| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ScheduledThreadPoolExecutor.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/ScheduledFuture.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Semaphore.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ScheduledThreadPoolExecutor.html)    [**NO FRAMES**](http://docs.google.com/ScheduledThreadPoolExecutor.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#44sinio) |

## **java.util.concurrent**

Class ScheduledThreadPoolExecutor

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

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

public class **ScheduledThreadPoolExecutor**extends [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html)implements [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html)

A [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) that can additionally schedule commands to run after a given delay, or to execute periodically. This class is preferable to [Timer](http://docs.google.com/java/util/Timer.html) when multiple worker threads are needed, or when the additional flexibility or capabilities of [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) (which this class extends) are required.

Delayed tasks execute no sooner than they are enabled, but without any real-time guarantees about when, after they are enabled, they will commence. Tasks scheduled for exactly the same execution time are enabled in first-in-first-out (FIFO) order of submission.

While this class inherits from [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html), a few of the inherited tuning methods are not useful for it. In particular, because it acts as a fixed-sized pool using corePoolSize threads and an unbounded queue, adjustments to maximumPoolSize have no useful effect.

**Extension notes:** This class overrides [AbstractExecutorService](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html) submit methods to generate internal objects to control per-task delays and scheduling. To preserve functionality, any further overrides of these methods in subclasses must invoke superclass versions, which effectively disables additional task customization. However, this class provides alternative protected extension method decorateTask (one version each for Runnable and Callable) that can be used to customize the concrete task types used to execute commands entered via execute, submit, schedule, scheduleAtFixedRate, and scheduleWithFixedDelay. By default, a ScheduledThreadPoolExecutor uses a task type extending [FutureTask](http://docs.google.com/java/util/concurrent/FutureTask.html). However, this may be modified or replaced using subclasses of the form:

public class CustomScheduledExecutor extends ScheduledThreadPoolExecutor {  
  
 static class CustomTask<V> implements RunnableScheduledFuture<V> { ... }  
  
 protected <V> RunnableScheduledFuture<V> decorateTask(  
 Runnable r, RunnableScheduledFuture<V> task) {  
 return new CustomTask<V>(r, task);  
 }  
  
 protected <V> RunnableScheduledFuture<V> decorateTask(  
 Callable<V> c, RunnableScheduledFuture<V> task) {  
 return new CustomTask<V>(c, task);  
 }  
 // ... add constructors, etc.  
 }

**Since:** 1.5

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from class java.util.concurrent.**[**ThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) |
| --- |
| [ThreadPoolExecutor.AbortPolicy](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.AbortPolicy.html), [ThreadPoolExecutor.CallerRunsPolicy](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.CallerRunsPolicy.html), [ThreadPoolExecutor.DiscardOldestPolicy](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.DiscardOldestPolicy.html), [ThreadPoolExecutor.DiscardPolicy](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.DiscardPolicy.html) |

| **Constructor Summary** | |
| --- | --- |
| [**ScheduledThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#ScheduledThreadPoolExecutor(int))(int corePoolSize)            Creates a new ScheduledThreadPoolExecutor with the given core pool size. |
| [**ScheduledThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#ScheduledThreadPoolExecutor(int,%20java.util.concurrent.RejectedExecutionHandler))(int corePoolSize, [RejectedExecutionHandler](http://docs.google.com/java/util/concurrent/RejectedExecutionHandler.html) handler)            Creates a new ScheduledThreadPoolExecutor with the given initial parameters. |
| [**ScheduledThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#ScheduledThreadPoolExecutor(int,%20java.util.concurrent.ThreadFactory))(int corePoolSize, [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)            Creates a new ScheduledThreadPoolExecutor with the given initial parameters. |
| [**ScheduledThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#ScheduledThreadPoolExecutor(int,%20java.util.concurrent.ThreadFactory,%20java.util.concurrent.RejectedExecutionHandler))(int corePoolSize, [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory, [RejectedExecutionHandler](http://docs.google.com/java/util/concurrent/RejectedExecutionHandler.html) handler)            Creates a new ScheduledThreadPoolExecutor with the given initial parameters. |

