| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MidiChannel.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/sound/midi/MetaMessage.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/MidiDevice.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/MidiChannel.html)    [**NO FRAMES**](http://docs.google.com/MidiChannel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.sound.midi**

Interface MidiChannel

public interface **MidiChannel**

A MidiChannel object represents a single MIDI channel. Generally, each MidiChannel method processes a like-named MIDI "channel voice" or "channel mode" message as defined by the MIDI specification. However, MidiChannel adds some "get" methods that retrieve the value most recently set by one of the standard MIDI channel messages. Similarly, methods for per-channel solo and mute have been added.

A [Synthesizer](http://docs.google.com/javax/sound/midi/Synthesizer.html) object has a collection of MidiChannels, usually one for each of the 16 channels prescribed by the MIDI 1.0 specification. The Synthesizer generates sound when its MidiChannels receive noteOn messages.

See the MIDI 1.0 Specification for more information about the prescribed behavior of the MIDI channel messages, which are not exhaustively documented here. The specification is titled MIDI Reference: The Complete MIDI 1.0 Detailed Specification, and is published by the MIDI Manufacturer's Association (<http://www.midi.org>).

MIDI was originally a protocol for reporting the gestures of a keyboard musician. This genesis is visible in the MidiChannel API, which preserves such MIDI concepts as key number, key velocity, and key pressure. It should be understood that the MIDI data does not necessarily originate with a keyboard player (the source could be a different kind of musician, or software). Some devices might generate constant values for velocity and pressure, regardless of how the note was performed. Also, the MIDI specification often leaves it up to the synthesizer to use the data in the way the implementor sees fit. For example, velocity data need not always be mapped to volume and/or brightness.

**See Also:**[Synthesizer.getChannels()](http://docs.google.com/javax/sound/midi/Synthesizer.html#getChannels())

| **Method Summary** | |
| --- | --- |
| void | [**allNotesOff**](http://docs.google.com/javax/sound/midi/MidiChannel.html#allNotesOff())()            Turns off all notes that are currently sounding on this channel. |
| void | [**allSoundOff**](http://docs.google.com/javax/sound/midi/MidiChannel.html#allSoundOff())()            Immediately turns off all sounding notes on this channel, ignoring the state of the Hold Pedal and the internal decay rate of the current Instrument. |
| void | [**controlChange**](http://docs.google.com/javax/sound/midi/MidiChannel.html#controlChange(int,%20int))(int controller, int value)            Reacts to a change in the specified controller's value. |
| int | [**getChannelPressure**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getChannelPressure())()            Obtains the channel's keyboard pressure. |
| int | [**getController**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getController(int))(int controller)            Obtains the current value of the specified controller. |
| boolean | [**getMono**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getMono())()            Obtains the current mono/poly mode. |
| boolean | [**getMute**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getMute())()            Obtains the current mute state for this channel. |
| boolean | [**getOmni**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getOmni())()            Obtains the current omni mode. |
| int | [**getPitchBend**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getPitchBend())()            Obtains the upward or downward pitch offset for this channel. |
| int | [**getPolyPressure**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getPolyPressure(int))(int noteNumber)            Obtains the pressure with which the specified key is being depressed. |
| int | [**getProgram**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getProgram())()            Obtains the current program number for this channel. |
| boolean | [**getSolo**](http://docs.google.com/javax/sound/midi/MidiChannel.html#getSolo())()            Obtains the current solo state for this channel. |
| boolean | [**localControl**](http://docs.google.com/javax/sound/midi/MidiChannel.html#localControl(boolean))(boolean on)            Turns local control on or off. |
| void | [**noteOff**](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int))(int noteNumber)            Turns the specified note off. |
| void | [**noteOff**](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int,%20int))(int noteNumber, int velocity)            Turns the specified note off. |
| void | [**noteOn**](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOn(int,%20int))(int noteNumber, int velocity)            Starts the specified note sounding. |
| void | [**programChange**](http://docs.google.com/javax/sound/midi/MidiChannel.html#programChange(int))(int program)            Changes a program (patch). |
| void | [**programChange**](http://docs.google.com/javax/sound/midi/MidiChannel.html#programChange(int,%20int))(int bank, int program)            Changes the program using bank and program (patch) numbers. |
| void | [**resetAllControllers**](http://docs.google.com/javax/sound/midi/MidiChannel.html#resetAllControllers())()            Resets all the implemented controllers to their default values. |
| void | [**setChannelPressure**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setChannelPressure(int))(int pressure)            Reacts to a change in the keyboard pressure. |
| void | [**setMono**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setMono(boolean))(boolean on)            Turns mono mode on or off. |
| void | [**setMute**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setMute(boolean))(boolean mute)            Sets the mute state for this channel. |
| void | [**setOmni**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setOmni(boolean))(boolean on)            Turns omni mode on or off. |
| void | [**setPitchBend**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPitchBend(int))(int bend)            Changes the pitch offset for all notes on this channel. |
| void | [**setPolyPressure**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPolyPressure(int,%20int))(int noteNumber, int pressure)            Reacts to a change in the specified note's key pressure. |
| void | [**setSolo**](http://docs.google.com/javax/sound/midi/MidiChannel.html#setSolo(boolean))(boolean soloState)            Sets the solo state for this channel. |

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

### noteOn

void **noteOn**(int noteNumber,  
 int velocity)

Starts the specified note sounding. The key-down velocity usually controls the note's volume and/or brightness. If velocity is zero, this method instead acts like [noteOff(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int)), terminating the note.

