**Queue interface Methods**

|  |  |
| --- | --- |
| **boolean add (E e)** | **boolean offer (E e)** |
| Inserts the specified element into this queue 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. | Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions. When using a capacity-restricted queue, this method is generally preferable to add(E), which can fail to insert an element only by throwing an exception. |
| **Throws:**  **IllegalStateException -** if the element cannot be added at this time due to capacity restrictions  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null and this queue does not permit null elements  **IllegalArgumentException -** if some property of this element prevents it from being added to this queue | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null and this queue does not permit null elements  **IllegalArgumentException -** if some property of this element prevents it from being added to this queue |
| **Returns:**  true (as specified by Collection.add(E)) | **Returns:**  **true** if the element was added to this queue, else false |

|  |  |  |  |
| --- | --- | --- | --- |
| **E remove ()** | **E poll ()** | **E element ()** | **E peek ()** |
| Retrieves and **removes the head of this queue.** | Retrieves and **removes the head of this queue or returns null if this queue is empty.** | Retrieves, **but does not remove**, **the head of this queue.** | Retrieves, **but does not remove, the head of this queue, or returns null if this queue is empty.** |
| **Returns: -**the head of this queue | **Returns: -**the head of this queue, or null if this queue is empty | **Returns: -**the head of this queue | **Returns: -**the head of this queue, or null if this queue is empty |
| **Throws:** -**NoSuchElementException** - if this queue is empty |  | **Throws: -** **NoSuchElementException** - if this queue is empty |  |
| This method differs from poll only in that it throws an exception if this queue is empty. |  | This method differs from peek only in that it throws an exception if this queue is empty. |  |

**BlockingQueue Interface Methods.**

|  |  |  |  |
| --- | --- | --- | --- |
| **boolean add (E e)** | **boolean offer (E e)** | **void put (E e) throws InterruptedException** | **boolean offer (E e, long timeout, TimeUnit unit) throws InterruptedException** |
| Inserts the specified element into this queue 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 queue, it is generally preferable to use offer. | Inserts the specified element into this queue 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 queue, this method is generally preferable to add(E), which can fail to insert an element only by throwing an exception. | Inserts the specified element into this queue, waiting if necessary, for space to become available. | Inserts the specified element into this queue, waiting up to the specified wait time if necessary, for space to become available. |
| **Returns:**  true (as specified by Collection.add(E)) | **Returns:**  true if the element was added to this queue, else false |  | **Returns:**  true if successful, or false if the specified waiting time elapses before space is available |
| **Throws:**  **IllegalStateException -** if the element cannot be added at this time due to capacity restrictions  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue | **Throws:**  **InterruptedException -** if interrupted while waiting  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue | **Throws:**  **InterruptedException -** if interrupted while waiting  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E take () throws InterruptedException** | **E poll (long timeout, TimeUnit unit) throws InterruptedException** | **boolean remove (Object o)** | **int drainTo(Collection<? super E> c)** | **int drainTo(Collection<? super E> c, int maxElements)** |
| Retrieves and removes the head of this queue, waiting if necessary, until an element becomes available. | Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary, for an element to become available. | Removes a single instance of the specified element from this queue, if it is present. More formally, removes an element e such that o.equals(e), if this queue contains one or more such elements. Returns true if this queue contained the specified element (or equivalently, if this queue changed as a result of the call). | Removes all available elements from this queue and adds them to the given collection. This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress. | Removes at most the given number of available elements from this queue and adds them to the given collection. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress. |
| **Returns: -** the head of this queue | **Returns:**  the head of this queue, or null if the specified waiting time elapses before an element is available | **Returns:**  true if this queue changed as a result of the call | **Returns:**  the number of elements transferred | **Returns:**  the number of elements transferred |
| **Throws:**  **InterruptedException -** if interrupted while waiting | **Throws:**  **InterruptedException -** if interrupted while waiting | **Throws:**  **ClassCastException -** if the class of the specified element is incompatible with this queue (optional)  **NullPointerException -** if the specified element is null (optional) | **Throws:**  **UnsupportedOperationException -** if addition of elements is not supported by the specified collection  **ClassCastException -** if the class of an element of this queue prevents it from being added to the specified collection  **NullPointerException -** if the specified collection is null  IllegalArgumentException - if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection | **Throws:**  **UnsupportedOperationException -** if addition of elements is not supported by the specified collection  **ClassCastException -** if the class of an element of this queue prevents it from being added to the specified collection  **NullPointerException -** if the specified collection is null  **IllegalArgumentException -** if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection |

