| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PriorityBlockingQueue.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/LinkedBlockingQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/RejectedExecutionException.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/PriorityBlockingQueue.html)    [**NO FRAMES**](http://docs.google.com/PriorityBlockingQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#1ksv4uv) |

## **java.util.concurrent**

Class PriorityBlockingQueue<E>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.util.AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<E>  
 [java.util.AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<E>  
 **java.util.concurrent.PriorityBlockingQueue<E>**

**Type Parameters:**E - the type of elements held in this collection **All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Iterable](http://docs.google.com/java/lang/Iterable.html)<E>, [Collection](http://docs.google.com/java/util/Collection.html)<E>, [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>, [Queue](http://docs.google.com/java/util/Queue.html)<E>

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

An unbounded [blocking queue](http://docs.google.com/java/util/concurrent/BlockingQueue.html) that uses the same ordering rules as class [PriorityQueue](http://docs.google.com/java/util/PriorityQueue.html) and supplies blocking retrieval operations. While this queue is logically unbounded, attempted additions may fail due to resource exhaustion (causing OutOfMemoryError). This class does not permit null elements. A priority queue relying on [natural ordering](http://docs.google.com/java/lang/Comparable.html) also does not permit insertion of non-comparable objects (doing so results in ClassCastException).

This class and its iterator implement all of the *optional* methods of the [Collection](http://docs.google.com/java/util/Collection.html) and [Iterator](http://docs.google.com/java/util/Iterator.html) interfaces. The Iterator provided in method [iterator()](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#iterator()) is *not* guaranteed to traverse the elements of the PriorityBlockingQueue in any particular order. If you need ordered traversal, consider using Arrays.sort(pq.toArray()). Also, method drainTo can be used to *remove* some or all elements in priority order and place them in another collection.

Operations on this class make no guarantees about the ordering of elements with equal priority. If you need to enforce an ordering, you can define custom classes or comparators that use a secondary key to break ties in primary priority values. For example, here is a class that applies first-in-first-out tie-breaking to comparable elements. To use it, you would insert a new FIFOEntry(anEntry) instead of a plain entry object.

class FIFOEntry<E extends Comparable<? super E>>  
 implements Comparable<FIFOEntry<E>> {  
 final static AtomicLong seq = new AtomicLong();  
 final long seqNum;  
 final E entry;  
 public FIFOEntry(E entry) {  
 seqNum = seq.getAndIncrement();  
 this.entry = entry;  
 }  
 public E getEntry() { return entry; }  
 public int compareTo(FIFOEntry<E> other) {  
 int res = entry.compareTo(other.entry);  
 if (res == 0 && other.entry != this.entry)  
 res = (seqNum < other.seqNum ? -1 : 1);  
 return res;  
 }  
 }

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

**Since:** 1.5 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.util.concurrent.PriorityBlockingQueue)

| **Constructor Summary** | |
| --- | --- |
| [**PriorityBlockingQueue**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#PriorityBlockingQueue())()            Creates a PriorityBlockingQueue with the default initial capacity (11) that orders its elements according to their [natural ordering](http://docs.google.com/java/lang/Comparable.html). |
| [**PriorityBlockingQueue**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#PriorityBlockingQueue(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> c)            Creates a PriorityBlockingQueue containing the elements in the specified collection. |
| [**PriorityBlockingQueue**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#PriorityBlockingQueue(int))(int initialCapacity)            Creates a PriorityBlockingQueue with the specified initial capacity that orders its elements according to their [natural ordering](http://docs.google.com/java/lang/Comparable.html). |
| [**PriorityBlockingQueue**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#PriorityBlockingQueue(int,%20java.util.Comparator))(int initialCapacity, [Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> comparator)            Creates a PriorityBlockingQueue with the specified initial capacity that orders its elements according to the specified comparator. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#add(E))([E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) e)            Inserts the specified element into this priority queue. |
| void | [**clear**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#clear())()            Atomically removes all of the elements from this queue. |
| [Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> | [**comparator**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#comparator())()            Returns the comparator used to order the elements in this queue, or null if this queue uses the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of its elements. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Returns true if this queue contains the specified element. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#drainTo(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> c)            Removes all available elements from this queue and adds them to the given collection. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#drainTo(java.util.Collection,%20int))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> c, int maxElements)            Removes at most the given number of available elements from this queue and adds them to the given collection. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#iterator())()            Returns an iterator over the elements in this queue. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#offer(E))([E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) e)            Inserts the specified element into this priority queue. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element into this priority queue. |
| [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) | [**peek**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.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/concurrent/PriorityBlockingQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.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/concurrent/PriorityBlockingQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.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 this queue, waiting up to the specified wait time if necessary for an element to become available. |
| void | [**put**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#put(E))([E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) e)            Inserts the specified element into this priority queue. |
| int | [**remainingCapacity**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#remainingCapacity())()            Always returns Integer.MAX\_VALUE because a PriorityBlockingQueue is not capacity constrained. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Removes a single instance of the specified element from this queue, if it is present. |
| int | [**size**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#size())()            Returns the number of elements in this collection. |
| [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) | [**take**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#take())()            Retrieves and removes the head of this queue, waiting if necessary until an element becomes available. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#toArray())()            Returns an array containing all of the elements in this queue. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#toArray(T%5B%5D))(T[] a)            Returns an array containing all of the elements in this queue; the runtime type of the returned array is that of the specified array. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#toString())()            Returns a string representation of this collection. |