**Parameters:**noteNumber - the MIDI note number, from 0 to 127 (60 = Middle C)velocity - the speed with which the key was depressed**See Also:**[noteOff(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int,%20int))

### noteOff

void **noteOff**(int noteNumber,  
 int velocity)

Turns the specified note off. The key-up velocity, if not ignored, can be used to affect how quickly the note decays. In any case, the note might not die away instantaneously; its decay rate is determined by the internals of the Instrument. If the Hold Pedal (a controller; see [controlChange](http://docs.google.com/javax/sound/midi/MidiChannel.html#controlChange(int,%20int))) is down, the effect of this method is deferred until the pedal is released.

**Parameters:**noteNumber - the MIDI note number, from 0 to 127 (60 = Middle C)velocity - the speed with which the key was released**See Also:**[noteOff(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int)), [noteOn(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOn(int,%20int)), [allNotesOff()](http://docs.google.com/javax/sound/midi/MidiChannel.html#allNotesOff()), [allSoundOff()](http://docs.google.com/javax/sound/midi/MidiChannel.html#allSoundOff())

### noteOff

void **noteOff**(int noteNumber)

Turns the specified note off.

**Parameters:**noteNumber - the MIDI note number, from 0 to 127 (60 = Middle C)**See Also:**[noteOff(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int,%20int))

### setPolyPressure

void **setPolyPressure**(int noteNumber,  
 int pressure)

Reacts to a change in the specified note's key pressure. Polyphonic key pressure allows a keyboard player to press multiple keys simultaneously, each with a different amount of pressure. The pressure, if not ignored, is typically used to vary such features as the volume, brightness, or vibrato of the note. It is possible that the underlying synthesizer does not support this MIDI message. In order to verify that setPolyPressure was successful, use getPolyPressure.

**Parameters:**noteNumber - the MIDI note number, from 0 to 127 (60 = Middle C)pressure - value for the specified key, from 0 to 127 (127 = maximum pressure)**See Also:**[getPolyPressure(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#getPolyPressure(int))

### getPolyPressure

int **getPolyPressure**(int noteNumber)

Obtains the pressure with which the specified key is being depressed.

**Parameters:**noteNumber - the MIDI note number, from 0 to 127 (60 = Middle C) If the device does not support setting poly pressure, this method always returns 0. Calling setPolyPressure will have no effect then. **Returns:**the amount of pressure for that note, from 0 to 127 (127 = maximum pressure)**See Also:**[setPolyPressure(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPolyPressure(int,%20int))

### setChannelPressure

void **setChannelPressure**(int pressure)

Reacts to a change in the keyboard pressure. Channel pressure indicates how hard the keyboard player is depressing the entire keyboard. This can be the maximum or average of the per-key pressure-sensor values, as set by setPolyPressure. More commonly, it is a measurement of a single sensor on a device that doesn't implement polyphonic key pressure. Pressure can be used to control various aspects of the sound, as described under [setPolyPressure](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPolyPressure(int,%20int)). It is possible that the underlying synthesizer does not support this MIDI message. In order to verify that setChannelPressure was successful, use getChannelPressure.

**Parameters:**pressure - the pressure with which the keyboard is being depressed, from 0 to 127 (127 = maximum pressure)**See Also:**[setPolyPressure(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPolyPressure(int,%20int)), [getChannelPressure()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getChannelPressure())

### getChannelPressure

int **getChannelPressure**()

Obtains the channel's keyboard pressure. If the device does not support setting channel pressure, this method always returns 0. Calling setChannelPressure will have no effect then.

**Returns:**the amount of pressure for that note, from 0 to 127 (127 = maximum pressure)**See Also:**[setChannelPressure(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setChannelPressure(int))

### controlChange

void **controlChange**(int controller,  
 int value)

Reacts to a change in the specified controller's value. A controller is some control other than a keyboard key, such as a switch, slider, pedal, wheel, or breath-pressure sensor. The MIDI 1.0 Specification provides standard numbers for typical controllers on MIDI devices, and describes the intended effect for some of the controllers. The way in which an Instrument reacts to a controller change may be specific to the Instrument.

