| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Timer.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/javax/swing/SwingWorker.StateValue.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/ToolTipManager.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/Timer.html)    [**NO FRAMES**](http://docs.google.com/Timer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

## **javax.swing**

Class Timer

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.Timer**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html)

public class **Timer**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

Fires one or more ActionEvents at specified intervals. An example use is an animation object that uses a Timer as the trigger for drawing its frames.

Setting up a timer involves creating a Timer object, registering one or more action listeners on it, and starting the timer using the start method. For example, the following code creates and starts a timer that fires an action event once per second (as specified by the first argument to the Timer constructor). The second argument to the Timer constructor specifies a listener to receive the timer's action events.

int delay = 1000; //milliseconds  
 ActionListener taskPerformer = new ActionListener() {  
 public void actionPerformed(ActionEvent evt) {  
 *//...Perform a task...*  
 }  
 };  
 new Timer(delay, taskPerformer).start();

Timers are constructed by specifying both a delay parameter and an ActionListener. The delay parameter is used to set both the initial delay and the delay between event firing, in milliseconds. Once the timer has been started, it waits for the initial delay before firing its first ActionEvent to registered listeners. After this first event, it continues to fire events every time the between-event delay has elapsed, until it is stopped.

After construction, the initial delay and the between-event delay can be changed independently, and additional ActionListeners may be added.

If you want the timer to fire only the first time and then stop, invoke setRepeats(false) on the timer.

Although all Timers perform their waiting using a single, shared thread (created by the first Timer object that executes), the action event handlers for Timers execute on another thread -- the event-dispatching thread. This means that the action handlers for Timers can safely perform operations on Swing components. However, it also means that the handlers must execute quickly to keep the GUI responsive.

In v 1.3, another Timer class was added to the Java platform: java.util.Timer. Both it and javax.swing.Timer provide the same basic functionality, but java.util.Timer is more general and has more features. The javax.swing.Timer has two features that can make it a little easier to use with GUIs. First, its event handling metaphor is familiar to GUI programmers and can make dealing with the event-dispatching thread a bit simpler. Second, its automatic thread sharing means that you don't have to take special steps to avoid spawning too many threads. Instead, your timer uses the same thread used to make cursors blink, tool tips appear, and so on.

You can find further documentation and several examples of using timers by visiting [How to Use Timers](http://java.sun.com/docs/books/tutorial/uiswing/misc/timer.html), a section in *The Java Tutorial.* For more examples and help in choosing between this Timer class and java.util.Timer, see [Using Timers in Swing Applications](http://java.sun.com/products/jfc/tsc/articles/timer/), an article in *The Swing Connection.*

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**See Also:**[java.util.Timer](http://docs.google.com/java/util/Timer.html)

| **Field Summary** | |
| --- | --- |
| protected  [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) | [**listenerList**](http://docs.google.com/javax/swing/Timer.html#listenerList) |

| **Constructor Summary** | |
| --- | --- |
| [**Timer**](http://docs.google.com/javax/swing/Timer.html#Timer(int,%20java.awt.event.ActionListener))(int delay, [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)            Creates a Timer and initializes both the initial delay and between-event delay to delay milliseconds. |

