| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LinkedBlockingDeque.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/FutureTask.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/LinkedBlockingDeque.html)    [**NO FRAMES**](http://docs.google.com/LinkedBlockingDeque.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#lnxbz9) |

## **java.util.concurrent**

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

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

An optionally-bounded [blocking deque](http://docs.google.com/java/util/concurrent/BlockingDeque.html) based on linked nodes.

The optional capacity bound constructor argument serves as a way to prevent excessive 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 deque above capacity.

Most operations run in constant time (ignoring time spent blocking). Exceptions include [remove](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#remove(java.lang.Object)), [removeFirstOccurrence](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeFirstOccurrence(java.lang.Object)), [removeLastOccurrence](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeLastOccurrence(java.lang.Object)), [contains](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#contains(java.lang.Object)), [iterator.remove()](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#iterator()), and the bulk operations, all of which run in linear time.

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

| **Constructor Summary** | |
| --- | --- |
| [**LinkedBlockingDeque**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#LinkedBlockingDeque())()            Creates a LinkedBlockingDeque with a capacity of [Integer.MAX\_VALUE](http://docs.google.com/java/lang/Integer.html#MAX_VALUE). |
| [**LinkedBlockingDeque**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#LinkedBlockingDeque(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> c)            Creates a LinkedBlockingDeque 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. |
| [**LinkedBlockingDeque**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#LinkedBlockingDeque(int))(int capacity)            Creates a LinkedBlockingDeque with the given (fixed) capacity. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#add(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the end of this deque unless it would violate capacity restrictions. |
| void | [**addFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#addFirst(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an IllegalStateException if no space is currently available. |
| void | [**addLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#addLast(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an IllegalStateException if no space is currently available. |
| void | [**clear**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#clear())()            Atomically removes all of the elements from this deque. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this deque contains the specified element. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> | [**descendingIterator**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#descendingIterator())()            Returns an iterator over the elements in this deque in reverse sequential order. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#drainTo(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html#drainTo(java.util.Collection,%20int))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> c, int maxElements)            Removes at most the given number of available elements from this queue and adds them to the given collection. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**element**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#element())()            Retrieves, but does not remove, the head of the queue represented by this deque. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**getFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#getFirst())()            Retrieves, but does not remove, the first element of this deque. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**getLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#getLast())()            Retrieves, but does not remove, the last element of this deque. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#iterator())()            Returns an iterator over the elements in this deque in proper sequence. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offer(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque) if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque), waiting up to the specified wait time if necessary for space to become available. |
| boolean | [**offerFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offerFirst(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. |
| boolean | [**offerFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offerFirst(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element at the front of this deque, waiting up to the specified wait time if necessary for space to become available. |
| boolean | [**offerLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offerLast(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. |
| boolean | [**offerLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element at the end of this deque, waiting up to the specified wait time if necessary for space to become available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**peek**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#peek())()            Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**peekFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#peekFirst())()            Retrieves, but does not remove, the first element of this deque, or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**peekLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#peekLast())()            Retrieves, but does not remove, the last element of this deque, or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**poll**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#poll())()            Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**poll**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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 the queue represented by this deque (in other words, the first element of this deque), waiting up to the specified wait time if necessary for an element to become available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**pollFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#pollFirst())()            Retrieves and removes the first element of this deque, or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**pollFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Retrieves and removes the first element of this deque, waiting up to the specified wait time if necessary for an element to become available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**pollLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#pollLast())()            Retrieves and removes the last element of this deque, or returns null if this deque is empty. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**pollLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#pollLast(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Retrieves and removes the last element of this deque, waiting up to the specified wait time if necessary for an element to become available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**pop**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#pop())()            Pops an element from the stack represented by this deque. |
| void | [**push**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#push(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Pushes an element onto the stack represented by this deque. |
| void | [**put**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#put(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque), waiting if necessary for space to become available. |
| void | [**putFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#putFirst(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the front of this deque, waiting if necessary for space to become available. |
| void | [**putLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#putLast(E))([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)            Inserts the specified element at the end of this deque, waiting if necessary for space to become available. |
| int | [**remainingCapacity**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#remainingCapacity())()            Returns the number of additional elements that this deque can ideally (in the absence of memory or resource constraints) accept without blocking. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**remove**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#remove())()            Retrieves and removes the head of the queue represented by this deque. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the first occurrence of the specified element from this deque. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**removeFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeFirst())()            Retrieves and removes the first element of this deque. |
| boolean | [**removeFirstOccurrence**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeFirstOccurrence(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the first occurrence of the specified element from this deque. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**removeLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeLast())()            Retrieves and removes the last element of this deque. |
| boolean | [**removeLastOccurrence**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeLastOccurrence(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes the last occurrence of the specified element from this deque. |
| int | [**size**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#size())()            Returns the number of elements in this deque. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**take**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#take())()            Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), waiting if necessary until an element becomes available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**takeFirst**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#takeFirst())()            Retrieves and removes the first element of this deque, waiting if necessary until an element becomes available. |
| [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) | [**takeLast**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#takeLast())()            Retrieves and removes the last element of this deque, 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/LinkedBlockingDeque.html#toArray())()            Returns an array containing all of the elements in this deque, in proper sequence (from first to last element). |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this deque, 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/LinkedBlockingDeque.html#toString())()            Returns a string representation of this collection. |