The MIDI 1.0 Specification defines both 7-bit controllers and 14-bit controllers. Continuous controllers, such as wheels and sliders, typically have 14 bits (two MIDI bytes), while discrete controllers, such as switches, typically have 7 bits (one MIDI byte). Refer to the specification to see the expected resolution for each type of control.

Controllers 64 through 95 (0x40 - 0x5F) allow 7-bit precision. The value of a 7-bit controller is set completely by the value argument. An additional set of controllers provide 14-bit precision by using two controller numbers, one for the most significant 7 bits and another for the least significant 7 bits. Controller numbers 0 through 31 (0x00 - 0x1F) control the most significant 7 bits of 14-bit controllers; controller numbers 32 through 63 (0x20 - 0x3F) control the least significant 7 bits of these controllers. For example, controller number 7 (0x07) controls the upper 7 bits of the channel volume controller, and controller number 39 (0x27) controls the lower 7 bits. The value of a 14-bit controller is determined by the interaction of the two halves. When the most significant 7 bits of a controller are set (using controller numbers 0 through 31), the lower 7 bits are automatically set to 0. The corresponding controller number for the lower 7 bits may then be used to further modulate the controller value. It is possible that the underlying synthesizer does not support a specific controller message. In order to verify that a call to controlChange was successful, use getController.

**Parameters:**controller - the controller number (0 to 127; see the MIDI 1.0 Specification for the interpretation)value - the value to which the specified controller is changed (0 to 127)**See Also:**[getController(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#getController(int))

### getController

int **getController**(int controller)

Obtains the current value of the specified controller. The return value is represented with 7 bits. For 14-bit controllers, the MSB and LSB controller value needs to be obtained separately. For example, the 14-bit value of the volume controller can be calculated by multiplying the value of controller 7 (0x07, channel volume MSB) with 128 and adding the value of controller 39 (0x27, channel volume LSB). If the device does not support setting a specific controller, this method returns 0 for that controller. Calling controlChange will have no effect then.

**Parameters:**controller - the number of the controller whose value is desired. The allowed range is 0-127; see the MIDI 1.0 Specification for the interpretation. **Returns:**the current value of the specified controller (0 to 127)**See Also:**[controlChange(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#controlChange(int,%20int))

### programChange

void **programChange**(int program)

Changes a program (patch). This selects a specific instrument from the currently selected bank of instruments.

The MIDI specification does not dictate whether notes that are already sounding should switch to the new instrument (timbre) or continue with their original timbre until terminated by a note-off.

The program number is zero-based (expressed from 0 to 127). Note that MIDI hardware displays and literature about MIDI typically use the range 1 to 128 instead. It is possible that the underlying synthesizer does not support a specific program. In order to verify that a call to programChange was successful, use getProgram.

**Parameters:**program - the program number to switch to (0 to 127)**See Also:**[programChange(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#programChange(int,%20int)), [getProgram()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getProgram())

### programChange

void **programChange**(int bank,  
 int program)

Changes the program using bank and program (patch) numbers. It is possible that the underlying synthesizer does not support a specific bank, or program. In order to verify that a call to programChange was successful, use getProgram and getController. Since banks are changed by way of control changes, you can verify the current bank with the following statement:

int bank = (getController(0) \* 128)  
 + getController(32);

**Parameters:**bank - the bank number to switch to (0 to 16383)program - the program (patch) to use in the specified bank (0 to 127)**See Also:**[programChange(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#programChange(int)), [getProgram()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getProgram())

### getProgram

int **getProgram**()

Obtains the current program number for this channel.

**Returns:**the program number of the currently selected patch**See Also:**[Patch.getProgram()](http://docs.google.com/javax/sound/midi/Patch.html#getProgram()), [Synthesizer.loadInstrument(javax.sound.midi.Instrument)](http://docs.google.com/javax/sound/midi/Synthesizer.html#loadInstrument(javax.sound.midi.Instrument)), [programChange(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#programChange(int))

### setPitchBend

void **setPitchBend**(int bend)

Changes the pitch offset for all notes on this channel. This affects all currently sounding notes as well as subsequent ones. (For pitch bend to cease, the value needs to be reset to the center position.)

The MIDI specification stipulates that pitch bend be a 14-bit value, where zero is maximum downward bend, 16383 is maximum upward bend, and 8192 is the center (no pitch bend). The actual amount of pitch change is not specified; it can be changed by a pitch-bend sensitivity setting. However, the General MIDI specification says that the default range should be two semitones up and down from center. It is possible that the underlying synthesizer does not support this MIDI message. In order to verify that setPitchBend was successful, use getPitchBend.