| **Method Summary** | |
| --- | --- |
| protected   | <V> [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> | | --- | | [**decorateTask**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#decorateTask(java.util.concurrent.Callable,%20java.util.concurrent.RunnableScheduledFuture))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<V> callable, [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> task)            Modifies or replaces the task used to execute a callable. |
| protected   | <V> [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> | | --- | | [**decorateTask**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#decorateTask(java.lang.Runnable,%20java.util.concurrent.RunnableScheduledFuture))([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable, [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> task)            Modifies or replaces the task used to execute a runnable. |
| void | [**execute**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#execute(java.lang.Runnable))([Runnable](http://docs.google.com/java/lang/Runnable.html) command)            Executes command with zero required delay. |
| boolean | [**getContinueExistingPeriodicTasksAfterShutdownPolicy**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#getContinueExistingPeriodicTasksAfterShutdownPolicy())()            Gets the policy on whether to continue executing existing periodic tasks even when this executor has been shutdown. |
| boolean | [**getExecuteExistingDelayedTasksAfterShutdownPolicy**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#getExecuteExistingDelayedTasksAfterShutdownPolicy())()            Gets the policy on whether to execute existing delayed tasks even when this executor has been shutdown. |
| [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[Runnable](http://docs.google.com/java/lang/Runnable.html)> | [**getQueue**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#getQueue())()            Returns the task queue used by this executor. |
| boolean | [**remove**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#remove(java.lang.Runnable))([Runnable](http://docs.google.com/java/lang/Runnable.html) task)            Removes this task from the executor's internal queue if it is present, thus causing it not to be run if it has not already started. |
| | <V> [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<V> | | --- | | [**schedule**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#schedule(java.util.concurrent.Callable,%20long,%20java.util.concurrent.TimeUnit))([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<V> callable, long delay, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Creates and executes a ScheduledFuture that becomes enabled after the given delay. |
| [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> | [**schedule**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#schedule(java.lang.Runnable,%20long,%20java.util.concurrent.TimeUnit))([Runnable](http://docs.google.com/java/lang/Runnable.html) command, long delay, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Creates and executes a one-shot action that becomes enabled after the given delay. |
| [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> | [**scheduleAtFixedRate**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#scheduleAtFixedRate(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit))([Runnable](http://docs.google.com/java/lang/Runnable.html) command, long initialDelay, long period, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Creates and executes a periodic action that becomes enabled first after the given initial delay, and subsequently with the given period; that is executions will commence after initialDelay then initialDelay+period, then initialDelay + 2 \* period, and so on. |
| [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> | [**scheduleWithFixedDelay**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#scheduleWithFixedDelay(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit))([Runnable](http://docs.google.com/java/lang/Runnable.html) command, long initialDelay, long delay, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)            Creates and executes a periodic action that becomes enabled first after the given initial delay, and subsequently with the given delay between the termination of one execution and the commencement of the next. |
| void | [**setContinueExistingPeriodicTasksAfterShutdownPolicy**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean))(boolean value)            Sets the policy on whether to continue executing existing periodic tasks even when this executor has been shutdown. |
| void | [**setExecuteExistingDelayedTasksAfterShutdownPolicy**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean))(boolean value)            Sets the policy on whether to execute existing delayed tasks even when this executor has been shutdown. |
| void | [**shutdown**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#shutdown())()            Initiates an orderly shutdown in which previously submitted tasks are executed, but no new tasks will be accepted. |
| [List](http://docs.google.com/java/util/List.html)<[Runnable](http://docs.google.com/java/lang/Runnable.html)> | [**shutdownNow**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#shutdownNow())()            Attempts to stop all actively executing tasks, halts the processing of waiting tasks, and returns a list of the tasks that were awaiting execution. |
| | <T> [Future](http://docs.google.com/java/util/concurrent/Future.html)<T> | | --- | | [**submit**](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.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/ScheduledThreadPoolExecutor.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/ScheduledThreadPoolExecutor.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.util.concurrent.**[**ThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) |
| --- |
| [afterExecute](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#afterExecute(java.lang.Runnable,%20java.lang.Throwable)), [allowCoreThreadTimeOut](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#allowCoreThreadTimeOut(boolean)), [allowsCoreThreadTimeOut](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#allowsCoreThreadTimeOut()), [awaitTermination](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#awaitTermination(long,%20java.util.concurrent.TimeUnit)), [beforeExecute](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#beforeExecute(java.lang.Thread,%20java.lang.Runnable)), [finalize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#finalize()), [getActiveCount](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getActiveCount()), [getCompletedTaskCount](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getCompletedTaskCount()), [getCorePoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getCorePoolSize()), [getKeepAliveTime](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getKeepAliveTime(java.util.concurrent.TimeUnit)), [getLargestPoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getLargestPoolSize()), [getMaximumPoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getMaximumPoolSize()), [getPoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getPoolSize()), [getRejectedExecutionHandler](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getRejectedExecutionHandler()), [getTaskCount](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getTaskCount()), [getThreadFactory](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getThreadFactory()), [isShutdown](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#isShutdown()), [isTerminated](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#isTerminated()), [isTerminating](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#isTerminating()), [prestartAllCoreThreads](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#prestartAllCoreThreads()), [prestartCoreThread](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#prestartCoreThread()), [purge](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#purge()), [setCorePoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#setCorePoolSize(int)), [setKeepAliveTime](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#setKeepAliveTime(long,%20java.util.concurrent.TimeUnit)), [setMaximumPoolSize](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#setMaximumPoolSize(int)), [setRejectedExecutionHandler](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#setRejectedExecutionHandler(java.util.concurrent.RejectedExecutionHandler)), [setThreadFactory](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#setThreadFactory(java.util.concurrent.ThreadFactory)), [terminated](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#terminated()) |