| **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)) |

| **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)), [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.**[**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** |
| --- |

### LinkedBlockingDeque

public **LinkedBlockingDeque**()

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

### LinkedBlockingDeque

public **LinkedBlockingDeque**(int capacity)

Creates a LinkedBlockingDeque with the given (fixed) capacity.

**Parameters:**capacity - the capacity of this deque **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if capacity is less than 1

### LinkedBlockingDeque

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

Creates a LinkedBlockingDeque 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** |
| --- |

### addFirst

public void **addFirst**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E)) Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an IllegalStateException if no space is currently available. When using a capacity-restricted deque, it is generally preferable to use [offerFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E)).

**Specified by:**[addFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[addFirst](http://docs.google.com/java/util/Deque.html#addFirst(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to capacity restrictions [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### addLast

public void **addLast**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addLast(E)) Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions, throwing an IllegalStateException if no space is currently available. When using a capacity-restricted deque, it is generally preferable to use [offerLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E)).

**Specified by:**[addLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addLast(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[addLast](http://docs.google.com/java/util/Deque.html#addLast(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to capacity restrictions [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### offerFirst

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E)) Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. When using a capacity-restricted deque, this method is generally preferable to the [addFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E)) method, which can fail to insert an element only by throwing an exception.

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

### offerLast

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E)) Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. When using a capacity-restricted deque, this method is generally preferable to the [addLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addLast(E)) method, which can fail to insert an element only by throwing an exception.

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

### putFirst

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putFirst(E)) Inserts the specified element at the front of this deque, waiting if necessary for space to become available.

**Specified by:**[putFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putFirst(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### putLast

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putLast(E)) Inserts the specified element at the end of this deque, waiting if necessary for space to become available.

**Specified by:**[putLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putLast(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### offerFirst

public boolean **offerFirst**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E,%20long,%20java.util.concurrent.TimeUnit)) Inserts the specified element at the front of this deque, waiting up to the specified wait time if necessary for space to become available.

**Specified by:**[offerFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E,%20long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### offerLast

public boolean **offerLast**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit)) Inserts the specified element at the end of this deque, waiting up to the specified wait time if necessary for space to become available.

**Specified by:**[offerLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### removeFirst

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#removeFirst()) Retrieves and removes the first element of this deque. This method differs from [pollFirst](http://docs.google.com/java/util/Deque.html#pollFirst()) only in that it throws an exception if this deque is empty.

