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

## **java.util.concurrent**

Interface CompletionService<V>

**All Known Implementing Classes:** [ExecutorCompletionService](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)

public interface **CompletionService<V>**

A service that decouples the production of new asynchronous tasks from the consumption of the results of completed tasks. Producers submit tasks for execution. Consumers take completed tasks and process their results in the order they complete. A CompletionService can for example be used to manage asynchronous IO, in which tasks that perform reads are submitted in one part of a program or system, and then acted upon in a different part of the program when the reads complete, possibly in a different order than they were requested.

Typically, a CompletionService relies on a separate [Executor](http://docs.google.com/java/util/concurrent/Executor.html) to actually execute the tasks, in which case the CompletionService only manages an internal completion queue. The [ExecutorCompletionService](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html) class provides an implementation of this approach.

Memory consistency effects: Actions in a thread prior to submitting a task to a CompletionService [*happen-before*](http://docs.google.com/package-summary.html#MemoryVisibility) actions taken by that task, which in turn *happen-before* actions following a successful return from the corresponding take().

| **Method Summary** | |
| --- | --- |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> | [**poll**](http://docs.google.com/java/util/concurrent/CompletionService.html#poll())()            Retrieves and removes the Future representing the next completed task or null if none are present. |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> | [**poll**](http://docs.google.com/java/util/concurrent/CompletionService.html#poll(long,%20java.util.concurrent.TimeUnit))(long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Retrieves and removes the Future representing the next completed task, waiting if necessary up to the specified wait time if none are yet present. |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> | [**submit**](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> task)            Submits a value-returning task for execution and returns a Future representing the pending results of the task. |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> | [**submit**](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.lang.Runnable,%20V))([Runnable](http://docs.google.com/java/lang/Runnable.html) task, [V](http://docs.google.com/java/util/concurrent/CompletionService.html) result)            Submits a Runnable task for execution and returns a Future representing that task. |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> | [**take**](http://docs.google.com/java/util/concurrent/CompletionService.html#take())()            Retrieves and removes the Future representing the next completed task, waiting if none are yet present. |

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

### submit

[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> **submit**([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> task)

Submits a value-returning task for execution and returns a Future representing the pending results of the task. Upon completion, this task may be taken or polled.

**Parameters:**task - the task to submit **Returns:**a Future representing pending completion of the task **Throws:** [RejectedExecutionException](http://docs.google.com/java/util/concurrent/RejectedExecutionException.html) - if the task cannot be scheduled for execution [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the task is null

### submit

[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> **submit**([Runnable](http://docs.google.com/java/lang/Runnable.html) task,  
 [V](http://docs.google.com/java/util/concurrent/CompletionService.html) result)

Submits a Runnable task for execution and returns a Future representing that task. Upon completion, this task may be taken or polled.

**Parameters:**task - the task to submitresult - the result to return upon successful completion **Returns:**a Future representing pending completion of the task, and whose get() method will return the given result value upon completion **Throws:** [RejectedExecutionException](http://docs.google.com/java/util/concurrent/RejectedExecutionException.html) - if the task cannot be scheduled for execution [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the task is null

### take

[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> **take**()  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Retrieves and removes the Future representing the next completed task, waiting if none are yet present.

**Returns:**the Future representing the next completed task **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### poll

[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.html)> **poll**()

Retrieves and removes the Future representing the next completed task or null if none are present.

**Returns:**the Future representing the next completed task, or null if none are present

### poll

[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/CompletionService.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 Future representing the next completed task, waiting if necessary up to the specified wait time if none are yet present.

**Parameters:**timeout - how long to wait before giving up, in units of unitunit - a TimeUnit determining how to interpret the timeout parameter **Returns:**the Future representing the next completed task or null if the specified waiting time elapses before one is present **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

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

[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).