| **Methods inherited from class java.util.**[**AbstractQueue**](http://docs.google.com/java/util/AbstractQueue.html) |
| --- |
| [addAll](http://docs.google.com/java/util/AbstractQueue.html#addAll(java.util.Collection)), [element](http://docs.google.com/java/util/AbstractQueue.html#element()), [remove](http://docs.google.com/java/util/AbstractQueue.html#remove()) |

| **Methods inherited from class java.util.**[**AbstractCollection**](http://docs.google.com/java/util/AbstractCollection.html) |
| --- |
| [containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)), [isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()), [removeAll](http://docs.google.com/java/util/AbstractCollection.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/AbstractCollection.html#retainAll(java.util.Collection)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Methods inherited from interface java.util.**[**Queue**](http://docs.google.com/java/util/Queue.html) |
| --- |
| [element](http://docs.google.com/java/util/Queue.html#element()), [remove](http://docs.google.com/java/util/Queue.html#remove()) |

| **Methods inherited from interface java.util.**[**Collection**](http://docs.google.com/java/util/Collection.html) |
| --- |
| [addAll](http://docs.google.com/java/util/Collection.html#addAll(java.util.Collection)), [containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)), [equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Collection.html#hashCode()), [isEmpty](http://docs.google.com/java/util/Collection.html#isEmpty()), [removeAll](http://docs.google.com/java/util/Collection.html#removeAll(java.util.Collection)), [retainAll](http://docs.google.com/java/util/Collection.html#retainAll(java.util.Collection)) |

| **Constructor Detail** |
| --- |

### PriorityBlockingQueue

public **PriorityBlockingQueue**()

Creates a PriorityBlockingQueue with the default initial capacity (11) that orders its elements according to their [natural ordering](http://docs.google.com/java/lang/Comparable.html).

### PriorityBlockingQueue

public **PriorityBlockingQueue**(int initialCapacity)

Creates a PriorityBlockingQueue with the specified initial capacity that orders its elements according to their [natural ordering](http://docs.google.com/java/lang/Comparable.html).

**Parameters:**initialCapacity - the initial capacity for this priority queue **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if initialCapacity is less than 1

### PriorityBlockingQueue

public **PriorityBlockingQueue**(int initialCapacity,  
 [Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> comparator)

Creates a PriorityBlockingQueue with the specified initial capacity that orders its elements according to the specified comparator.

**Parameters:**initialCapacity - the initial capacity for this priority queuecomparator - the comparator that will be used to order this priority queue. If null, the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of the elements will be used. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if initialCapacity is less than 1

### PriorityBlockingQueue

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

Creates a PriorityBlockingQueue containing the elements in the specified collection. If the specified collection is a [SortedSet](http://docs.google.com/java/util/SortedSet.html) or a [PriorityQueue](http://docs.google.com/java/util/PriorityQueue.html), this priority queue will be ordered according to the same ordering. Otherwise, this priority queue will be ordered according to the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of its elements.

**Parameters:**c - the collection whose elements are to be placed into this priority queue **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if elements of the specified collection cannot be compared to one another according to the priority queue's ordering [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection or any of its elements are null

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

### add

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

Inserts the specified element into this priority queue.