**Specified by:**[removeFirst](http://docs.google.com/java/util/Deque.html#removeFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the head of this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### removeLast

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#removeLast()) Retrieves and removes the last element of this deque. This method differs from [pollLast](http://docs.google.com/java/util/Deque.html#pollLast()) only in that it throws an exception if this deque is empty.

**Specified by:**[removeLast](http://docs.google.com/java/util/Deque.html#removeLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the tail of this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### pollFirst

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#pollFirst()) Retrieves and removes the first element of this deque, or returns null if this deque is empty.

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

### pollLast

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#pollLast()) Retrieves and removes the last element of this deque, or returns null if this deque is empty.

**Specified by:**[pollLast](http://docs.google.com/java/util/Deque.html#pollLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the tail of this deque, or null if this deque is empty

### takeFirst

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

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

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

### takeLast

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

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

**Specified by:**[takeLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#takeLast()) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the tail of this deque **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### pollFirst

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) **pollFirst**(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:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the first element of this deque, waiting up to the specified wait time if necessary for an element to become available.

**Specified by:**[pollFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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 deque, 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

### pollLast

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) **pollLast**(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:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollLast(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the last element of this deque, waiting up to the specified wait time if necessary for an element to become available.

**Specified by:**[pollLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollLast(long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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 tail of this deque, 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

### getFirst

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#getFirst()) Retrieves, but does not remove, the first element of this deque. This method differs from [peekFirst](http://docs.google.com/java/util/Deque.html#peekFirst()) only in that it throws an exception if this deque is empty.

**Specified by:**[getFirst](http://docs.google.com/java/util/Deque.html#getFirst()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the head of this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### getLast

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#getLast()) Retrieves, but does not remove, the last element of this deque. This method differs from [peekLast](http://docs.google.com/java/util/Deque.html#peekLast()) only in that it throws an exception if this deque is empty.

**Specified by:**[getLast](http://docs.google.com/java/util/Deque.html#getLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the tail of this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### peekFirst

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#peekFirst()) Retrieves, but does not remove, the first element of this deque, or returns null if this deque is empty.

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

### peekLast

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#peekLast()) Retrieves, but does not remove, the last element of this deque, or returns null if this deque is empty.

**Specified by:**[peekLast](http://docs.google.com/java/util/Deque.html#peekLast()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the tail of this deque, or null if this deque is empty

### removeFirstOccurrence

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#removeFirstOccurrence(java.lang.Object)) Removes the first occurrence of the specified element from this deque. If the deque does not contain the element, it is unchanged. More formally, removes the first element e such that o.equals(e) (if such an element exists). Returns true if this deque contained the specified element (or equivalently, if this deque changed as a result of the call).

**Specified by:**[removeFirstOccurrence](http://docs.google.com/java/util/concurrent/BlockingDeque.html#removeFirstOccurrence(java.lang.Object)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[removeFirstOccurrence](http://docs.google.com/java/util/Deque.html#removeFirstOccurrence(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**o - element to be removed from this deque, if present **Returns:**true if an element was removed as a result of this call

### removeLastOccurrence

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#removeLastOccurrence(java.lang.Object)) Removes the last occurrence of the specified element from this deque. If the deque does not contain the element, it is unchanged. More formally, removes the last element e such that o.equals(e) (if such an element exists). Returns true if this deque contained the specified element (or equivalently, if this deque changed as a result of the call).

**Specified by:**[removeLastOccurrence](http://docs.google.com/java/util/concurrent/BlockingDeque.html#removeLastOccurrence(java.lang.Object)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[removeLastOccurrence](http://docs.google.com/java/util/Deque.html#removeLastOccurrence(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**o - element to be removed from this deque, if present **Returns:**true if an element was removed as a result of this call

### add

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

Inserts the specified element at the end of this deque unless it would violate capacity restrictions. When using a capacity-restricted deque, it is generally preferable to use method [offer](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#offer(E)).

This method is equivalent to [addLast(E)](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#addLast(E)).