| **Methods inherited from class java.util.concurrent.**[**AbstractExecutorService**](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html) |
| --- |
| [invokeAll](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAll(java.util.Collection)), [invokeAll](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAll(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)), [invokeAny](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAny(java.util.Collection)), [invokeAny](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#invokeAny(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)), [newTaskFor](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#newTaskFor(java.util.concurrent.Callable)), [newTaskFor](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#newTaskFor(java.lang.Runnable,%20T)) |

| **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)), [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)), [invokeAll](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection)), [invokeAll](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAll(java.util.Collection,%20long,%20java.util.concurrent.TimeUnit)), [invokeAny](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection)), [invokeAny](http://docs.google.com/java/util/concurrent/ExecutorService.html#invokeAny(java.util.Collection,%20long,%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()) |

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

### ScheduledThreadPoolExecutor

public **ScheduledThreadPoolExecutor**(int corePoolSize)

Creates a new ScheduledThreadPoolExecutor with the given core pool size.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idle **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0

### ScheduledThreadPoolExecutor

public **ScheduledThreadPoolExecutor**(int corePoolSize,  
 [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory)

Creates a new ScheduledThreadPoolExecutor with the given initial parameters.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idlethreadFactory - the factory to use when the executor creates a new thread **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0 [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory is null

### ScheduledThreadPoolExecutor

public **ScheduledThreadPoolExecutor**(int corePoolSize,  
 [RejectedExecutionHandler](http://docs.google.com/java/util/concurrent/RejectedExecutionHandler.html) handler)

Creates a new ScheduledThreadPoolExecutor with the given initial parameters.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idlehandler - the handler to use when execution is blocked because the thread bounds and queue capacities are reached **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0 [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if handler is null

### ScheduledThreadPoolExecutor

public **ScheduledThreadPoolExecutor**(int corePoolSize,  
 [ThreadFactory](http://docs.google.com/java/util/concurrent/ThreadFactory.html) threadFactory,  
 [RejectedExecutionHandler](http://docs.google.com/java/util/concurrent/RejectedExecutionHandler.html) handler)

Creates a new ScheduledThreadPoolExecutor with the given initial parameters.