**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/PriorityBlockingQueue.html)>**Specified by:**[add](http://docs.google.com/java/util/concurrent/BlockingQueue.html#add(E)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.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/PriorityBlockingQueue.html)>**Overrides:**[add](http://docs.google.com/java/util/AbstractQueue.html#add(E)) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.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:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the specified element cannot be compared with elements currently in the priority queue according to the priority queue's ordering [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### offer

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

Inserts the specified element into this priority queue.

**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/PriorityBlockingQueue.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/PriorityBlockingQueue.html)> **Parameters:**e - the element to add **Returns:**true (as specified by [Queue.offer(E)](http://docs.google.com/java/util/Queue.html#offer(E))) **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the specified element cannot be compared with elements currently in the priority queue according to the priority queue's ordering [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### put

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

Inserts the specified element into this priority queue. As the queue is unbounded this method will never block.

**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/PriorityBlockingQueue.html)> **Parameters:**e - the element to add **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the specified element cannot be compared with elements currently in the priority queue according to the priority queue's ordering [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### offer

public boolean **offer**([E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) e,  
 long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

Inserts the specified element into this priority queue. As the queue is unbounded this method will never block.

**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/PriorityBlockingQueue.html)> **Parameters:**e - the element to addtimeout - This parameter is ignored as the method never blocksunit - This parameter is ignored as the method never blocks **Returns:**true **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the specified element cannot be compared with elements currently in the priority queue according to the priority queue's ordering [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### poll

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

**Description copied from interface:** [**Queue**](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.

**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/PriorityBlockingQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### take

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

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

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

### poll

public [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html) **poll**(long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#poll(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

**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/PriorityBlockingQueue.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 queue, 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

### peek

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

**Description copied from interface:** [**Queue**](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.

**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/PriorityBlockingQueue.html)> **Returns:**the head of this queue, or null if this queue is empty

### comparator

public [Comparator](http://docs.google.com/java/util/Comparator.html)<? super [E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **comparator**()

Returns the comparator used to order the elements in this queue, or null if this queue uses the [natural ordering](http://docs.google.com/java/lang/Comparable.html) of its elements.

**Returns:**the comparator used to order the elements in this queue, or null if this queue uses the natural ordering of its elements

### size

public int **size**()

**Description copied from interface:** [**Collection**](http://docs.google.com/java/util/Collection.html#size()) Returns the number of elements in this collection. If this collection contains more than Integer.MAX\_VALUE elements, returns Integer.MAX\_VALUE.

**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/PriorityBlockingQueue.html)>**Specified by:**[size](http://docs.google.com/java/util/AbstractCollection.html#size()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Returns:**the number of elements in this collection

### remainingCapacity

public int **remainingCapacity**()

Always returns Integer.MAX\_VALUE because a PriorityBlockingQueue is not capacity constrained.

**Specified by:**[remainingCapacity](http://docs.google.com/java/util/concurrent/BlockingQueue.html#remainingCapacity()) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Returns:**Integer.MAX\_VALUE

### remove

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

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 and only if this queue contained the specified element (or equivalently, if this queue changed as a result of the call).

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

### contains

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

Returns true if this queue contains the specified element. More formally, returns true if and only if this queue contains at least one element e such that o.equals(e).

**Specified by:**[contains](http://docs.google.com/java/util/Collection.html#contains(java.lang.Object)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Specified by:**[contains](http://docs.google.com/java/util/concurrent/BlockingQueue.html#contains(java.lang.Object)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Overrides:**[contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Parameters:**o - object to be checked for containment in this queue **Returns:**true if this queue contains the specified element

### toArray

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

Returns an array containing all of the elements in this queue. The returned array elements are in no particular order.

The returned array will be "safe" in that no references to it are maintained by this queue. (In other words, this method must allocate a new array). The caller is thus free to modify the returned array.

This method acts as bridge between array-based and collection-based APIs.

**Specified by:**[toArray](http://docs.google.com/java/util/Collection.html#toArray()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Returns:**an array containing all of the elements in this queue

### toString

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

**Description copied from class:** [**AbstractCollection**](http://docs.google.com/java/util/AbstractCollection.html#toString()) Returns a string representation of this collection. The string representation consists of a list of the collection's elements in the order they are returned by its iterator, enclosed in square brackets ("[]"). Adjacent elements are separated by the characters ", " (comma and space). Elements are converted to strings as by [String.valueOf(Object)](http://docs.google.com/java/lang/String.html#valueOf(java.lang.Object)).

