| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractExecutorService.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   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ArrayBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/AbstractExecutorService.html)    [**NO FRAMES**](http://docs.google.com/AbstractExecutorService.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

## **java.util.concurrent**

Class AbstractExecutorService

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.concurrent.AbstractExecutorService**

**All Implemented Interfaces:** [Executor](http://docs.google.com/java/util/concurrent/Executor.html), [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Direct Known Subclasses:** [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html)

public abstract class **AbstractExecutorService**extends [Object](http://docs.google.com/java/lang/Object.html)implements [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html)

Provides default implementations of [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) execution methods. This class implements the submit, invokeAny and invokeAll methods using a [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html) returned by newTaskFor, which defaults to the [FutureTask](http://docs.google.com/java/util/concurrent/FutureTask.html) class provided in this package. For example, the implementation of submit(Runnable) creates an associated RunnableFuture that is executed and returned. Subclasses may override the newTaskFor methods to return RunnableFuture implementations other than FutureTask.

**Extension example**. Here is a sketch of a class that customizes [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) to use a CustomTask class instead of the default FutureTask:

public class CustomThreadPoolExecutor extends ThreadPoolExecutor {  
  
 static class CustomTask<V> implements RunnableFuture<V> {...}  
  
 protected <V> RunnableFuture<V> newTaskFor(Callable<V> c) {  
 return new CustomTask<V>(c);  
 }  
 protected <V> RunnableFuture<V> newTaskFor(Runnable r, V v) {  
 return new CustomTask<V>(r, v);  
 }  
 // ... add constructors, etc.  
 }

**Since:** 1.5

| **Constructor Summary** | |
| --- | --- |
| [**AbstractExecutorService**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#AbstractExecutorService())() |

| **Method Summary** | |
| --- | --- |
| | <T> [List](http://docs.google.com/java/util/List.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<T>> | | --- | | [**invokeAll**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAll(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks)            Executes the given tasks, returning a list of Futures holding their status and results when all complete. |
| | <T> [List](http://docs.google.com/java/util/List.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<T>> | | --- | | [**invokeAll**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAll(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Executes the given tasks, returning a list of Futures holding their status and results when all complete or the timeout expires, whichever happens first. |
| | <T> T | | --- | | [**invokeAny**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAny(java.util.Collection))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks)            Executes the given tasks, returning the result of one that has completed successfully (i.e., without throwing an exception), if any do. |
| | <T> T | | --- | | [**invokeAny**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAny(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit))([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks, long timeout, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Executes the given tasks, returning the result of one that has completed successfully (i.e., without throwing an exception), if any do before the given timeout elapses. |
| protected   | <T> [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<T> | | --- | | [**newTaskFor**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#newTaskFor(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> callable)            Returns a RunnableFuture for the given callable task. |
| protected   | <T> [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<T> | | --- | | [**newTaskFor**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#newTaskFor(java.lang.Runnable,%20T))([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable, T value)            Returns a RunnableFuture for the given runnable and default value. |
| | <T> [Future](http://docs.google.com/java/util/concurrent/Future.html)<T> | | --- | | [**submit**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.util.concurrent.Callable))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> 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)<?> | [**submit**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.lang.Runnable))([Runnable](http://docs.google.com/java/lang/Runnable.html) task)            Submits a Runnable task for execution and returns a Future representing that task. |
| | <T> [Future](http://docs.google.com/java/util/concurrent/Future.html)<T> | | --- | | [**submit**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.lang.Runnable,%20T))([Runnable](http://docs.google.com/java/lang/Runnable.html) task, T result)            Submits a Runnable task for execution and returns a Future representing that task. |

| **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)) |

| **Methods inherited from interface java.util.concurrent.**[**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html) |
| --- |
| [awaitTermination](http://docs.google.com/java/util/concurrent/ExecutorService.html#awaitTermination(long,%20java.util.concurrent.TimeUnit)), [isShutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#isShutdown()), [isTerminated](http://docs.google.com/java/util/concurrent/ExecutorService.html#isTerminated()), [shutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdown()), [shutdownNow](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdownNow()) |

| **Methods inherited from interface java.util.concurrent.**[**Executor**](http://docs.google.com/java/util/concurrent/Executor.html) |
| --- |
| [execute](http://docs.google.com/java/util/concurrent/Executor.html#execute(java.lang.Runnable)) |

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

### AbstractExecutorService

public **AbstractExecutorService**()

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

### newTaskFor

protected <T> [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<T> **newTaskFor**([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable,  
 T value)

Returns a RunnableFuture for the given runnable and default value.

**Parameters:**runnable - the runnable task being wrappedvalue - the default value for the returned future **Returns:**a RunnableFuture which when run will run the underlying runnable and which, as a Future, will yield the given value as its result and provide for cancellation of the underlying task.**Since:** 1.6

### newTaskFor

protected <T> [RunnableFuture](http://docs.google.com/java/util/concurrent/RunnableFuture.html)<T> **newTaskFor**([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T> callable)

Returns a RunnableFuture for the given callable task.

