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

## **java.util**

Interface Deque<E>

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

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

A linear collection that supports element insertion and removal at both ends. The name *deque* is short for "double ended queue" and is usually pronounced "deck". Most Deque implementations place no fixed limits on the number of elements they may contain, but this interface supports capacity-restricted deques as well as those with no fixed size limit.

This interface defines methods to access the elements at both ends of the deque. Methods are provided to insert, remove, and examine the element. Each of these methods exists in two forms: one throws an exception if the operation fails, the other returns a special value (either null or false, depending on the operation). The latter form of the insert operation is designed specifically for use with capacity-restricted Deque implementations; in most implementations, insert operations cannot fail.

The twelve methods described above are summarized in the following table:

|  | **First Element (Head)** | | **Last Element (Tail)** | |
| --- | --- | --- | --- | --- |
|  | *Throws exception* | *Special value* | *Throws exception* | *Special value* |
| **Insert** | [addFirst(e)](http://docs.google.com/java/util/Deque.html#addFirst(E)) | [offerFirst(e)](http://docs.google.com/java/util/Deque.html#offerFirst(E)) | [addLast(e)](http://docs.google.com/java/util/Deque.html#addLast(E)) | [offerLast(e)](http://docs.google.com/java/util/Deque.html#offerLast(E)) |
| **Remove** | [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()) | [pollFirst()](http://docs.google.com/java/util/Deque.html#pollFirst()) | [removeLast()](http://docs.google.com/java/util/Deque.html#removeLast()) | [pollLast()](http://docs.google.com/java/util/Deque.html#pollLast()) |
| **Examine** | [getFirst()](http://docs.google.com/java/util/Deque.html#getFirst()) | [peekFirst()](http://docs.google.com/java/util/Deque.html#peekFirst()) | [getLast()](http://docs.google.com/java/util/Deque.html#getLast()) | [peekLast()](http://docs.google.com/java/util/Deque.html#peekLast()) |

This interface extends the [Queue](http://docs.google.com/java/util/Queue.html) interface. When a deque is used as a queue, FIFO (First-In-First-Out) behavior results. Elements are added at the end of the deque and removed from the beginning. The methods inherited from the Queue interface are precisely equivalent to Deque methods as indicated in the following table:

| **Queue Method** | **Equivalent Deque Method** |
| --- | --- |
| [add(e)](http://docs.google.com/java/util/Queue.html#add(E)) | [addLast(e)](http://docs.google.com/java/util/Deque.html#addLast(E)) |
| [offer(e)](http://docs.google.com/java/util/Queue.html#offer(E)) | [offerLast(e)](http://docs.google.com/java/util/Deque.html#offerLast(E)) |
| [remove()](http://docs.google.com/java/util/Queue.html#remove()) | [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()) |
| [poll()](http://docs.google.com/java/util/Queue.html#poll()) | [pollFirst()](http://docs.google.com/java/util/Deque.html#pollFirst()) |
| [element()](http://docs.google.com/java/util/Queue.html#element()) | [getFirst()](http://docs.google.com/java/util/Deque.html#getFirst()) |
| [peek()](http://docs.google.com/java/util/Queue.html#peek()) | [peekFirst()](http://docs.google.com/java/util/Deque.html#peek()) |

Deques can also be used as LIFO (Last-In-First-Out) stacks. This interface should be used in preference to the legacy [Stack](http://docs.google.com/java/util/Stack.html) class. When a deque is used as a stack, elements are pushed and popped from the beginning of the deque. Stack methods are precisely equivalent to Deque methods as indicated in the table below:

| **Stack Method** | **Equivalent Deque Method** |
| --- | --- |
| [push(e)](http://docs.google.com/java/util/Deque.html#push(E)) | [addFirst(e)](http://docs.google.com/java/util/Deque.html#addFirst(E)) |
| [pop()](http://docs.google.com/java/util/Deque.html#pop()) | [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()) |
| [peek()](http://docs.google.com/java/util/Deque.html#peek()) | [peekFirst()](http://docs.google.com/java/util/Deque.html#peekFirst()) |

Note that the [peek](http://docs.google.com/java/util/Deque.html#peek()) method works equally well when a deque is used as a queue or a stack; in either case, elements are drawn from the beginning of the deque.

