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

## **java.util**

Interface Queue<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> **All Known Subinterfaces:** [BlockingDeque](http://docs.google.com/java/util/concurrent/BlockingDeque.html)<E>, [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>, [Deque](http://docs.google.com/java/util/Deque.html)<E> **All Known Implementing Classes:** [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html), [ArrayBlockingQueue](http://docs.google.com/java/util/concurrent/ArrayBlockingQueue.html), [ArrayDeque](http://docs.google.com/java/util/ArrayDeque.html), [ConcurrentLinkedQueue](http://docs.google.com/java/util/concurrent/ConcurrentLinkedQueue.html), [DelayQueue](http://docs.google.com/java/util/concurrent/DelayQueue.html), [LinkedBlockingDeque](http://docs.google.com/java/util/concurrent/LinkedBlockingDeque.html), [LinkedBlockingQueue](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html), [LinkedList](http://docs.google.com/java/util/LinkedList.html), [PriorityBlockingQueue](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html), [PriorityQueue](http://docs.google.com/java/util/PriorityQueue.html), [SynchronousQueue](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)

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

A collection designed for holding elements prior to processing. Besides basic [Collection](http://docs.google.com/java/util/Collection.html) operations, queues provide additional insertion, extraction, and inspection operations. 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 Queue implementations; in most implementations, insert operations cannot fail.

|  | *Throws exception* | *Returns special value* |
| --- | --- | --- |
| **Insert** | [add(e)](http://docs.google.com/java/util/Queue.html#add(E)) | [offer(e)](http://docs.google.com/java/util/Queue.html#offer(E)) |
| **Remove** | [remove()](http://docs.google.com/java/util/Queue.html#remove()) | [poll()](http://docs.google.com/java/util/Queue.html#poll()) |
| **Examine** | [element()](http://docs.google.com/java/util/Queue.html#element()) | [peek()](http://docs.google.com/java/util/Queue.html#peek()) |

Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner. Among the exceptions are priority queues, which order elements according to a supplied comparator, or the elements' natural ordering, and LIFO queues (or stacks) which order the elements LIFO (last-in-first-out). Whatever the ordering used, the *head* of the queue is that element which would be removed by a call to [remove()](http://docs.google.com/java/util/Queue.html#remove()) or [poll()](http://docs.google.com/java/util/Queue.html#poll()). In a FIFO queue, all new elements are inserted at the  *tail* of the queue. Other kinds of queues may use different placement rules. Every Queue implementation must specify its ordering properties.

The [offer](http://docs.google.com/java/util/Queue.html#offer(E)) method inserts an element if possible, otherwise returning false. This differs from the [Collection.add](http://docs.google.com/java/util/Collection.html#add(E)) method, which can fail to add an element only by throwing an unchecked exception. The offer method is designed for use when failure is a normal, rather than exceptional occurrence, for example, in fixed-capacity (or "bounded") queues.

The [remove()](http://docs.google.com/java/util/Queue.html#remove()) and [poll()](http://docs.google.com/java/util/Queue.html#poll()) methods remove and return the head of the queue. Exactly which element is removed from the queue is a function of the queue's ordering policy, which differs from implementation to implementation. The remove() and poll() methods differ only in their behavior when the queue is empty: the remove() method throws an exception, while the poll() method returns null.

The [element()](http://docs.google.com/java/util/Queue.html#element()) and [peek()](http://docs.google.com/java/util/Queue.html#peek()) methods return, but do not remove, the head of the queue.

The Queue interface does not define the *blocking queue methods*, which are common in concurrent programming. These methods, which wait for elements to appear or for space to become available, are defined in the [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html) interface, which extends this interface.

Queue implementations generally do not allow insertion of null elements, although some implementations, such as [LinkedList](http://docs.google.com/java/util/LinkedList.html), do not prohibit insertion of null. Even in the implementations that permit it, null should not be inserted into a Queue, as null is also used as a special return value by the poll method to indicate that the queue contains no elements.

Queue implementations generally do not define element-based versions of methods equals and hashCode but instead inherit the identity based versions from class Object, because element-based equality is not always well-defined for queues with the same elements but different ordering properties.

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

**Since:** 1.5 **See Also:**[Collection](http://docs.google.com/java/util/Collection.html), [LinkedList](http://docs.google.com/java/util/LinkedList.html), [PriorityQueue](http://docs.google.com/java/util/PriorityQueue.html), [LinkedBlockingQueue](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html), [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html), [ArrayBlockingQueue](http://docs.google.com/java/util/concurrent/ArrayBlockingQueue.html), [LinkedBlockingQueue](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html), [PriorityBlockingQueue](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/Queue.html#add(E))([E](http://docs.google.com/java/util/Queue.html) 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. |
| [E](http://docs.google.com/java/util/Queue.html) | [**element**](http://docs.google.com/java/util/Queue.html#element())()            Retrieves, but does not remove, the head of this queue. |
| boolean | [**offer**](http://docs.google.com/java/util/Queue.html#offer(E))([E](http://docs.google.com/java/util/Queue.html) e)            Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions. |
| [E](http://docs.google.com/java/util/Queue.html) | [**peek**](http://docs.google.com/java/util/Queue.html#peek())()            Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty. |
| [E](http://docs.google.com/java/util/Queue.html) | [**poll**](http://docs.google.com/java/util/Queue.html#poll())()            Retrieves and removes the head of this queue, or returns null if this queue is empty. |
| [E](http://docs.google.com/java/util/Queue.html) | [**remove**](http://docs.google.com/java/util/Queue.html#remove())()            Retrieves and removes the head of this queue. |

| **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()), [contains](http://docs.google.com/java/util/Collection.html#contains(java.lang.Object)), [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()), [iterator](http://docs.google.com/java/util/Collection.html#iterator()), [remove](http://docs.google.com/java/util/Collection.html#remove(java.lang.Object)), [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)), [size](http://docs.google.com/java/util/Collection.html#size()), [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** |
| --- |

### add

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

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

### offer

boolean **offer**([E](http://docs.google.com/java/util/Queue.html) e)

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)](http://docs.google.com/java/util/Queue.html#add(E)), 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 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 queue [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null and this queue does not permit null elements [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if some property of this element prevents it from being added to this queue

### remove

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

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

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

### poll

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

Retrieves and removes the head of this queue, or returns null if this queue is empty.

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

### element

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

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

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

### peek

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

Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.

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

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Queue.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/PropertyResourceBundle.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Random.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/Queue.html)    [**NO FRAMES**](http://docs.google.com/Queue.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).