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

A [blocking queue](http://docs.google.com/java/util/concurrent/BlockingQueue.html) in which each insert operation must wait for a corresponding remove operation by another thread, and vice versa. A synchronous queue does not have any internal capacity, not even a capacity of one. You cannot peek at a synchronous queue because an element is only present when you try to remove it; you cannot insert an element (using any method) unless another thread is trying to remove it; you cannot iterate as there is nothing to iterate. The *head* of the queue is the element that the first queued inserting thread is trying to add to the queue; if there is no such queued thread then no element is available for removal and poll() will return null. For purposes of other Collection methods (for example contains), a SynchronousQueue acts as an empty collection. This queue does not permit null elements.

Synchronous queues are similar to rendezvous channels used in CSP and Ada. They are well suited for handoff designs, in which an object running in one thread must sync up with an object running in another thread in order to hand it some information, event, or task.

This class supports an optional fairness policy for ordering waiting producer and consumer threads. By default, this ordering is not guaranteed. However, a queue constructed with fairness set to true grants threads access in FIFO order.

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

| **Constructor Summary** | |
| --- | --- |
| [**SynchronousQueue**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#SynchronousQueue())()            Creates a SynchronousQueue with nonfair access policy. |
| [**SynchronousQueue**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#SynchronousQueue(boolean))(boolean fair)            Creates a SynchronousQueue with the specified fairness policy. |

| **Method Summary** | |
| --- | --- |
| void | [**clear**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#clear())()            Does nothing. |
| boolean | [**contains**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#contains(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Always returns false. |
| boolean | [**containsAll**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#containsAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Returns false unless the given collection is empty. |
| int | [**drainTo**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#drainTo(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.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/SynchronousQueue.html#drainTo(java.util.Collection,%20int))([Collection](http://docs.google.com/java/util/Collection.html)<? super [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> c, int maxElements)            Removes at most the given number of available elements from this queue and adds them to the given collection. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#isEmpty())()            Always returns true. |
| [Iterator](http://docs.google.com/java/util/Iterator.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> | [**iterator**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#iterator())()            Returns an empty iterator in which hasNext always returns false. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#offer(E))([E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) e)            Inserts the specified element into this queue, if another thread is waiting to receive it. |
| boolean | [**offer**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#offer(E,%20long,%20java.util.concurrent.TimeUnit))([E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) o, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it. |
| [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) | [**peek**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#peek())()            Always returns null. |
| [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#poll())()            Retrieves and removes the head of this queue, if another thread is currently making an element available. |
| [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) | [**poll**](http://docs.google.com/java/util/concurrent/SynchronousQueue.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 up to the specified wait time, for another thread to insert it. |
| void | [**put**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#put(E))([E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) o)            Adds the specified element to this queue, waiting if necessary for another thread to receive it. |
| int | [**remainingCapacity**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#remainingCapacity())()            Always returns zero. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#remove(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Always returns false. |
| boolean | [**removeAll**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#removeAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Always returns false. |
| boolean | [**retainAll**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#retainAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<?> c)            Always returns false. |
| int | [**size**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#size())()            Always returns zero. |
| [E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) | [**take**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#take())()            Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**toArray**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#toArray())()            Returns a zero-length array. |
| | <T> T[] | | --- | | [**toArray**](http://docs.google.com/java/util/concurrent/SynchronousQueue.html#toArray(T%5B%5D))(T[] a)            Sets the zeroeth element of the specified array to null (if the array has non-zero length) and returns it. |

| **Methods inherited from class java.util.**[**AbstractQueue**](http://docs.google.com/java/util/AbstractQueue.html) |
| --- |
| [add](http://docs.google.com/java/util/AbstractQueue.html#add(E)), [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) |
| --- |
| [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) |
| --- |
| [add](http://docs.google.com/java/util/concurrent/BlockingQueue.html#add(E)) |

| **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)), [equals](http://docs.google.com/java/util/Collection.html#equals(java.lang.Object)), [hashCode](http://docs.google.com/java/util/Collection.html#hashCode()) |

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

### SynchronousQueue

public **SynchronousQueue**()

Creates a SynchronousQueue with nonfair access policy.

### SynchronousQueue

public **SynchronousQueue**(boolean fair)

Creates a SynchronousQueue with the specified fairness policy.

**Parameters:**fair - if true, waiting threads contend in FIFO order for access; otherwise the order is unspecified.

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

### put

public void **put**([E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html) o)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Adds the specified element to this queue, waiting if necessary for another thread to receive it.

**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/SynchronousQueue.html)> **Parameters:**o - the element to add **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [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/SynchronousQueue.html) o,  
 long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Inserts the specified element into this queue, waiting if necessary up to the specified wait time for another thread to receive it.

