| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BlockingDeque.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/ArrayBlockingQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/BlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/BlockingDeque.html)    [**NO FRAMES**](http://docs.google.com/BlockingDeque.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

## **java.util.concurrent**

Interface BlockingDeque<E>

**Type Parameters:**E - the type of elements held in this collection **All Superinterfaces:** [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>, [Collection](http://docs.google.com/java/util/Collection.html)<E>, [Deque](http://docs.google.com/java/util/Deque.html)<E>, [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Queue](http://docs.google.com/java/util/Queue.html)<E> **All Known Implementing Classes:** [LinkedBlockingDeque](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html)

public interface **BlockingDeque<E>**extends [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>, [Deque](http://docs.google.com/java/util/Deque.html)<E>

A [Deque](http://docs.google.com/java/util/Deque.html) that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.

BlockingDeque methods come in four forms, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: one throws an exception, the second returns a special value (either null or false, depending on the operation), the third blocks the current thread indefinitely until the operation can succeed, and the fourth blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:

| **First Element (Head)** | | | | |
| --- | --- | --- | --- | --- |
|  | *Throws exception* | *Special value* | *Blocks* | *Times out* |
| **Insert** | [addFirst(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E)) | [offerFirst(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E)) | [putFirst(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putFirst(E)) | [offerFirst(e, time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerFirst(E,%20long,%20java.util.concurrent.TimeUnit)) |
| **Remove** | [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()) | [pollFirst()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)) | [takeFirst()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#takeFirst()) | [pollFirst(time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)) |
| **Examine** | [getFirst()](http://docs.google.com/java/util/Deque.html#getFirst()) | [peekFirst()](http://docs.google.com/java/util/Deque.html#peekFirst()) | *not applicable* | *not applicable* |
| **Last Element (Tail)** | | | | |
|  | *Throws exception* | *Special value* | *Blocks* | *Times out* |
| **Insert** | [addLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addLast(E)) | [offerLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E)) | [putLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putLast(E)) | [offerLast(e, time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit)) |
| **Remove** | [removeLast()](http://docs.google.com/java/util/Deque.html#removeLast()) | [pollLast()](http://docs.google.com/java/util/Deque.html#pollLast()) | [takeLast()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#takeLast()) | [pollLast(time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollLast(long,%20java.util.concurrent.TimeUnit)) |
| **Examine** | [getLast()](http://docs.google.com/java/util/Deque.html#getLast()) | [peekLast()](http://docs.google.com/java/util/Deque.html#peekLast()) | *not applicable* | *not applicable* |

Like any [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html), a BlockingDeque is thread safe, does not permit null elements, and may (or may not) be capacity-constrained.

A BlockingDeque implementation may be used directly as a FIFO BlockingQueue. The methods inherited from the BlockingQueue interface are precisely equivalent to BlockingDeque methods as indicated in the following table:

| **BlockingQueue Method** | **Equivalent BlockingDeque Method** |
| --- | --- |
| **Insert** | |
| [add(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#add(E)) | [addLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addLast(E)) |
| [offer(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E)) | [offerLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E)) |
| [put(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#put(E)) | [putLast(e)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#putLast(E)) |
| [offer(e, time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E,%20long,%20java.util.concurrent.TimeUnit)) | [offerLast(e, time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit)) |
| **Remove** | |
| [remove()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#remove()) | [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()) |
| [poll()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#poll()) | [pollFirst()](http://docs.google.com/java/util/Deque.html#pollFirst()) |
| [take()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#take()) | [takeFirst()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#takeFirst()) |
| [poll(time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#poll(long,%20java.util.concurrent.TimeUnit)) | [pollFirst(time, unit)](http://docs.google.com/java/util/concurrent/BlockingDeque.html#pollFirst(long,%20java.util.concurrent.TimeUnit)) |
| **Examine** | |
| [element()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#element()) | [getFirst()](http://docs.google.com/java/util/Deque.html#getFirst()) |
| [peek()](http://docs.google.com/java/util/concurrent/BlockingDeque.html#peek()) | [peekFirst()](http://docs.google.com/java/util/Deque.html#peekFirst()) |