**Overrides:**[toString](http://docs.google.com/java/util/AbstractCollection.html#toString()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Returns:**a string representation of this collection

### drainTo

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

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection)) Removes all available elements from this queue and adds them to the given collection. This operation may be more efficient than repeatedly polling this queue. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Specified by:**[drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Parameters:**c - the collection to transfer elements into **Returns:**the number of elements transferred **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if addition of elements is not supported by the specified collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this queue prevents it from being added to the specified collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

### drainTo

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

**Description copied from interface:** [**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection,%20int)) Removes at most the given number of available elements from this queue and adds them to the given collection. A failure encountered while attempting to add elements to collection c may result in elements being in neither, either or both collections when the associated exception is thrown. Attempts to drain a queue to itself result in IllegalArgumentException. Further, the behavior of this operation is undefined if the specified collection is modified while the operation is in progress.

**Specified by:**[drainTo](http://docs.google.com/java/util/concurrent/BlockingQueue.html#drainTo(java.util.Collection,%20int)) in interface [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Parameters:**c - the collection to transfer elements intomaxElements - the maximum number of elements to transfer **Returns:**the number of elements transferred **Throws:** [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if addition of elements is not supported by the specified collection [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if the class of an element of this queue prevents it from being added to the specified collection [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified collection is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified collection is this queue, or some property of an element of this queue prevents it from being added to the specified collection

### clear

public void **clear**()

Atomically removes all of the elements from this queue. The queue will be empty after this call returns.

**Specified by:**[clear](http://docs.google.com/java/util/Collection.html#clear()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Overrides:**[clear](http://docs.google.com/java/util/AbstractQueue.html#clear()) in class [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>

### toArray

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

Returns an array containing all of the elements in this queue; the runtime type of the returned array is that of the specified array. The returned array elements are in no particular order. If the queue fits in the specified array, it is returned therein. Otherwise, a new array is allocated with the runtime type of the specified array and the size of this queue.

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

Like the [toArray()](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html#toArray()) method, this method acts as bridge between array-based and collection-based APIs. Further, this method allows precise control over the runtime type of the output array, and may, under certain circumstances, be used to save allocation costs.

Suppose x is a queue known to contain only strings. The following code can be used to dump the queue into a newly allocated array of String:

String[] y = x.toArray(new String[0]);

Note that toArray(new Object[0]) is identical in function to toArray().

**Specified by:**[toArray](http://docs.google.com/java/util/Collection.html#toArray(T%5B%5D)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Overrides:**[toArray](http://docs.google.com/java/util/AbstractCollection.html#toArray(T%5B%5D)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Parameters:**a - the array into which the elements of the queue are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose **Returns:**an array containing all of the elements in this queue **Throws:** [ArrayStoreException](http://docs.google.com/java/lang/ArrayStoreException.html) - if the runtime type of the specified array is not a supertype of the runtime type of every element in this queue [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### iterator

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

Returns an iterator over the elements in this queue. The iterator does not return the elements in any particular order. The returned Iterator is a "weakly consistent" iterator that will never throw [ConcurrentModificationException](http://docs.google.com/java/util/ConcurrentModificationException.html), and guarantees to traverse elements as they existed upon construction of the iterator, and may (but is not guaranteed to) reflect any modifications subsequent to construction.

**Specified by:**[iterator](http://docs.google.com/java/lang/Iterable.html#iterator()) in interface [Iterable](http://docs.google.com/java/lang/Iterable.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Specified by:**[iterator](http://docs.google.com/java/util/Collection.html#iterator()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)>**Specified by:**[iterator](http://docs.google.com/java/util/AbstractCollection.html#iterator()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/PriorityBlockingQueue.html)> **Returns:**an iterator over the elements in this queue

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/PriorityBlockingQueue.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/LinkedBlockingQueue.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/RejectedExecutionException.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/PriorityBlockingQueue.html)    [**NO FRAMES**](http://docs.google.com/PriorityBlockingQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#1ksv4uv) |

[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).