**TransferQueue interface methods**

|  |  |  |
| --- | --- | --- |
| **void transfer (E e) throws InterruptedException** | **boolean tryTransfer(E e)** | **boolean tryTransfer(E e, long timeout, TimeUnit unit) throws InterruptedException** |
| Transfers the element to a consumer, waiting if necessary, to do so.  More precisely, transfers the specified element immediately if there exists a consumer already waiting to receive it (in BlockingQueue.take() or timed poll), else waits until the element is received by a consumer. | Transfers the element to a waiting consumer immediately, if possible.  More precisely, transfers the specified element immediately if there exists a consumer already waiting to receive it (in BlockingQueue.take() or timed poll), otherwise returning false without enqueuing the element. | Transfers the element to a consumer if it is possible to do so before the timeout elapses.  More precisely, transfers the specified element immediately if there exists a consumer already waiting to receive it (in BlockingQueue.take() or timed poll), else waits until the element is received by a consumer, returning false if the specified wait time elapses before the element can be transferred. |
|  | **Returns:**  true if the element was transferred, else false | **Returns:**  true if successful, or false if the specified waiting time elapses before completion, in which case the element is not left enqueued |
| **Throws:**  **InterruptedException -** if interrupted while waiting, in which case the element is not left enqueued  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue | **Throws:**  **InterruptedException -** if interrupted while waiting, in which case the element is not left enqueued  **ClassCastException -** if the class of the specified element prevents it from being added to this queue  **NullPointerException -** if the specified element is null  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this queue |

**Deque Interface**

**Summary of Deque methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **First Element (Head)** |  | **Last Element (Tail)** |  |
|  | *Throws exception* | *Special value* | *Throws exception* | *Special value* |
| **Insert** | [addFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#addFirst-E-) | [offerFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#offerFirst-E-) | [addLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#addLast-E-) | [offerLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#offerLast-E-) |
| **Remove** | [removeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeFirst--) | [pollFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pollFirst--) | [removeLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeLast--) | [pollLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pollLast--) |
| **Examine** | [getFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getFirst--) | [peekFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekFirst--) | [getLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getLast--) | [peekLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekLast--) |

**Comparison of Queue and Deque methods**

|  |  |
| --- | --- |
| **Queue Method** | **Equivalent Deque Method** |
| [add(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#add-E-) | [addLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#addLast-E-) |
| [offer(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#offer-E-) | [offerLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#offerLast-E-) |
| [remove()](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#remove--) | [removeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeFirst--) |
| [poll()](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#poll--) | [pollFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pollFirst--) |
| [element()](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#element--) | [getFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getFirst--) |
| [peek()](https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html#peek--) | [peekFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peek--) |