**Parameters:**corePoolSize - the number of threads to keep in the pool, even if they are idlethreadFactory - the factory to use when the executor creates a new threadhandler - the handler to use when execution is blocked because the thread bounds and queue capacities are reached. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if corePoolSize < 0 [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if threadFactory or handler is null

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

### remove

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

**Description copied from class:** [**ThreadPoolExecutor**](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#remove(java.lang.Runnable)) Removes this task from the executor's internal queue if it is present, thus causing it not to be run if it has not already started.

This method may be useful as one part of a cancellation scheme. It may fail to remove tasks that have been converted into other forms before being placed on the internal queue. For example, a task entered using submit might be converted into a form that maintains Future status. However, in such cases, method [ThreadPoolExecutor.purge()](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#purge()) may be used to remove those Futures that have been cancelled.

**Overrides:**[remove](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#remove(java.lang.Runnable)) in class [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) **Parameters:**task - the task to remove **Returns:**true if the task was removed

### decorateTask

protected <V> [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> **decorateTask**([Runnable](http://docs.google.com/java/lang/Runnable.html) runnable,  
 [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> task)

Modifies or replaces the task used to execute a runnable. This method can be used to override the concrete class used for managing internal tasks. The default implementation simply returns the given task.

**Parameters:**runnable - the submitted Runnabletask - the task created to execute the runnable **Returns:**a task that can execute the runnable**Since:** 1.6

### decorateTask

protected <V> [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> **decorateTask**([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<V> callable,  
 [RunnableScheduledFuture](http://docs.google.com/java/util/concurrent/RunnableScheduledFuture.html)<V> task)

Modifies or replaces the task used to execute a callable. This method can be used to override the concrete class used for managing internal tasks. The default implementation simply returns the given task.

**Parameters:**callable - the submitted Callabletask - the task created to execute the callable **Returns:**a task that can execute the callable**Since:** 1.6

### schedule

public [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> **schedule**([Runnable](http://docs.google.com/java/lang/Runnable.html) command,  
 long delay,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

**Description copied from interface:** [**ScheduledExecutorService**](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#schedule(java.lang.Runnable,%20long,%20java.util.concurrent.TimeUnit)) Creates and executes a one-shot action that becomes enabled after the given delay.

**Specified by:**[schedule](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#schedule(java.lang.Runnable,%20long,%20java.util.concurrent.TimeUnit)) in interface [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **Parameters:**command - the task to executedelay - the time from now to delay executionunit - the time unit of the delay parameter **Returns:**a ScheduledFuture representing pending completion of the task and whose get() method will return null upon completion

### schedule

public <V> [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<V> **schedule**([Callable](http://docs.google.com/java/util/concurrent/Callable.html)<V> callable,  
 long delay,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

**Description copied from interface:** [**ScheduledExecutorService**](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#schedule(java.util.concurrent.Callable,%20long,%20java.util.concurrent.TimeUnit)) Creates and executes a ScheduledFuture that becomes enabled after the given delay.

**Specified by:**[schedule](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#schedule(java.util.concurrent.Callable,%20long,%20java.util.concurrent.TimeUnit)) in interface [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **Parameters:**callable - the function to executedelay - the time from now to delay executionunit - the time unit of the delay parameter **Returns:**a ScheduledFuture that can be used to extract result or cancel

### scheduleAtFixedRate

public [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> **scheduleAtFixedRate**([Runnable](http://docs.google.com/java/lang/Runnable.html) command,  
 long initialDelay,  
 long period,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

**Description copied from interface:** [**ScheduledExecutorService**](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#scheduleAtFixedRate(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit)) Creates and executes a periodic action that becomes enabled first after the given initial delay, and subsequently with the given period; that is executions will commence after initialDelay then initialDelay+period, then initialDelay + 2 \* period, and so on. If any execution of the task encounters an exception, subsequent executions are suppressed. Otherwise, the task will only terminate via cancellation or termination of the executor. If any execution of this task takes longer than its period, then subsequent executions may start late, but will not concurrently execute.