**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/SynchronousQueue.html)> **Parameters:**o - the element to addtimeout - how long to wait before giving up, in units of unitunit - a TimeUnit determining how to interpret the timeout parameter **Returns:**true if successful, or false if the specified waiting time elapses before a consumer appears. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [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/SynchronousQueue.html) e)

Inserts the specified element into this queue, if another thread is waiting to receive it.

**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/SynchronousQueue.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/SynchronousQueue.html)> **Parameters:**e - the element to add **Returns:**true if the element was added to this queue, else false **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified element is null

### take

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

Retrieves and removes the head of this queue, waiting if necessary for another thread to insert it.

**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/SynchronousQueue.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/SynchronousQueue.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 up to the specified wait time, for another thread to insert it.

**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/SynchronousQueue.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 present. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### poll

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

Retrieves and removes the head of this queue, if another thread is currently making an element available.

**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/SynchronousQueue.html)> **Returns:**the head of this queue, or null if no element is available.

### isEmpty

public boolean **isEmpty**()

Always returns true. A SynchronousQueue has no internal capacity.

**Specified by:**[isEmpty](http://docs.google.com/java/util/Collection.html#isEmpty()) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)>**Overrides:**[isEmpty](http://docs.google.com/java/util/AbstractCollection.html#isEmpty()) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> **Returns:**true

### size

public int **size**()

Always returns zero. A SynchronousQueue has no internal capacity.

**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/SynchronousQueue.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/SynchronousQueue.html)> **Returns:**zero.

### remainingCapacity

public int **remainingCapacity**()

Always returns zero. A SynchronousQueue has no internal capacity.

**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/SynchronousQueue.html)> **Returns:**zero.

### clear

public void **clear**()

Does nothing. A SynchronousQueue has no internal capacity.

**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/SynchronousQueue.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/SynchronousQueue.html)>

### contains

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

Always returns false. A SynchronousQueue has no internal capacity.

**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/SynchronousQueue.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/SynchronousQueue.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/SynchronousQueue.html)> **Parameters:**o - the element **Returns:**false

### remove

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

Always returns false. A SynchronousQueue has no internal capacity.

**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/SynchronousQueue.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/SynchronousQueue.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/SynchronousQueue.html)> **Parameters:**o - the element to remove **Returns:**false

### containsAll

public boolean **containsAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Returns false unless the given collection is empty. A SynchronousQueue has no internal capacity.

**Specified by:**[containsAll](http://docs.google.com/java/util/Collection.html#containsAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)>**Overrides:**[containsAll](http://docs.google.com/java/util/AbstractCollection.html#containsAll(java.util.Collection)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> **Parameters:**c - the collection **Returns:**false unless given collection is empty**See Also:**[AbstractCollection.contains(Object)](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))

### removeAll

public boolean **removeAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Always returns false. A SynchronousQueue has no internal capacity.

**Specified by:**[removeAll](http://docs.google.com/java/util/Collection.html#removeAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)>**Overrides:**[removeAll](http://docs.google.com/java/util/AbstractCollection.html#removeAll(java.util.Collection)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> **Parameters:**c - the collection **Returns:**false**See Also:**[AbstractCollection.remove(Object)](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)), [AbstractCollection.contains(Object)](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))

### retainAll

public boolean **retainAll**([Collection](http://docs.google.com/java/util/Collection.html)<?> c)

Always returns false. A SynchronousQueue has no internal capacity.

**Specified by:**[retainAll](http://docs.google.com/java/util/Collection.html#retainAll(java.util.Collection)) in interface [Collection](http://docs.google.com/java/util/Collection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)>**Overrides:**[retainAll](http://docs.google.com/java/util/AbstractCollection.html#retainAll(java.util.Collection)) in class [AbstractCollection](http://docs.google.com/java/util/AbstractCollection.html)<[E](http://docs.google.com/java/util/concurrent/SynchronousQueue.html)> **Parameters:**c - the collection **Returns:**false**See Also:**[AbstractCollection.remove(Object)](http://docs.google.com/java/util/AbstractCollection.html#remove(java.lang.Object)), [AbstractCollection.contains(Object)](http://docs.google.com/java/util/AbstractCollection.html#contains(java.lang.Object))

### peek

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

Always returns null. A SynchronousQueue does not return elements unless actively waited on.

**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/SynchronousQueue.html)> **Returns:**null

### iterator

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

Returns an empty iterator in which hasNext always returns false.

**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/SynchronousQueue.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/SynchronousQueue.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/SynchronousQueue.html)> **Returns:**an empty iterator

### toArray

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

Returns a zero-length array.

**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/SynchronousQueue.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/SynchronousQueue.html)> **Returns:**a zero-length array

### toArray

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

Sets the zeroeth element of the specified array to null (if the array has non-zero length) and returns it.

**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/SynchronousQueue.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/SynchronousQueue.html)> **Parameters:**a - the array **Returns:**the specified array **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the specified array is null

### drainTo

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

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