**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/LinkedBlockingDeque.html)>**Specified by:**[add](http://docs.google.com/java/util/concurrent/BlockingDeque.html#add(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[add](http://docs.google.com/java/util/concurrent/BlockingQueue.html#add(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[add](http://docs.google.com/java/util/Deque.html#add(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.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/LinkedBlockingDeque.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:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to capacity restrictions [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/LinkedBlockingDeque.html) e)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E)) Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque) if it is possible to do so immediately without violating capacity restrictions, returning true upon success and false if no space is currently available. When using a capacity-restricted deque, this method is generally preferable to the [BlockingDeque.add(E)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#add(E)) method, which can fail to insert an element only by throwing an exception.

This method is equivalent to [offerLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E)).

**Specified by:**[offer](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**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/LinkedBlockingDeque.html)>**Specified by:**[offer](http://docs.google.com/java/util/Deque.html#offer(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.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

### put

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#put(E)) Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque), waiting if necessary for space to become available.

This method is equivalent to [putLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putLast(E)).

**Specified by:**[put](http://docs.google.com/java/util/concurrent/BlockingDeque.html#put(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**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/LinkedBlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### offer

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E,%20long,%20java.util.concurrent.TimeUnit)) Inserts the specified element into the queue represented by this deque (in other words, at the tail of this deque), waiting up to the specified wait time if necessary for space to become available.

This method is equivalent to [offerLast](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit)).

**Specified by:**[offer](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E,%20long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**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/LinkedBlockingDeque.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 the element was added to this deque, else false **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### remove

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

Retrieves and removes the head of the queue represented by this deque. This method differs from [poll](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#poll()) only in that it throws an exception if this deque is empty.

This method is equivalent to [removeFirst](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeFirst()).

**Specified by:**[remove](http://docs.google.com/java/util/concurrent/BlockingDeque.html#remove()) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[remove](http://docs.google.com/java/util/Deque.html#remove()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[remove](http://docs.google.com/java/util/Queue.html#remove()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Overrides:**[remove](http://docs.google.com/java/util/AbstractQueue.html#remove()) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the head of the queue represented by this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### poll

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#poll()) Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty.

This method is equivalent to [Deque.pollFirst()](http://docs.google.com/java/util/Deque.html#pollFirst()).

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

### take

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

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

This method is equivalent to [takeFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#takeFirst()).

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

### poll

public [E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#poll(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque), waiting up to the specified wait time if necessary for an element to become available.

This method is equivalent to [pollFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)).

**Specified by:**[poll](http://docs.google.com/java/util/concurrent/BlockingDeque.html#poll(long,%20java.util.concurrent.TimeUnit)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**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/LinkedBlockingDeque.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 deque, 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

### element

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

Retrieves, but does not remove, the head of the queue represented by this deque. This method differs from [peek](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#peek()) only in that it throws an exception if this deque is empty.

This method is equivalent to [getFirst](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#getFirst()).

**Specified by:**[element](http://docs.google.com/java/util/concurrent/BlockingDeque.html#element()) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[element](http://docs.google.com/java/util/Deque.html#element()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[element](http://docs.google.com/java/util/Queue.html#element()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Overrides:**[element](http://docs.google.com/java/util/AbstractQueue.html#element()) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the head of the queue represented by this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### peek

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

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#peek()) Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty.

This method is equivalent to [peekFirst](http://docs.google.com/java/util/Deque.html#peekFirst()).

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

### remainingCapacity

public int **remainingCapacity**()

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

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/LinkedBlockingDeque.html)> **Returns:**the remaining capacity

### drainTo

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

### push

public void **push**([E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html) e)

**Description copied from interface:** [**BlockingDeque**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#push(E)) Pushes an element onto the stack represented by this deque. In other words, inserts the element at the front of this deque unless it would violate capacity restrictions.

This method is equivalent to [addFirst](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E)).