**Specified by:**[scheduleAtFixedRate](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#scheduleAtFixedRate(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit)) in interface [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **Parameters:**command - the task to executeinitialDelay - the time to delay first executionperiod - the period between successive executionsunit - the time unit of the initialDelay and period parameters **Returns:**a ScheduledFuture representing pending completion of the task, and whose get() method will throw an exception upon cancellation

### scheduleWithFixedDelay

public [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<?> **scheduleWithFixedDelay**([Runnable](http://docs.google.com/java/lang/Runnable.html) command,  
 long initialDelay,  
 long delay,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) unit)

**Description copied from interface:** [**ScheduledExecutorService**](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#scheduleWithFixedDelay(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit)) Creates and executes a periodic action that becomes enabled first after the given initial delay, and subsequently with the given delay between the termination of one execution and the commencement of the next. If any execution of the task encounters an exception, subsequent executions are suppressed. Otherwise, the task will only terminate via cancellation or termination of the executor.

**Specified by:**[scheduleWithFixedDelay](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html#scheduleWithFixedDelay(java.lang.Runnable,%20long,%20long,%20java.util.concurrent.TimeUnit)) in interface [ScheduledExecutorService](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.html) **Parameters:**command - the task to executeinitialDelay - the time to delay first executiondelay - the delay between the termination of one execution and the commencement of the nextunit - the time unit of the initialDelay and delay parameters **Returns:**a ScheduledFuture representing pending completion of the task, and whose get() method will throw an exception upon cancellation

### execute

public void **execute**([Runnable](http://docs.google.com/java/lang/Runnable.html) command)

Executes command with zero required delay. This has effect equivalent to schedule(command, 0, anyUnit). Note that inspections of the queue and of the list returned by shutdownNow will access the zero-delayed [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html), not the command itself.

**Specified by:**[execute](http://docs.google.com/java/util/concurrent/Executor.html#execute(java.lang.Runnable)) in interface [Executor](http://docs.google.com/java/util/concurrent/Executor.html)**Overrides:**[execute](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#execute(java.lang.Runnable)) in class [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) **Parameters:**command - the task to execute **Throws:** [RejectedExecutionException](http://docs.google.com/java/util/concurrent/RejectedExecutionException.html) - at discretion of RejectedExecutionHandler, if task cannot be accepted for execution because the executor has been shut down. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if command is null

### 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)**Overrides:**[submit](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.lang.Runnable)) in class [AbstractExecutorService](http://docs.google.com/java/util/concurrent/AbstractExecutorService.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)**Overrides:**[submit](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.lang.Runnable,%20T)) in class [AbstractExecutorService](http://docs.google.com/java/util/concurrent/AbstractExecutorService.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)**Overrides:**[submit](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html#submit(java.util.concurrent.Callable)) in class [AbstractExecutorService](http://docs.google.com/java/util/concurrent/AbstractExecutorService.html) **Parameters:**task - the task to submit **Returns:**a Future representing pending completion of the task

### setContinueExistingPeriodicTasksAfterShutdownPolicy

public void **setContinueExistingPeriodicTasksAfterShutdownPolicy**(boolean value)

Sets the policy on whether to continue executing existing periodic tasks even when this executor has been shutdown. In this case, these tasks will only terminate upon shutdownNow, or after setting the policy to false when already shutdown. This value is by default false.

**Parameters:**value - if true, continue after shutdown, else don't.**See Also:**[getContinueExistingPeriodicTasksAfterShutdownPolicy()](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#getContinueExistingPeriodicTasksAfterShutdownPolicy())

### getContinueExistingPeriodicTasksAfterShutdownPolicy

public boolean **getContinueExistingPeriodicTasksAfterShutdownPolicy**()

Gets the policy on whether to continue executing existing periodic tasks even when this executor has been shutdown. In this case, these tasks will only terminate upon shutdownNow or after setting the policy to false when already shutdown. This value is by default false.