| **Method Summary** | |
| --- | --- |
| void | [**addActionListener**](http://docs.google.com/javax/swing/Timer.html#addActionListener(java.awt.event.ActionListener))([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)            Adds an action listener to the Timer. |
| protected  void | [**fireActionPerformed**](http://docs.google.com/javax/swing/Timer.html#fireActionPerformed(java.awt.event.ActionEvent))([ActionEvent](http://docs.google.com/java/awt/event/ActionEvent.html) e)            Notifies all listeners that have registered interest for notification on this event type. |
| [String](http://docs.google.com/java/lang/String.html) | [**getActionCommand**](http://docs.google.com/javax/swing/Timer.html#getActionCommand())()            Returns the string that will be delivered as the action command in ActionEvents fired by this timer. |
| [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html)[] | [**getActionListeners**](http://docs.google.com/javax/swing/Timer.html#getActionListeners())()            Returns an array of all the action listeners registered on this timer. |
| int | [**getDelay**](http://docs.google.com/javax/swing/Timer.html#getDelay())()            Returns the delay, in milliseconds, between firings of action events. |
| int | [**getInitialDelay**](http://docs.google.com/javax/swing/Timer.html#getInitialDelay())()            Returns the Timer's initial delay. |
| | <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)>  T[] | | --- | | [**getListeners**](http://docs.google.com/javax/swing/Timer.html#getListeners(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)            Returns an array of all the objects currently registered as *Foo*Listeners upon this Timer. |
| static boolean | [**getLogTimers**](http://docs.google.com/javax/swing/Timer.html#getLogTimers())()            Returns true if logging is enabled. |
| boolean | [**isCoalesce**](http://docs.google.com/javax/swing/Timer.html#isCoalesce())()            Returns true if the Timer coalesces multiple pending action events. |
| boolean | [**isRepeats**](http://docs.google.com/javax/swing/Timer.html#isRepeats())()            Returns true (the default) if the Timer will send an action event to its listeners multiple times. |
| boolean | [**isRunning**](http://docs.google.com/javax/swing/Timer.html#isRunning())()            Returns true if the Timer is running. |
| void | [**removeActionListener**](http://docs.google.com/javax/swing/Timer.html#removeActionListener(java.awt.event.ActionListener))([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)            Removes the specified action listener from the Timer. |
| void | [**restart**](http://docs.google.com/javax/swing/Timer.html#restart())()            Restarts the Timer, canceling any pending firings and causing it to fire with its initial delay. |
| void | [**setActionCommand**](http://docs.google.com/javax/swing/Timer.html#setActionCommand(java.lang.String))([String](http://docs.google.com/java/lang/String.html) command)            Sets the string that will be delivered as the action command in ActionEvents fired by this timer. |
| void | [**setCoalesce**](http://docs.google.com/javax/swing/Timer.html#setCoalesce(boolean))(boolean flag)            Sets whether the Timer coalesces multiple pending ActionEvent firings. |
| void | [**setDelay**](http://docs.google.com/javax/swing/Timer.html#setDelay(int))(int delay)            Sets the Timer's between-event delay, the number of milliseconds between successive action events. |
| void | [**setInitialDelay**](http://docs.google.com/javax/swing/Timer.html#setInitialDelay(int))(int initialDelay)            Sets the Timer's initial delay, the time in milliseconds to wait after the timer is started before firing the first event. |
| static void | [**setLogTimers**](http://docs.google.com/javax/swing/Timer.html#setLogTimers(boolean))(boolean flag)            Enables or disables the timer log. |
| void | [**setRepeats**](http://docs.google.com/javax/swing/Timer.html#setRepeats(boolean))(boolean flag)            If flag is false, instructs the Timer to send only one action event to its listeners. |
| void | [**start**](http://docs.google.com/javax/swing/Timer.html#start())()            Starts the Timer, causing it to start sending action events to its listeners. |
| void | [**stop**](http://docs.google.com/javax/swing/Timer.html#stop())()            Stops the Timer, causing it to stop sending action events to its listeners. |

| **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)) |

| **Field Detail** |
| --- |

### listenerList

protected [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) **listenerList**

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

### Timer

public **Timer**(int delay,  
 [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)

Creates a Timer and initializes both the initial delay and between-event delay to delay milliseconds. If delay is less than or equal to zero, the timer fires as soon as it is started. If listener is not null, it's registered as an action listener on the timer.

**Parameters:**delay - milliseconds for the initial and between-event delaylistener - an initial listener; can be null**See Also:**[addActionListener(java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#addActionListener(java.awt.event.ActionListener)), [setInitialDelay(int)](http://docs.google.com/javax/swing/Timer.html#setInitialDelay(int)), [setRepeats(boolean)](http://docs.google.com/javax/swing/Timer.html#setRepeats(boolean))

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

### addActionListener

public void **addActionListener**([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)

Adds an action listener to the Timer.

**Parameters:**listener - the listener to add**See Also:**[Timer(int, java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#Timer(int,%20java.awt.event.ActionListener))

### removeActionListener

public void **removeActionListener**([ActionListener](http://docs.google.com/java/awt/event/ActionListener.html) listener)

Removes the specified action listener from the Timer.

**Parameters:**listener - the listener to remove

### getActionListeners

public [ActionListener](http://docs.google.com/java/awt/event/ActionListener.html)[] **getActionListeners**()

Returns an array of all the action listeners registered on this timer.

**Returns:**all of the timer's ActionListeners or an empty array if no action listeners are currently registered**Since:** 1.4 **See Also:**[addActionListener(java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#addActionListener(java.awt.event.ActionListener)), [removeActionListener(java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#removeActionListener(java.awt.event.ActionListener))

### fireActionPerformed

protected void **fireActionPerformed**([ActionEvent](http://docs.google.com/java/awt/event/ActionEvent.html) e)

Notifies all listeners that have registered interest for notification on this event type.

**Parameters:**e - the action event to fire**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### getListeners

public <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)> T[] **getListeners**([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)

Returns an array of all the objects currently registered as *Foo*Listeners upon this Timer. *Foo*Listeners are registered using the add*Foo*Listener method.

You can specify the listenerType argument with a class literal, such as *Foo*Listener.class. For example, you can query a Timer instance t for its action listeners with the following code:

ActionListener[] als = (ActionListener[])(t.getListeners(ActionListener.class));

If no such listeners exist, this method returns an empty array.