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

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

**Since:** 1.6

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#add(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 throwing an IllegalStateException if no space is currently available. |
| void | [**addFirst**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#addFirst(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#addLast(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this deque contains the specified element. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**element**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#element())()            Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque). |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#iterator())()            Returns an iterator over the elements in this deque in proper sequence. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#offerFirst(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#offerFirst(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#offerLast(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#offerLast(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html) | [**peek**](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. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**poll**](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. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**poll**](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html) | [**pollFirst**](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html) | [**pollLast**](http://docs.google.com/java/util/concurrent/BlockingDeque.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. |
| void | [**push**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#push(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) e)            Pushes an element onto the stack represented by this deque. |
| void | [**put**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#put(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#putFirst(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#putLast(E))([E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) e)            Inserts the specified element at the end of this deque, waiting if necessary for space to become available. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**remove**](http://docs.google.com/java/util/concurrent/BlockingDeque.html#remove())()            Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque). |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/BlockingDeque.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. |
| boolean | [**removeFirstOccurrence**](http://docs.google.com/java/util/concurrent/BlockingDeque.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. |
| boolean | [**removeLastOccurrence**](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.html#size())()            Returns the number of elements in this deque. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**take**](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. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**takeFirst**](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. |
| [E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) | [**takeLast**](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. |

| **Methods inherited from interface java.util.concurrent.**[**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html) |
| --- |
| [drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection)), [drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection,%20int)), [remainingCapacity](http://docs.google.com/java/util/concurrent/BlockingQueue.html#remainingCapacity()) |

| **Methods inherited from interface java.util.**[**Deque**](http://docs.google.com/java/util/Deque.html) |
| --- |
| [descendingIterator](http://docs.google.com/java/util/Deque.html#descendingIterator()), [getFirst](http://docs.google.com/java/util/Deque.html#getFirst()), [getLast](http://docs.google.com/java/util/Deque.html#getLast()), [peekFirst](http://docs.google.com/java/util/Deque.html#peekFirst()), [peekLast](http://docs.google.com/java/util/Deque.html#peekLast()), [pollFirst](http://docs.google.com/java/util/Deque.html#pollFirst()), [pollLast](http://docs.google.com/java/util/Deque.html#pollLast()), [pop](http://docs.google.com/java/util/Deque.html#pop()), [removeFirst](http://docs.google.com/java/util/Deque.html#removeFirst()), [removeLast](http://docs.google.com/java/util/Deque.html#removeLast()) |

| **Methods inherited from interface java.util.**[**Collection**](http://docs.google.com/java/util/Collection.html) |
| --- |
| [addAll](http://docs.google.com/java/util/Collection.html#addAll(java.util.Collection)), [clear](http://docs.google.com/java/util/Collection.html#clear()), [containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)), [equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Collection.html#hashCode()), [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)), [toArray](http://docs.google.com/java/util/Collection.html#toArray()), [toArray](http://docs.google.com/java/util/Collection.html#toArray(T%5B%5D)) |

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

### addFirst

void **addFirst**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. 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/Deque.html#addFirst(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### addLast

void **addLast**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. 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/Deque.html#addLast(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offerFirst

boolean **offerFirst**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. 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/Deque.html#offerFirst(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.html)> **Parameters:**e - the element to add **Returns:**true if the element was added to this deque, else false **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offerLast

boolean **offerLast**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. 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/Deque.html#offerLast(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.html)> **Parameters:**e - the element to add **Returns:**true if the element was added to this deque, else false **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### putFirst

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

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

**Parameters:**e - the element to add **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### putLast

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

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

**Parameters:**e - the element to add **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offerFirst

boolean **offerFirst**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 front of this deque, waiting up to the specified wait time if necessary for space to become available.