This interface provides two methods to remove interior elements, [removeFirstOccurrence](http://docs.google.com/java/util/Deque.html#removeFirstOccurrence(java.lang.Object)) and [removeLastOccurrence](http://docs.google.com/java/util/Deque.html#removeLastOccurrence(java.lang.Object)).

Unlike the [List](http://docs.google.com/java/util/List.html) interface, this interface does not provide support for indexed access to elements.

While Deque implementations are not strictly required to prohibit the insertion of null elements, they are strongly encouraged to do so. Users of any Deque implementations that do allow null elements are strongly encouraged *not* to take advantage of the ability to insert nulls. This is so because null is used as a special return value by various methods to indicated that the deque is empty.

Deque implementations generally do not define element-based versions of the equals and hashCode methods, but instead inherit the identity-based versions from class Object.

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/Deque.html#add(E))([E](http://docs.google.com/java/util/Deque.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/Deque.html#addFirst(E))([E](http://docs.google.com/java/util/Deque.html) e)            Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions. |
| void | [**addLast**](http://docs.google.com/java/util/Deque.html#addLast(E))([E](http://docs.google.com/java/util/Deque.html) e)            Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions. |
| boolean | [**contains**](http://docs.google.com/java/util/Deque.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/Deque.html)> | [**descendingIterator**](http://docs.google.com/java/util/Deque.html#descendingIterator())()            Returns an iterator over the elements in this deque in reverse sequential order. |
| [E](http://docs.google.com/java/util/Deque.html) | [**element**](http://docs.google.com/java/util/Deque.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). |
| [E](http://docs.google.com/java/util/Deque.html) | [**getFirst**](http://docs.google.com/java/util/Deque.html#getFirst())()            Retrieves, but does not remove, the first element of this deque. |
| [E](http://docs.google.com/java/util/Deque.html) | [**getLast**](http://docs.google.com/java/util/Deque.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/Deque.html)> | [**iterator**](http://docs.google.com/java/util/Deque.html#iterator())()            Returns an iterator over the elements in this deque in proper sequence. |
| boolean | [**offer**](http://docs.google.com/java/util/Deque.html#offer(E))([E](http://docs.google.com/java/util/Deque.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 | [**offerFirst**](http://docs.google.com/java/util/Deque.html#offerFirst(E))([E](http://docs.google.com/java/util/Deque.html) e)            Inserts the specified element at the front of this deque unless it would violate capacity restrictions. |
| boolean | [**offerLast**](http://docs.google.com/java/util/Deque.html#offerLast(E))([E](http://docs.google.com/java/util/Deque.html) e)            Inserts the specified element at the end of this deque unless it would violate capacity restrictions. |
| [E](http://docs.google.com/java/util/Deque.html) | [**peek**](http://docs.google.com/java/util/Deque.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/Deque.html) | [**peekFirst**](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. |
| [E](http://docs.google.com/java/util/Deque.html) | [**peekLast**](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. |
| [E](http://docs.google.com/java/util/Deque.html) | [**poll**](http://docs.google.com/java/util/Deque.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/Deque.html) | [**pollFirst**](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. |
| [E](http://docs.google.com/java/util/Deque.html) | [**pollLast**](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. |
| [E](http://docs.google.com/java/util/Deque.html) | [**pop**](http://docs.google.com/java/util/Deque.html#pop())()            Pops an element from the stack represented by this deque. |
| void | [**push**](http://docs.google.com/java/util/Deque.html#push(E))([E](http://docs.google.com/java/util/Deque.html) e)            Pushes an element onto the stack represented by this deque (in other words, at the head 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. |
| [E](http://docs.google.com/java/util/Deque.html) | [**remove**](http://docs.google.com/java/util/Deque.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/Deque.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/Deque.html) | [**removeFirst**](http://docs.google.com/java/util/Deque.html#removeFirst())()            Retrieves and removes the first element of this deque. |
| boolean | [**removeFirstOccurrence**](http://docs.google.com/java/util/Deque.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/Deque.html) | [**removeLast**](http://docs.google.com/java/util/Deque.html#removeLast())()            Retrieves and removes the last element of this deque. |
| boolean | [**removeLastOccurrence**](http://docs.google.com/java/util/Deque.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/Deque.html#size())()            Returns the number of elements in this deque. |

| **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/Deque.html) e)

Inserts the specified element at the front of this deque if it is possible to do so immediately without violating capacity restrictions. When using a capacity-restricted deque, it is generally preferable to use method [offerFirst(E)](http://docs.google.com/java/util/Deque.html#offerFirst(E)).

