public class

Timer

extends Object

Summary: Ctors | Methods | Inherited Methods | [Expand All]

Added in API level 1

java.lang.Object
    java.util.Timer

Class Overview

Timers schedule one-shot or recurring `tasks` for execution. Prefer `ScheduledThreadPoolExecutor` for new code.

Each timer has one thread on which tasks are executed sequentially. When this thread is busy running a task, runnable tasks may be subject to delays.

One-shot are scheduled to run at an absolute time or after a relative delay.

Recurring tasks are scheduled with either a fixed period or a fixed rate:

- With the default **fixed-period execution**, each successive run of a task is scheduled relative to the start time of the previous run, so two runs are never fired closer together in time than the

specified `period`.

- With **fixed-rate execution**, the start time of each successive run of a task is scheduled without regard for when the previous run took place. This may result in a series of bunched-up runs (one launched immediately after another) if delays prevent the timer from starting tasks on time.

When a timer is no longer needed, users should call `cancel()`, which releases the timer's thread and other resources. Timers not explicitly cancelled may hold resources indefinitely.

This class does not offer guarantees about the real-time nature of task scheduling. Multiple threads can share a single timer without synchronization.

Summary

Public Constructors	
	Timer (String name, boolean isDaemon) Creates a new named `Timer` which may be specified to be run as a daemon thread.
	Timer (String name) Creates a new named `Timer` which does not run as a daemon thread.
	Timer (boolean isDaemon) Creates a new `Timer` which may be specified to be run as a daemon thread.
	Timer () Creates a new non-daemon `Timer`.

Public Methods

void	cancel () Cancels the `Timer` and all scheduled tasks.
int	purge () Removes all canceled tasks from the task queue.
void	schedule (TimerTask task, Date when, long period) Schedule a task for repeated fixed-delay execution after a specific time has been reached.
void	schedule (TimerTask task, long delay, long period) Schedule a task for repeated fixed-delay execution after a specific delay.
void	schedule (TimerTask task, Date when) Schedule a task for single execution.
void	schedule (TimerTask task, long delay) Schedule a task for single execution after a specified delay.
void	scheduleAtFixedRate (TimerTask task, long delay, long period) Schedule a task for repeated fixed-rate execution after a specific delay has passed.
void	scheduleAtFixedRate (TimerTask task, Date when, long period) Schedule a task for repeated fixed-rate execution after a specific time has been reached.

Inherited Methods [Expand]

- From class java.lang.Object

Public Constructors

public **Timer** (String name, boolean isDaemon)

Added in API level 1

Creates a new named `Timer` which may be specified to be run as a daemon thread.

Parameters

name	the name of the `Timer`.
isDaemon	true if `Timer`'s thread should be a daemon thread.

Throws

NullPointerException	is `name` is `null`

public **Timer** (String name)

Added in API level 1

Creates a new named `Timer` which does not run as a daemon thread.

Parameters

name	the name of the Timer.

Throws

NullPointerException	is `name` is `null`

public **Timer** (boolean isDaemon)

Added in API level 1

Creates a new `Timer` which may be specified to be run as a daemon thread.

Parameters

isDaemon `true` if the `Timer`'s thread should be a daemon thread.

public **Timer** ()

Added in API level 1

Creates a new non-daemon `Timer`.

Public Methods

public void **cancel** ()

Added in API level 1

Cancels the `Timer` and all scheduled tasks. If there is a currently running task it is not affected. No more tasks may be scheduled on this `Timer`. Subsequent calls do nothing.

public int **purge** ()

Added in API level 1

Removes all canceled tasks from the task queue. If there are no other references on the tasks, then after this call they are free to be garbage collected.

Returns

the number of canceled tasks that were removed from the task queue.

public void **schedule** (TimerTask task, Date when, long period)

Added in API level 1

Schedule a task for repeated fixed-delay execution after a specific time has been reached.

Parameters

task the task to schedule.

when	time of first execution.
period	amount of time in milliseconds between subsequent executions.

Throws

IllegalArgumentException	if `when.getTime() < 0` or `period <= 0`.
IllegalStateException	if the `Timer` has been canceled, or if the task has been scheduled or canceled.

public void **schedule** (TimerTask task, long delay, long period)

Added in API level 1

Schedule a task for repeated fixed-delay execution after a specific delay.

Parameters

task	the task to schedule.
delay	amount of time in milliseconds before first execution.
period	amount of time in milliseconds between subsequent executions.

Throws

IllegalArgumentException	if `delay < 0` or `period <= 0`.
IllegalStateException	if the `Timer` has been canceled, or if the task has been scheduled or canceled.

public void **schedule** (TimerTask task, Date when)

Added in API level 1

Schedule a task for single execution. If `when` is less than the current time, it will be scheduled to be executed as soon as possible.

Parameters

task the task to schedule.

when time of execution.

Throws

IllegalArgumentException if `when.getTime() < 0`.

IllegalStateException if the `Timer` has been canceled, or if the task has been scheduled or canceled.

public void **schedule** (TimerTask task, long delay)

Added in API level 1

Schedule a task for single execution after a specified delay.

Parameters

task the task to schedule.

delay amount of time in milliseconds before execution.

Throws

IllegalArgumentException if `delay < 0`.

IllegalStateException if the `Timer` has been canceled, or if the task has been scheduled or canceled.

public void **scheduleAtFixedRate** (TimerTask task, long delay, long period)

Added in API level 1

Schedule a task for repeated fixed-rate execution after a specific delay has passed.

Parameters

task the task to schedule.

delay	amount of time in milliseconds before first execution.
period	amount of time in milliseconds between subsequent executions.

Throws

IllegalArgumentException	if `delay < 0` or `period <= 0`.
IllegalStateException	if the `Timer` has been canceled, or if the task has been scheduled or canceled.

public void **scheduleAtFixedRate** (TimerTask task, Date when, long period)

Added in API level 1

Schedule a task for repeated fixed-rate execution after a specific time has been reached.

Parameters

task	the task to schedule.
when	time of first execution.
period	amount of time in milliseconds between subsequent executions.

Throws

IllegalArgumentException	if `when.getTime() < 0` or `period <= 0`.
IllegalStateException	if the `Timer` has been canceled, or if the task has been scheduled or canceled.