**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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offerLast

boolean **offerLast**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 end of this deque, waiting up to the specified wait time if necessary for space to become available.

**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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### takeFirst

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

Retrieves and removes the first element of this deque, waiting if necessary until an element becomes available.

**Returns:**the head of this deque **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### takeLast

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

Retrieves and removes the last element of this deque, waiting if necessary until an element becomes available.

**Returns:**the tail of this deque **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### pollFirst

[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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)

Retrieves and removes the first element of this deque, waiting up to the specified wait time if necessary for an element to become available.

**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

[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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)

Retrieves and removes the last element of this deque, waiting up to the specified wait time if necessary for an element to become available.

**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

### removeFirstOccurrence

boolean **removeFirstOccurrence**([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).

**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/BlockingDeque.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 **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element is incompatible with this deque (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null (optional)

### removeLastOccurrence

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

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/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/BlockingDeque.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 **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element is incompatible with this deque (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null (optional)

### add

boolean **add**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 throwing an IllegalStateException if no space is currently available. When using a capacity-restricted deque, it is generally preferable to use [offer](http://docs.google.com/java/util/concurrent/BlockingDeque.html#offer(E)).

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

**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/BlockingDeque.html)>**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/BlockingDeque.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/BlockingDeque.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/BlockingDeque.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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offer

boolean **offer**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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. When using a capacity-restricted deque, this method is generally preferable to the [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/BlockingQueue.html#offer(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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/BlockingDeque.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/BlockingDeque.html)> **Parameters:**e - the element to add **Returns:**true if the element was added to this queue, else false **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### put

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

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/BlockingQueue.html#put(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.html)> **Parameters:**e - the element to add **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### offer

boolean **offer**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 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/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/BlockingDeque.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:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### remove

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

Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque). This method differs from [poll](http://docs.google.com/java/util/concurrent/BlockingDeque.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/Deque.html#removeFirst()).

**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/BlockingDeque.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/BlockingDeque.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

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

### take

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

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

### poll

[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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)

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/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/BlockingDeque.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

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

Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque). This method differs from [peek](http://docs.google.com/java/util/concurrent/BlockingDeque.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/Deque.html#getFirst()).

**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/BlockingDeque.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/BlockingDeque.html)> **Returns:**the head of this deque **Throws:** [NoSuchElementException](http://docs.google.com/java/util/NoSuchElementException.html) - if this deque is empty

### peek

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

### remove

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/BlockingDeque.html#removeFirstOccurrence(java.lang.Object)).

**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/BlockingDeque.html)>**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/BlockingDeque.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/BlockingDeque.html)> **Parameters:**o - element to be removed from this deque, if present **Returns:**true if this deque changed as a result of the call **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element is incompatible with this deque (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null (optional)

### contains

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/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/BlockingDeque.html)>**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/BlockingDeque.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/BlockingDeque.html)> **Parameters:**o - object to be checked for containment in this deque **Returns:**true if this deque contains the specified element **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element is incompatible with this deque (optional) [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null (optional)

### size

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/BlockingDeque.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/BlockingDeque.html)> **Returns:**the number of elements in this deque

### iterator

[Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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).

**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/BlockingDeque.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/BlockingDeque.html)>**Specified by:**[iterator](http://docs.google.com/java/lang/Iterable.html#iterator()) in interface [Iterable](http://docs.google.com/java/lang/Iterable.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.html)> **Returns:**an iterator over the elements in this deque in proper sequence

### push

void **push**([E](http://docs.google.com/java/util/concurrent/BlockingDeque.html) 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/Deque.html#push(E)) in interface [Deque](http://docs.google.com/java/util/Deque.html)<[E](http://docs.google.com/java/util/concurrent/BlockingDeque.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 [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of the specified element prevents it from being added to this deque [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BlockingDeque.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/ArrayBlockingQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/BlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/BlockingDeque.html)    [**NO FRAMES**](http://docs.google.com/BlockingDeque.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#1t3h5sf) |

[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).