**Parameters:**listenerType - the type of listeners requested; this parameter should specify an interface that descends from java.util.EventListener **Returns:**an array of all objects registered as *Foo*Listeners on this timer, or an empty array if no such listeners have been added **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if listenerType doesn't specify a class or interface that implements java.util.EventListener**Since:** 1.3 **See Also:**[getActionListeners()](http://docs.google.com/javax/swing/Timer.html#getActionListeners()), [addActionListener(java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#addActionListener(java.awt.event.ActionListener)), [removeActionListener(java.awt.event.ActionListener)](http://docs.google.com/javax/swing/Timer.html#removeActionListener(java.awt.event.ActionListener))

### setLogTimers

public static void **setLogTimers**(boolean flag)

Enables or disables the timer log. When enabled, a message is posted to System.out whenever the timer goes off.

**Parameters:**flag - true to enable logging**See Also:**[getLogTimers()](http://docs.google.com/javax/swing/Timer.html#getLogTimers())

### getLogTimers

public static boolean **getLogTimers**()

Returns true if logging is enabled.

**Returns:**true if logging is enabled; otherwise, false**See Also:**[setLogTimers(boolean)](http://docs.google.com/javax/swing/Timer.html#setLogTimers(boolean))

### setDelay

public void **setDelay**(int delay)

Sets the Timer's between-event delay, the number of milliseconds between successive action events. This does not affect the initial delay property, which can be set by the setInitialDelay method.

**Parameters:**delay - the delay in milliseconds**See Also:**[setInitialDelay(int)](http://docs.google.com/javax/swing/Timer.html#setInitialDelay(int))

### getDelay

public int **getDelay**()

Returns the delay, in milliseconds, between firings of action events.

**See Also:**[setDelay(int)](http://docs.google.com/javax/swing/Timer.html#setDelay(int)), [getInitialDelay()](http://docs.google.com/javax/swing/Timer.html#getInitialDelay())

### setInitialDelay

public void **setInitialDelay**(int initialDelay)

Sets the Timer's initial delay, the time in milliseconds to wait after the timer is started before firing the first event. Upon construction, this is set to be the same as the between-event delay, but then its value is independent and remains unaffected by changes to the between-event delay.

**Parameters:**initialDelay - the initial delay, in milliseconds**See Also:**[setDelay(int)](http://docs.google.com/javax/swing/Timer.html#setDelay(int))

### getInitialDelay

public int **getInitialDelay**()

Returns the Timer's initial delay.

**See Also:**[setInitialDelay(int)](http://docs.google.com/javax/swing/Timer.html#setInitialDelay(int)), [setDelay(int)](http://docs.google.com/javax/swing/Timer.html#setDelay(int))

### setRepeats

public void **setRepeats**(boolean flag)

If flag is false, instructs the Timer to send only one action event to its listeners.

**Parameters:**flag - specify false to make the timer stop after sending its first action event

### isRepeats

public boolean **isRepeats**()

Returns true (the default) if the Timer will send an action event to its listeners multiple times.

**See Also:**[setRepeats(boolean)](http://docs.google.com/javax/swing/Timer.html#setRepeats(boolean))

### setCoalesce

public void **setCoalesce**(boolean flag)

Sets whether the Timer coalesces multiple pending ActionEvent firings. A busy application may not be able to keep up with a Timer's event generation, causing multiple action events to be queued. When processed, the application sends these events one after the other, causing the Timer's listeners to receive a sequence of events with no delay between them. Coalescing avoids this situation by reducing multiple pending events to a single event. Timers coalesce events by default.

**Parameters:**flag - specify false to turn off coalescing

### isCoalesce

public boolean **isCoalesce**()

Returns true if the Timer coalesces multiple pending action events.

**See Also:**[setCoalesce(boolean)](http://docs.google.com/javax/swing/Timer.html#setCoalesce(boolean))

### setActionCommand

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

Sets the string that will be delivered as the action command in ActionEvents fired by this timer. null is an acceptable value.

**Parameters:**command - the action command**Since:** 1.6

### getActionCommand

public [String](http://docs.google.com/java/lang/String.html) **getActionCommand**()

Returns the string that will be delivered as the action command in ActionEvents fired by this timer. May be null, which is also the default.

**Returns:**the action command used in firing events**Since:** 1.6

### start

public void **start**()

Starts the Timer, causing it to start sending action events to its listeners.

**See Also:**[stop()](http://docs.google.com/javax/swing/Timer.html#stop())

### isRunning

public boolean **isRunning**()

Returns true if the Timer is running.

**See Also:**[start()](http://docs.google.com/javax/swing/Timer.html#start())

### stop

public void **stop**()

Stops the Timer, causing it to stop sending action events to its listeners.

**See Also:**[start()](http://docs.google.com/javax/swing/Timer.html#start())

### restart

public void **restart**()

Restarts the Timer, canceling any pending firings and causing it to fire with its initial delay.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Timer.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/javax/swing/SwingWorker.StateValue.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/ToolTipManager.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/Timer.html)    [**NO FRAMES**](http://docs.google.com/Timer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

[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).