**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 and this deque does not permit null elements [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/Deque.html) e)

Inserts the specified element at the end of this deque if it is possible to do so immediately without violating capacity restrictions. When using a capacity-restricted deque, it is generally preferable to use method [offerLast(E)](http://docs.google.com/java/util/Deque.html#offerLast(E)).

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

**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 and this deque does not permit null elements [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/Deque.html) e)

Inserts the specified element at the front of this deque unless it would violate capacity restrictions. When using a capacity-restricted deque, this method is generally preferable to the [addFirst(E)](http://docs.google.com/java/util/Deque.html#addFirst(E)) method, which can fail to insert an element only by throwing an exception.

**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 and this deque does not permit null elements [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/Deque.html) e)

Inserts the specified element at the end of this deque unless it would violate capacity restrictions. When using a capacity-restricted deque, this method is generally preferable to the [addLast(E)](http://docs.google.com/java/util/Deque.html#addLast(E)) method, which can fail to insert an element only by throwing an exception.

**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 and this deque does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### removeFirst

[E](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.

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

### removeLast

[E](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.

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

### pollFirst

[E](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.

**Returns:**the head of this deque, or null if this deque is empty

### pollLast

[E](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.

**Returns:**the tail of this deque, or null if this deque is empty

### getFirst

[E](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.

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

### getLast

[E](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.

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

### peekFirst

[E](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.

**Returns:**the head of this deque, or null if this deque is empty

### peekLast

[E](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.

**Returns:**the tail of this deque, or null if this deque is empty

### 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==null ? e==null : 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).

**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 and this deque does not permit null elements (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==null ? e==null : 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).

**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 and this deque does not permit null elements (optional)

### add

boolean **add**([E](http://docs.google.com/java/util/Deque.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/Deque.html#offer(E)).

This method is equivalent to [addLast(E)](http://docs.google.com/java/util/Deque.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/Deque.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/Deque.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 and this deque does not permit null elements [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/Deque.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/Deque.html#add(E)) method, which can fail to insert an element only by throwing an exception.

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

**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/Deque.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 and this deque does not permit null elements [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/Deque.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/Deque.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/Queue.html#remove()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/Deque.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/Deque.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 [pollFirst()](http://docs.google.com/java/util/Deque.html#pollFirst()).

**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/Deque.html)> **Returns:**the first element of this deque, or null if this deque is empty

### element

[E](http://docs.google.com/java/util/Deque.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/Deque.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/Queue.html#element()) in interface [Queue](http://docs.google.com/java/util/Queue.html)<[E](http://docs.google.com/java/util/Deque.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

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

### push

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

Pushes an element onto the stack represented by this deque (in other words, at the head 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.

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

**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 and this deque does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of the specified element prevents it from being added to this deque

### pop

[E](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 [removeFirst()](http://docs.google.com/java/util/Deque.html#removeFirst()).

**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

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==null ? e==null : 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(java.lang.Object)](http://docs.google.com/java/util/Deque.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/Deque.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 and this deque does not permit null elements (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==null ? e==null : 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/Deque.html)> **Parameters:**o - element whose presence in this deque is to be tested **Returns:**true if this deque contains the specified element **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the type 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 and this deque does not permit null elements (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/Deque.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/Deque.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/Deque.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/Deque.html)> **Returns:**an iterator over the elements in this deque in proper sequence

### descendingIterator

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

**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/Deque.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/Date.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Dictionary.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Deque.html)    [**NO FRAMES**](http://docs.google.com/Deque.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#tyjcwt) |

[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).