containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)), [isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()), [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)) |

| **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.concurrent.**[**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html) |
| --- |
| [add](http://docs.google.com/java/util/concurrent/BlockingQueue.html#add(E)), [contains](http://docs.google.com/java/util/concurrent/BlockingQueue.html#contains(java.lang.Object)) |

| **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)), [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()), [isEmpty](http://docs.google.com/java/util/Collection.html#isEmpty()), [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** |
| --- |

### LinkedBlockingQueue

public **LinkedBlockingQueue**()

Creates a LinkedBlockingQueue with a capacity of [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE).

### LinkedBlockingQueue

public **LinkedBlockingQueue**(int capacity)

Creates a LinkedBlockingQueue with the given (fixed) capacity.

**Parameters:**capacity - the capacity of this queue **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if capacity is not greater than zero

### LinkedBlockingQueue

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

Creates a LinkedBlockingQueue with a capacity of [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE), 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** |
| --- |

### size

public int **size**()

Returns the number of elements in this queue.

**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/LinkedBlockingQueue.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/LinkedBlockingQueue.html)> **Returns:**the number of elements in this queue

### remainingCapacity

public int **remainingCapacity**()

Returns the number of additional elements that this queue can ideally (in the absence of memory or resource constraints) accept without blocking. This is always equal to the initial capacity of this queue less the current size of this queue.

Note that you *cannot* always tell if an attempt to insert an element will succeed by inspecting remainingCapacity because it may be the case that another thread is about to insert or remove an element.

**Specified by:**[remainingCapacity](http://docs.google.com/java/util/concurrent/BlockingQueue.html#remainingCapacity()) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Returns:**the remaining capacity

### put

public void **put**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) e)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Inserts the specified element at the tail of this queue, waiting if necessary for space to become available.

**Specified by:**[put](http://docs.google.com/java/util/concurrent/BlockingQueue.html#put(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Parameters:**e - the element to add **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [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/LinkedBlockingQueue.html) e,  
 long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Inserts the specified element at the tail of this queue, waiting if necessary up to the specified wait time for space to become available.

**Specified by:**[offer](http://docs.google.com/java/util/concurrent/BlockingQueue.html#offer(E,%20long,%20java.util.concurrent.TimeUnit)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Parameters:**e - the element to addtimeout - how long to wait before giving up, in units of unitunit - a TimeUnit determining how to interpret the timeout parameter **Returns:**true if successful, or false if the specified waiting time elapses before space is available. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [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/LinkedBlockingQueue.html) e)

Inserts the specified element at the tail of this queue if it is possible to do so immediately without exceeding the queue's capacity, returning true upon success and false if this queue is full. When using a capacity-restricted queue, this method is generally preferable to method [add](http://docs.google.com/java/util/concurrent/BlockingQueue.html#add(E)), which can fail to insert an element only by throwing an exception.

**Specified by:**[offer](http://docs.google.com/java/util/concurrent/BlockingQueue.html#offer(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)>**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/LinkedBlockingQueue.html)> **Parameters:**e - the element to add **Returns:**true if the element was added to this queue, else false **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### take

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) **take**()  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#take()) Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

**Specified by:**[take](http://docs.google.com/java/util/concurrent/BlockingQueue.html#take()) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Returns:**the head of this queue **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### poll

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) **poll**(long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#poll(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

**Specified by:**[poll](http://docs.google.com/java/util/concurrent/BlockingQueue.html#poll(long,%20java.util.concurrent.TimeUnit)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Parameters:**timeout - how long to wait before giving up, in units of unitunit - a TimeUnit determining how to interpret the timeout parameter **Returns:**the head of this queue, or null if the specified waiting time elapses before an element is available **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### poll

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### peek

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### 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/LinkedBlockingQueue.html)>**Specified by:**[remove](http://docs.google.com/java/util/concurrent/BlockingQueue.html#remove(java.lang.Object)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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

### toString

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

**Description copied from class:** [**AbstractCollection**](http://docs.google.com/java/util/AbstractCollection.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/util/AbstractCollection.html#toString()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Returns:**a string representation of this collection

### clear

public void **clear**()

Atomically removes all of the elements from this queue. The queue 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/concurrent/LinkedBlockingQueue.html)>**Overrides:**[clear](http://docs.google.com/java/util/AbstractQueue.html#clear()) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)>

### drainTo

public int **drainTo**([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> c)

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection)) Removes all available elements from this queue and adds them to the given collection. This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Specified by:**[drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Parameters:**c - the collection to transfer elements into **Returns:**the number of elements transferred **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if addition of elements is not supported by the specified collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this queue prevents it from being added to the specified collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

### drainTo

public int **drainTo**([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> c,  
 int maxElements)

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection,%20int)) Removes at most the given number of available elements from this queue and adds them to the given collection. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Specified by:**[drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection,%20int)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html)> **Parameters:**c - the collection to transfer elements intomaxElements - the maximum number of elements to transfer **Returns:**the number of elements transferred **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if addition of elements is not supported by the specified collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this queue prevents it from being added to the specified collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

### iterator

public [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingQueue.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/LinkedBlockingDeque.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/LinkedBlockingQueue.html)    [**NO FRAMES**](http://docs.google.com/LinkedBlockingQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

[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).