**Parameters:**callable - the callable task being wrapped **Returns:**a RunnableFuture which when run will call the underlying callable and which, as a Future, will yield the callable's result as its result and provide for cancellation of the underlying task.**Since:** 1.6

### submit

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

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable)) Submits a Runnable task for execution and returns a Future representing that task. The Future's get method will return null upon *successful* completion.

**Specified by:**[submit](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**task - the task to submit **Returns:**a Future representing pending completion of the task

### submit

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

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable,%20T)) Submits a Runnable task for execution and returns a Future representing that task. The Future's get method will return the given result upon successful completion.

**Specified by:**[submit](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable,%20T)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**task - the task to submitresult - the result to return **Returns:**a Future representing pending completion of the task

### submit

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

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.util.concurrent.Callable)) Submits a value-returning task for execution and returns a Future representing the pending results of the task. The Future's get method will return the task's result upon successful completion.

If you would like to immediately block waiting for a task, you can use constructions of the form result = exec.submit(aCallable).get();

Note: The [Executors](http://docs.google.com/java/util/concurrent/Executors.html) class includes a set of methods that can convert some other common closure-like objects, for example, [PrivilegedAction](http://docs.google.com/java/security/PrivilegedAction.html) to [Callable](http://docs.google.com/java/util/concurrent/Callable.html) form so they can be submitted.

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

### invokeAny

public <T> T **invokeAny**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html),  
 [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html)

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection)) Executes the given tasks, returning the result of one that has completed successfully (i.e., without throwing an exception), if any do. Upon normal or exceptional return, tasks that have not completed are cancelled. The results of this method are undefined if the given collection is modified while this operation is in progress.

**Specified by:**[invokeAny](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**tasks - the collection of tasks **Returns:**the result returned by one of the tasks **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html) - if no task successfully completes

### invokeAny

public <T> T **invokeAny**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks,  
 long timeout,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html),  
 [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html),  
 [TimeoutException](http://docs.google.com/java/util/concurrent/TimeoutException.html)

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)) Executes the given tasks, returning the result of one that has completed successfully (i.e., without throwing an exception), if any do before the given timeout elapses. Upon normal or exceptional return, tasks that have not completed are cancelled. The results of this method are undefined if the given collection is modified while this operation is in progress.

**Specified by:**[invokeAny](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**tasks - the collection of taskstimeout - the maximum time to waitunit - the time unit of the timeout argument **Returns:**the result returned by one of the tasks. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting [ExecutionException](http://docs.google.com/java/util/concurrent/ExecutionException.html) - if no task successfully completes [TimeoutException](http://docs.google.com/java/util/concurrent/TimeoutException.html) - if the given timeout elapses before any task successfully completes

### invokeAll

public <T> [List](http://docs.google.com/java/util/List.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<T>> **invokeAll**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

**Description copied from interface:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection)) Executes the given tasks, returning a list of Futures holding their status and results when all complete. [Future.isDone()](http://docs.google.com/java/util/concurrent/Future.html#isDone()) is true for each element of the returned list. Note that a *completed* task could have terminated either normally or by throwing an exception. The results of this method are undefined if the given collection is modified while this operation is in progress.

**Specified by:**[invokeAll](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**tasks - the collection of tasks **Returns:**A list of Futures representing the tasks, in the same sequential order as produced by the iterator for the given task list, each of which has completed. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting, in which case unfinished tasks are cancelled.

### invokeAll

public <T> [List](http://docs.google.com/java/util/List.html)<[Future](http://docs.google.com/java/util/concurrent/Future.html)<T>> **invokeAll**([Collection](http://docs.google.com/java/util/Collection.html)<? extends [Callable](http://docs.google.com/java/util/concurrent/Callable.html)<T>> tasks,  
 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:** [**ExecutorService**](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)) Executes the given tasks, returning a list of Futures holding their status and results when all complete or the timeout expires, whichever happens first. [Future.isDone()](http://docs.google.com/java/util/concurrent/Future.html#isDone()) is true for each element of the returned list. Upon return, tasks that have not completed are cancelled. Note that a *completed* task could have terminated either normally or by throwing an exception. The results of this method are undefined if the given collection is modified while this operation is in progress.

**Specified by:**[invokeAll](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **Parameters:**tasks - the collection of taskstimeout - the maximum time to waitunit - the time unit of the timeout argument **Returns:**a list of Futures representing the tasks, in the same sequential order as produced by the iterator for the given task list. If the operation did not time out, each task will have completed. If it did time out, some of these tasks will not have completed. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting, in which case unfinished tasks are cancelled

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractExecutorService.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   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/ArrayBlockingQueue.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/AbstractExecutorService.html)    [**NO FRAMES**](http://docs.google.com/AbstractExecutorService.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

[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).