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

## **java.util.concurrent**

Interface ScheduledExecutorService

**All Superinterfaces:** [Executor](http://docs.google.com/java/util/concurrent/Executor.html), [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) **All Known Implementing Classes:** [ScheduledThreadPoolExecutor](http://docs.google.com/java/util/concurrent/ScheduledThreadPoolExecutor.html)

public interface **ScheduledExecutorService**extends [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html)

An [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) that can schedule commands to run after a given delay, or to execute periodically.

The schedule methods create tasks with various delays and return a task object that can be used to cancel or check execution. The scheduleAtFixedRate and scheduleWithFixedDelay methods create and execute tasks that run periodically until cancelled.

Commands submitted using the [Executor.execute(java.lang.Runnable)](http://docs.google.com/java/util/concurrent/Executor.html#execute(java.lang.Runnable)) and [ExecutorService](http://docs.google.com/java/util/concurrent/ExecutorService.html) submit methods are scheduled with a requested delay of zero. Zero and negative delays (but not periods) are also allowed in schedule methods, and are treated as requests for immediate execution.

All schedule methods accept *relative* delays and periods as arguments, not absolute times or dates. It is a simple matter to transform an absolute time represented as a [Date](http://docs.google.com/java/util/Date.html) to the required form. For example, to schedule at a certain future date, you can use: schedule(task, date.getTime() - System.currentTimeMillis(), TimeUnit.MILLISECONDS). Beware however that expiration of a relative delay need not coincide with the current Date at which the task is enabled due to network time synchronization protocols, clock drift, or other factors. The [Executors](http://docs.google.com/java/util/concurrent/Executors.html) class provides convenient factory methods for the ScheduledExecutorService implementations provided in this package.

### Usage Example

Here is a class with a method that sets up a ScheduledExecutorService to beep every ten seconds for an hour:

import static java.util.concurrent.TimeUnit.\*;  
 class BeeperControl {  
 private final ScheduledExecutorService scheduler =  
 Executors.newScheduledThreadPool(1);  
  
 public void beepForAnHour() {  
 final Runnable beeper = new Runnable() {  
 public void run() { System.out.println("beep"); }  
 };  
 final ScheduledFuture<?> beeperHandle =  
 scheduler.scheduleAtFixedRate(beeper, 10, 10, SECONDS);  
 scheduler.schedule(new Runnable() {  
 public void run() { beeperHandle.cancel(true); }  
 }, 60 \* 60, SECONDS);  
 }  
 }

**Since:** 1.5

| **Method Summary** | |
| --- | --- |
| | <V> [ScheduledFuture](http://docs.google.com/java/util/concurrent/ScheduledFuture.html)<V> | | --- | | [**schedule**](http://docs.google.com/java/util/concurrent/ScheduledExecutorService.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/ScheduledExecutorService.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/ScheduledExecutorService.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/ScheduledExecutorService.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. |

| **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()), [shutdown](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdown()), [shutdownNow](http://docs.google.com/java/util/concurrent/ExecutorService.html#shutdownNow()), [submit](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.util.concurrent.Callable)), [submit](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable)), [submit](http://docs.google.com/java/util/concurrent/ExecutorService.html#submit(java.lang.Runnable,%20T)) |

| **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)) |

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

### schedule

[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)

Creates and executes a one-shot action that becomes enabled after the given delay.

**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 **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 command is null

### schedule

<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)

Creates and executes a ScheduledFuture that becomes enabled after the given delay.

**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 **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 callable is null

### scheduleAtFixedRate

[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)

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.

**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 **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 command is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if period less than or equal to zero

### scheduleWithFixedDelay

[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)

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.

**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 **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 command is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if delay less than or equal to zero

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

[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).