**Comparison of Stack and Deque methods**

|  |  |
| --- | --- |
| **Stack Method** | **Equivalent Deque Method** |
| [push(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#push-E-) | [addFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#addFirst-E-) |
| [pop()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pop--) | [removeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeFirst--) |
| [peek()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peek--) | [peekFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekFirst--) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **void addFirst(E e)** | **void addLast(E e)** | **boolean offerFirst(E e)** | **boolean offerLast(E e)** | **boolean add (E e)** | **boolean offer (E 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 method offerFirst(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 method offerLast(E).  This method is equivalent to add(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) method, which can fail to insert an element only by throwing an exception. | 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) method, which can fail to insert an element only by throwing an exception. | 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.  This method is equivalent to addLast(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) method, which can fail to insert an element only by throwing an exception.  This method is equivalent to offerLast(E). |
|  |  | **Returns:**  true if the element was added to this deque, else false | **Returns:**  true if the element was added to this deque, else false | **Returns:**  true (as specified by Collection.add(E)) | **Returns:**  true if the element was added to this deque, else false |
| **Throws:**  **IllegalStateException -** if the element cannot be added at this time due to capacity restrictions  **ClassCastException -** if the class of the specified element prevents it from being added to this deque  **NullPointerException - i**f the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque | **Throws:**  **IllegalStateException -** if the element cannot be added at this time due to capacity restrictions  **ClassCastException -** if the class of the specified element prevents it from being added to this deque  **NullPointerException -** if the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this deque  **NullPointerException -** if the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this deque  **NullPointerException** - if the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque | **Throws:**  **IllegalStateException -** if the element cannot be added at this time due to capacity restrictions  ClassCastException - if the class of the specified element prevents it from being added to this deque  **NullPointerException -** if the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque | **Throws:**  **ClassCastException -** if the class of the specified element prevents it from being added to this deque  **NullPointerException -** if the specified element is null and this deque does not permit null elements  **IllegalArgumentException -** if some property of the specified element prevents it from being added to this deque |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **E removeFirst()** | **E removeLast()** | **E pollFirst()** | **E pollLast()** | **E getFirst()** | **E getLast()** | **E peekFirst()** | **E peekLast()** |
| Retrieves and removes the first element of this deque. This method differs from pollFirst only in that it throws an exception if this deque is empty. | Retrieves and removes the last element of this deque. This method differs from pollLast only in that it throws an exception if this deque is empty. | Retrieves and removes the first element of this deque, or returns null if this deque is empty. | Retrieves and removes the last element of this deque, or returns null if this deque is empty. | Retrieves, but does not remove, the first element of this deque. This method differs from peekFirst only in that it throws an exception if this deque is empty. | Retrieves, but does not remove, the last element of this deque. This method differs from peekLast only in that it throws an exception if this deque is empty. | Retrieves, but does not remove, the first element of this deque, or returns null if this deque is empty. | Retrieves, but does not remove, the last element of this deque, or returns null if this deque is empty. |
| **Returns:**  the head of this deque | **Returns:** the tail of this deque | **Returns:** the head of this deque, or null if this deque is empty | **Returns:** the tail of this deque, or null if this deque is empty | **Returns:** the head of this deque | **Returns:** the tail of this deque | **Returns:** the head of this deque, or null if this deque is empty | **Returns:** the tail of this deque, or null if this deque is empty |
| **Throws:**  NoSuchElementException - if this deque is empty | **Throws:** NoSuchElementException - if this deque is empty |  |  | **Throws:** NoSuchElementException - if this deque is empty | **Throws:** NoSuchElementException - if this deque is empty |  |  |

**BlockingDeque Interface**

**Summary of BlockingDeque methods**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **First Element (Head)** |  |  |  |  |
|  | *Throws exception* | *Special value* | *Blocks* | *Times out* |
| **Insert** | [addFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#addFirst-E-) | [offerFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerFirst-E-) | [putFirst(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#putFirst-E-) | [offerFirst(e, time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerFirst-E-long-java.util.concurrent.TimeUnit-) |
| **Remove** | [removeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeFirst--) | [pollFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#pollFirst-long-java.util.concurrent.TimeUnit-) | [takeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#takeFirst--) | [pollFirst(time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#pollFirst-long-java.util.concurrent.TimeUnit-) |
| **Examine** | [getFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getFirst--) | [peekFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekFirst--) | *not applicable* | *not applicable* |
| **Last Element (Tail)** |  |  |  |  |
|  | *Throws exception* | *Special value* | *Blocks* | *Times out* |
| **Insert** | [addLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#addLast-E-) | [offerLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerLast-E-) | [putLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#putLast-E-) | [offerLast(e, time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerLast-E-long-java.util.concurrent.TimeUnit-) |
| **Remove** | [removeLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeLast--) | [pollLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pollLast--) | [takeLast()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#takeLast--) | [pollLast(time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#pollLast-long-java.util.concurrent.TimeUnit-) |
| **Examine** | [getLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getLast--) | [peekLast()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekLast--) | *not applicable* | *not applicable* |

**Comparison of BlockingQueue and BlockingDeque methods**

|  |  |
| --- | --- |
| **BlockingQueue Method** | **Equivalent BlockingDeque Method** |
| **Insert** |  |
| [add(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#add-E-) | [addLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#addLast-E-) |
| [offer(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offer-E-) | [offerLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerLast-E-) |
| [put(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#put-E-) | [putLast(e)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#putLast-E-) |
| [offer(e, time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offer-E-long-java.util.concurrent.TimeUnit-) | [offerLast(e, time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#offerLast-E-long-java.util.concurrent.TimeUnit-) |
| **Remove** |  |
| [remove()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#remove--) | [removeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#removeFirst--) |
| [poll()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#poll--) | [pollFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#pollFirst--) |
| [take()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#take--) | [takeFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#takeFirst--) |
| [poll(time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#poll-long-java.util.concurrent.TimeUnit-) | [pollFirst(time, unit)](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#pollFirst-long-java.util.concurrent.TimeUnit-) |
| **Examine** |  |
| [element()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#element--) | [getFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#getFirst--) |
| [peek()](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BlockingDeque.html#peek--) | [peekFirst()](https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html#peekFirst--) |