**Specified by:**[push](http://docs.google.com/java/util/concurrent/BlockingDeque.html#push(E)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[push](http://docs.google.com/java/util/Deque.html#push(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Parameters:**e - the element to push **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the element cannot be added at this time due to capacity restrictions [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### pop

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

**Description copied from interface:** [**Deque**](http://docs.google.com/java/util/Deque.html#pop()) Pops an element from the stack represented by this deque. In other words, removes and returns the first element of this deque.

This method is equivalent to [Deque.removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()).

**Specified by:**[pop](http://docs.google.com/java/util/Deque.html#pop()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**the element at the front of this deque (which is the top of the stack represented by this deque) **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### remove

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

Removes the first occurrence of the specified element from this deque. If the deque does not contain the element, it is unchanged. More formally, removes the first element e such that o.equals(e) (if such an element exists). Returns true if this deque contained the specified element (or equivalently, if this deque changed as a result of the call).

This method is equivalent to [removeFirstOccurrence](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html#removeFirstOccurrence(java.lang.Object)).

**Specified by:**[remove](http://docs.google.com/java/util/Collection.html#remove(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[remove](http://docs.google.com/java/util/concurrent/BlockingDeque.html#remove(java.lang.Object)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html)>**Specified by:**[remove](http://docs.google.com/java/util/Deque.html#remove(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Parameters:**o - element to be removed from this deque, if present **Returns:**true if this deque changed as a result of the call

### size

public int **size**()

Returns the number of elements in this deque.

**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/LinkedBlockingDeque.html)>**Specified by:**[size](http://docs.google.com/java/util/concurrent/BlockingDeque.html#size()) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[size](http://docs.google.com/java/util/Deque.html#size()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Returns:**the number of elements in this deque

### contains

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

Returns true if this deque contains the specified element. More formally, returns true if and only if this deque 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/LinkedBlockingDeque.html)>**Specified by:**[contains](http://docs.google.com/java/util/concurrent/BlockingDeque.html#contains(java.lang.Object)) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[contains](http://docs.google.com/java/util/concurrent/BlockingQueue.html#contains(java.lang.Object)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[contains](http://docs.google.com/java/util/Deque.html#contains(java.lang.Object)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Parameters:**o - object to be checked for containment in this deque **Returns:**true if this deque contains the specified element

### toArray

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

Returns an array containing all of the elements in this deque, in proper sequence (from first to last element).

The returned array will be "safe" in that no references to it are maintained by this deque. (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/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Returns:**an array containing all of the elements in this deque

### toArray

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

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

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

Like the [toArray()](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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 deque known to contain only strings. The following code can be used to dump the deque 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/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Parameters:**a - the array into which the elements of the deque 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 deque **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 deque [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/LinkedBlockingDeque.html)> **Returns:**a string representation of this collection

### clear

public void **clear**()

Atomically removes all of the elements from this deque. The deque 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/LinkedBlockingDeque.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/LinkedBlockingDeque.html)>

### iterator

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

Returns an iterator over the elements in this deque in proper sequence. The elements will be returned in order from first (head) to last (tail). 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/LinkedBlockingDeque.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/LinkedBlockingDeque.html)>**Specified by:**[iterator](http://docs.google.com/java/util/concurrent/BlockingDeque.html#iterator()) in interface [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)>**Specified by:**[iterator](http://docs.google.com/java/util/Deque.html#iterator()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.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/LinkedBlockingDeque.html)> **Returns:**an iterator over the elements in this deque in proper sequence

### descendingIterator

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

Returns an iterator over the elements in this deque in reverse sequential order. The elements will be returned in order from last (tail) to first (head). 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:**[descendingIterator](http://docs.google.com/java/util/Deque.html#descendingIterator()) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)> **Returns:**an iterator over the elements in this deque in reverse 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/LinkedBlockingDeque.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/FutureTask.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/LinkedBlockingDeque.html)    [**NO FRAMES**](http://docs.google.com/LinkedBlockingDeque.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [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).