**Returns:**true if will continue after shutdown**See Also:**[setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean)](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#setContinueExistingPeriodicTasksAfterShutdownPolicy(boolean))

### setExecuteExistingDelayedTasksAfterShutdownPolicy

public void **setExecuteExistingDelayedTasksAfterShutdownPolicy**(boolean value)

Sets the policy on whether to execute existing delayed tasks even when this executor has been shutdown. In this case, these tasks will only terminate upon shutdownNow, or after setting the policy to false when already shutdown. This value is by default true.

**Parameters:**value - if true, execute after shutdown, else don't.**See Also:**[getExecuteExistingDelayedTasksAfterShutdownPolicy()](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#getExecuteExistingDelayedTasksAfterShutdownPolicy())

### getExecuteExistingDelayedTasksAfterShutdownPolicy

public boolean **getExecuteExistingDelayedTasksAfterShutdownPolicy**()

Gets the policy on whether to execute existing delayed tasks even when this executor has been shutdown. In this case, these tasks will only terminate upon shutdownNow, or after setting the policy to false when already shutdown. This value is by default true.

**Returns:**true if will execute after shutdown**See Also:**[setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean)](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html#setExecuteExistingDelayedTasksAfterShutdownPolicy(boolean))

### shutdown

public void **shutdown**()

Initiates an orderly shutdown in which previously submitted tasks are executed, but no new tasks will be accepted. If the ExecuteExistingDelayedTasksAfterShutdownPolicy has been set false, existing delayed tasks whose delays have not yet elapsed are cancelled. And unless the ContinueExistingPeriodicTasksAfterShutdownPolicy has been set true, future executions of existing periodic tasks will be cancelled.

**Specified by:**[shutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdown()) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html)**Overrides:**[shutdown](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#shutdown()) in class [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html)

### shutdownNow

public [List](http://docs.google.com/java/util/List.html)<[Runnable](http://docs.google.com/java/lang/Runnable.html)> **shutdownNow**()

Attempts to stop all actively executing tasks, halts the processing of waiting tasks, and returns a list of the tasks that were awaiting execution.

There are no guarantees beyond best-effort attempts to stop processing actively executing tasks. This implementation cancels tasks via [Thread.interrupt()](http://docs.google.com/java/lang/Thread.html#interrupt()), so any task that fails to respond to interrupts may never terminate.

**Specified by:**[shutdownNow](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdownNow()) in interface [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html)**Overrides:**[shutdownNow](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#shutdownNow()) in class [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) **Returns:**list of tasks that never commenced execution. Each element of this list is a [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html), including those tasks submitted using execute, which are for scheduling purposes used as the basis of a zero-delay ScheduledFuture. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and shutting down this ExecutorService may manipulate threads that the caller is not permitted to modify because it does not hold [RuntimePermission](http://docs.google.com/java/lang/RuntimePermission.html)("modifyThread"), or the security manager's checkAccess method denies access.

### getQueue

public [BlockingQueue](http://docs.google.com/java/util/concurrent/BlockingQueue.html)<[Runnable](http://docs.google.com/java/lang/Runnable.html)> **getQueue**()

Returns the task queue used by this executor. Each element of this queue is a [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html), including those tasks submitted using execute which are for scheduling purposes used as the basis of a zero-delay ScheduledFuture. Iteration over this queue is *not* guaranteed to traverse tasks in the order in which they will execute.

**Overrides:**[getQueue](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html#getQueue()) in class [ThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ThreadPoolExecutor.html) **Returns:**the task 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/ScheduledThreadPoolExecutor.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/ScheduledFuture.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/Semaphore.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/ScheduledThreadPoolExecutor.html)    [**NO FRAMES**](http://docs.google.com/ScheduledThreadPoolExecutor.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#44sinio) |

[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).