| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ExecutorCompletionService.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/Executor.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Executors.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ExecutorCompletionService.html)    [**NO FRAMES**](http://docs.google.com/ExecutorCompletionService.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

## **java.util.concurrent**

Class ExecutorCompletionService<V>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.ExecutorCompletionService<V>**

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

public class **ExecutorCompletionService<V>**extends [Object](http://docs.google.com/java/lang/Object.html)implements [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<V>

A [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html) that uses a supplied [Executor](http://docs.google.com/java/util/concurrent/Executor.html) to execute tasks. This class arranges that submitted tasks are, upon completion, placed on a queue accessible using take. The class is lightweight enough to be suitable for transient use when processing groups of tasks.

**Usage Examples.** Suppose you have a set of solvers for a certain problem, each returning a value of some type Result, and would like to run them concurrently, processing the results of each of them that return a non-null value, in some method use(Result r). You could write this as:

void solve(Executor e,  
 Collection<Callable<Result>> solvers)  
 throws InterruptedException, ExecutionException {  
 CompletionService<Result> ecs  
 = new ExecutorCompletionService<Result>(e);  
 for (Callable<Result> s : solvers)  
 ecs.submit(s);  
 int n = solvers.size();  
 for (int i = 0; i < n; ++i) {  
 Result r = ecs.take().get();  
 if (r != null)  
 use(r);  
 }  
 }

Suppose instead that you would like to use the first non-null result of the set of tasks, ignoring any that encounter exceptions, and cancelling all other tasks when the first one is ready:

void solve(Executor e,  
 Collection<Callable<Result>> solvers)  
 throws InterruptedException {  
 CompletionService<Result> ecs  
 = new ExecutorCompletionService<Result>(e);  
 int n = solvers.size();  
 List<Future<Result>> futures  
 = new ArrayList<Future<Result>>(n);  
 Result result = null;  
 try {  
 for (Callable<Result> s : solvers)  
 futures.add(ecs.submit(s));  
 for (int i = 0; i < n; ++i) {  
 try {  
 Result r = ecs.take().get();  
 if (r != null) {  
 result = r;  
 break;  
 }  
 } catch (ExecutionException ignore) {}  
 }  
 }  
 finally {  
 for (Future<Result> f : futures)  
 f.cancel(true);  
 }  
  
 if (result != null)  
 use(result);  
 }

| **Constructor Summary** | |
| --- | --- |
| [**ExecutorCompletionService**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html#ExecutorCompletionService(java.util.concurrent.Executor))([Executor](http://docs.google.com/java/util/concurrent/Executor.html) executor)            Creates an ExecutorCompletionService using the supplied executor for base task execution and a [LinkedBlockingQueue](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) as a completion queue. |
| [**ExecutorCompletionService**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html#ExecutorCompletionService(java.util.concurrent.Executor,%20java.util.concurrent.BlockingQueue))([Executor](http://docs.google.com/java/util/concurrent/Executor.html) executor, [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)>> completionQueue)            Creates an ExecutorCompletionService using the supplied executor for base task execution and the supplied queue as its completion queue. |

| **Method Summary** | |
| --- | --- |
| [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)> | [**poll**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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/ExecutorCompletionService.html)> | [**poll**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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/ExecutorCompletionService.html)> | [**submit**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html#submit(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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/ExecutorCompletionService.html)> | [**submit**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html#submit(java.lang.Runnable,%20V))([Runnable](http://docs.google.com/java/lang/Runnable.html) task, [V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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/ExecutorCompletionService.html)> | [**take**](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html#take())()            Retrieves and removes the Future representing the next completed task, waiting if none are yet present. |

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

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

### ExecutorCompletionService

public **ExecutorCompletionService**([Executor](http://docs.google.com/java/util/concurrent/Executor.html) executor)

Creates an ExecutorCompletionService using the supplied executor for base task execution and a [LinkedBlockingQueue](http://docs.google.com/java/util/concurrent/LinkedBlockingQueue.html) as a completion queue.

**Parameters:**executor - the executor to use **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if executor is null

### ExecutorCompletionService

public **ExecutorCompletionService**([Executor](http://docs.google.com/java/util/concurrent/Executor.html) executor,  
 [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)>> completionQueue)

Creates an ExecutorCompletionService using the supplied executor for base task execution and the supplied queue as its completion queue.

**Parameters:**executor - the executor to usecompletionQueue - the queue to use as the completion queue normally one dedicated for use by this service **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if executor or completionQueue are null

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

### submit

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

**Description copied from interface:** [**CompletionService**](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.util.concurrent.Callable)) 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.

**Specified by:**[submit](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.util.concurrent.Callable)) in interface [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)> **Parameters:**task - the task to submit **Returns:**a Future representing pending completion of the task

### submit

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

**Description copied from interface:** [**CompletionService**](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.lang.Runnable,%20V)) Submits a Runnable task for execution and returns a Future representing that task. Upon completion, this task may be taken or polled.

**Specified by:**[submit](http://docs.google.com/java/util/concurrent/CompletionService.html#submit(java.lang.Runnable,%20V)) in interface [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)> **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

### take

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

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

**Specified by:**[take](http://docs.google.com/java/util/concurrent/CompletionService.html#take()) in interface [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)> **Returns:**the Future representing the next completed task **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting

### poll

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

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

**Specified by:**[poll](http://docs.google.com/java/util/concurrent/CompletionService.html#poll()) in interface [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.html)> **Returns:**the Future representing the next completed task, or null if none are present

### poll

public [Future](http://docs.google.com/java/util/concurrent/Future.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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:** [**CompletionService**](http://docs.google.com/java/util/concurrent/CompletionService.html#poll(long,%20java.util.concurrent.TimeUnit)) Retrieves and removes the Future representing the next completed task, waiting if necessary up to the specified wait time if none are yet present.

**Specified by:**[poll](http://docs.google.com/java/util/concurrent/CompletionService.html#poll(long,%20java.util.concurrent.TimeUnit)) in interface [CompletionService](http://docs.google.com/java/util/concurrent/CompletionService.html)<[V](http://docs.google.com/java/util/concurrent/ExecutorCompletionService.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 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/ExecutorCompletionService.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/Executor.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Executors.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ExecutorCompletionService.html)    [**NO FRAMES**](http://docs.google.com/ExecutorCompletionService.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

[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).