| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DelayQueue.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/Delayed.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Exchanger.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/DelayQueue.html)    [**NO FRAMES**](http://docs.google.com/DelayQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

## **java.util.concurrent**

Class DelayQueue<E extends [**Delayed**](http://docs.google.com/java/util/concurrent/Delayed.html)>

[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.DelayQueue<E>**

**Type Parameters:**E - the type of elements held in this collection **All Implemented Interfaces:** [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 **DelayQueue<E extends** [**Delayed**](http://docs.google.com/java/util/concurrent/Delayed.html)**>**extends [AbstractQueue](http://docs.google.com/java/util/AbstractQueue.html)<E>implements [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<E>

An unbounded [blocking queue](http://docs.google.com/java/util/concurrent/BlockingQueue.html) of Delayed elements, in which an element can only be taken when its delay has expired. The *head* of the queue is that Delayed element whose delay expired furthest in the past. If no delay has expired there is no head and poll will return null. Expiration occurs when an element's getDelay(TimeUnit.NANOSECONDS) method returns a value less than or equal to zero. Even though unexpired elements cannot be removed using take or poll, they are otherwise treated as normal elements. For example, the size method returns the count of both expired and unexpired elements. This queue does not permit null elements.

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.

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

**Since:** 1.5

| **Constructor Summary** | |
| --- | --- |
| [**DelayQueue**](http://docs.google.com/java/util/concurrent/DelayQueue.html#DelayQueue())()            Creates a new DelayQueue that is initially empty. |
| [**DelayQueue**](http://docs.google.com/java/util/concurrent/DelayQueue.html#DelayQueue(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [E](http://docs.google.com/java/util/concurrent/DelayQueue.html)> c)            Creates a DelayQueue initially containing the elements of the given collection of [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html) instances. |

| **Method Summary** | |
| --- | --- |
| boolean | [**add**](http://docs.google.com/java/util/concurrent/DelayQueue.html#add(E))([E](http://docs.google.com/java/util/concurrent/DelayQueue.html) e)            Inserts the specified element into this delay queue. |
| void | [**clear**](http://docs.google.com/java/util/concurrent/DelayQueue.html#clear())()            Atomically removes all of the elements from this delay queue. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/DelayQueue.html#drainTo(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/DelayQueue.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/DelayQueue.html#drainTo(java.util.Collection,%20int))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/DelayQueue.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/DelayQueue.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/DelayQueue.html#iterator())()            Returns an iterator over all the elements (both expired and unexpired) in this queue. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/DelayQueue.html#offer(E))([E](http://docs.google.com/java/util/concurrent/DelayQueue.html) e)            Inserts the specified element into this delay queue. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/DelayQueue.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/DelayQueue.html) e, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element into this delay queue. |
| [E](http://docs.google.com/java/util/concurrent/DelayQueue.html) | [**peek**](http://docs.google.com/java/util/concurrent/DelayQueue.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/DelayQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/DelayQueue.html#poll())()            Retrieves and removes the head of this queue, or returns null if this queue has no elements with an expired delay. |
| [E](http://docs.google.com/java/util/concurrent/DelayQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/DelayQueue.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 if necessary until an element with an expired delay is available on this queue, or the specified wait time expires. |
| void | [**put**](http://docs.google.com/java/util/concurrent/DelayQueue.html#put(E))([E](http://docs.google.com/java/util/concurrent/DelayQueue.html) e)            Inserts the specified element into this delay queue. |
| int | [**remainingCapacity**](http://docs.google.com/java/util/concurrent/DelayQueue.html#remainingCapacity())()            Always returns Integer.MAX\_VALUE because a DelayQueue is not capacity constrained. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/DelayQueue.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, whether or not it has expired. |
| int | [**size**](http://docs.google.com/java/util/concurrent/DelayQueue.html#size())()            Returns the number of elements in this collection. |
| [E](http://docs.google.com/java/util/concurrent/DelayQueue.html) | [**take**](http://docs.google.com/java/util/concurrent/DelayQueue.html#take())()            Retrieves and removes the head of this queue, waiting if necessary until an element with an expired delay is available on this queue. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/DelayQueue.html#toArray())()            Returns an array containing all of the elements in this queue. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/DelayQueue.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. |

| **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) |
| --- |
| [contains](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object)), [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)), [toString](http://docs.google.com/java/util/AbstractCollection.html#toString()) |

| **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.concurrent.**[**BlockingQueue**](http://docs.google.com/java/util/concurrent/BlockingQueue.html) |
| --- |
| [contains](http://docs.google.com/java/util/concurrent/BlockingQueue.html#contains(java.lang.Object)) |

| **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** |
| --- |

### DelayQueue

public **DelayQueue**()

Creates a new DelayQueue that is initially empty.

### DelayQueue

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

Creates a DelayQueue initially containing the elements of the given collection of [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html) instances.

**Parameters:**c - the collection of elements to initially contain **Throws:** [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/DelayQueue.html) e)

Inserts the specified element into this delay 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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:** [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/DelayQueue.html) e)

Inserts the specified element into this delay 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Parameters:**e - the element to add **Returns:**true **Throws:** [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/DelayQueue.html) e)

Inserts the specified element into this delay 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Parameters:**e - the element to add **Throws:** [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/DelayQueue.html) e,  
 long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

Inserts the specified element into this delay 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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:** [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/DelayQueue.html) **poll**()

Retrieves and removes the head of this queue, or returns null if this queue has no elements with an expired delay.

**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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Returns:**the head of this queue, or null if this queue has no elements with an expired delay

### take

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

Retrieves and removes the head of this queue, waiting if necessary until an element with an expired delay is available on this queue.

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

Retrieves and removes the head of this queue, waiting if necessary until an element with an expired delay is available on this queue, or the specified wait time expires.

**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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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 with an expired delay becomes 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/DelayQueue.html) **peek**()

Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty. Unlike poll, if no expired elements are available in the queue, this method returns the element that will expire next, if one exists.

**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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Returns:**the head of this queue, or null if this queue is empty.

### 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Returns:**the number of elements in this collection

### drainTo

public int **drainTo**([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/DelayQueue.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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 delay queue. The queue will be empty after this call returns. Elements with an unexpired delay are not waited for; they are simply discarded from the queue.

**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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)>

### remainingCapacity

public int **remainingCapacity**()

Always returns Integer.MAX\_VALUE because a DelayQueue 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Returns:**Integer.MAX\_VALUE

### 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Returns:**an array containing all of the elements in this queue

### 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/DelayQueue.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.

The following code can be used to dump a delay queue into a newly allocated array of Delayed:

Delayed[] a = q.toArray(new Delayed[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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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

### 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, whether or not it has expired.

**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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.html)> **Parameters:**o - element to be removed from this collection, if present **Returns:**true if an element was removed as a result of this call

### iterator

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

Returns an iterator over all the elements (both expired and unexpired) 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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.html) extends [Delayed](http://docs.google.com/java/util/concurrent/Delayed.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/DelayQueue.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/Delayed.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Exchanger.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/DelayQueue.html)    [**NO FRAMES**](http://docs.google.com/DelayQueue.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

[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).