**Parameters:**bend - the amount of pitch change, as a nonnegative 14-bit value (8192 = no bend)**See Also:**[getPitchBend()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getPitchBend())

### getPitchBend

int **getPitchBend**()

Obtains the upward or downward pitch offset for this channel. If the device does not support setting pitch bend, this method always returns 8192. Calling setPitchBend will have no effect then.

**Returns:**bend amount, as a nonnegative 14-bit value (8192 = no bend)**See Also:**[setPitchBend(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setPitchBend(int))

### resetAllControllers

void **resetAllControllers**()

Resets all the implemented controllers to their default values.

**See Also:**[controlChange(int, int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#controlChange(int,%20int))

### allNotesOff

void **allNotesOff**()

Turns off all notes that are currently sounding on this channel. The notes might not die away instantaneously; their decay rate is determined by the internals of the Instrument. If the Hold Pedal controller (see [controlChange](http://docs.google.com/javax/sound/midi/MidiChannel.html#controlChange(int,%20int))) is down, the effect of this method is deferred until the pedal is released.

**See Also:**[allSoundOff()](http://docs.google.com/javax/sound/midi/MidiChannel.html#allSoundOff()), [noteOff(int)](http://docs.google.com/javax/sound/midi/MidiChannel.html#noteOff(int))

### allSoundOff

void **allSoundOff**()

Immediately turns off all sounding notes on this channel, ignoring the state of the Hold Pedal and the internal decay rate of the current Instrument.

**See Also:**[allNotesOff()](http://docs.google.com/javax/sound/midi/MidiChannel.html#allNotesOff())

### localControl

boolean **localControl**(boolean on)

Turns local control on or off. The default is for local control to be on. The "on" setting means that if a device is capable of both synthesizing sound and transmitting MIDI messages, it will synthesize sound in response to the note-on and note-off messages that it itself transmits. It will also respond to messages received from other transmitting devices. The "off" setting means that the synthesizer will ignore its own transmitted MIDI messages, but not those received from other devices. It is possible that the underlying synthesizer does not support local control. In order to verify that a call to localControl was successful, check the return value.

**Parameters:**on - true to turn local control on, false to turn local control off **Returns:**the new local-control value, or false if local control is not supported

### setMono

void **setMono**(boolean on)

Turns mono mode on or off. In mono mode, the channel synthesizes only one note at a time. In poly mode (identical to mono mode off), the channel can synthesize multiple notes simultaneously. The default is mono off (poly mode on).

"Mono" is short for the word "monophonic," which in this context is opposed to the word "polyphonic" and refers to a single synthesizer voice per MIDI channel. It has nothing to do with how many audio channels there might be (as in "monophonic" versus "stereophonic" recordings). It is possible that the underlying synthesizer does not support mono mode. In order to verify that a call to setMono was successful, use getMono.

**Parameters:**on - true to turn mono mode on, false to turn it off (which means turning poly mode on).**See Also:**[getMono()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getMono()), [VoiceStatus](http://docs.google.com/javax/sound/midi/VoiceStatus.html)

### getMono

boolean **getMono**()

Obtains the current mono/poly mode. Synthesizers that do not allow changing mono/poly mode will always return the same value, regardless of calls to setMono.

**Returns:**true if mono mode is on, otherwise false (meaning poly mode is on).**See Also:**[setMono(boolean)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setMono(boolean))

### setOmni

void **setOmni**(boolean on)

Turns omni mode on or off. In omni mode, the channel responds to messages sent on all channels. When omni is off, the channel responds only to messages sent on its channel number. The default is omni off. It is possible that the underlying synthesizer does not support omni mode. In order to verify that setOmni was successful, use getOmni.

**Parameters:**on - true to turn omni mode on, false to turn it off.**See Also:**[getOmni()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getOmni()), [VoiceStatus](http://docs.google.com/javax/sound/midi/VoiceStatus.html)

### getOmni

boolean **getOmni**()

Obtains the current omni mode. Synthesizers that do not allow changing the omni mode will always return the same value, regardless of calls to setOmni.

**Returns:**true if omni mode is on, otherwise false (meaning omni mode is off).**See Also:**[setOmni(boolean)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setOmni(boolean))

### setMute

void **setMute**(boolean mute)

Sets the mute state for this channel. A value of true means the channel is to be muted, false means the channel can sound (if other channels are not soloed).

Unlike [allSoundOff()](http://docs.google.com/javax/sound/midi/MidiChannel.html#allSoundOff()), this method applies to only a specific channel, not to all channels. Further, it silences not only currently sounding notes, but also subsequently received notes. It is possible that the underlying synthesizer does not support muting channels. In order to verify that a call to setMute was successful, use getMute.

**Parameters:**mute - the new mute state**See Also:**[getMute()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getMute()), [setSolo(boolean)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setSolo(boolean))

### getMute

boolean **getMute**()

Obtains the current mute state for this channel. If the underlying synthesizer does not support muting this channel, this method always returns false.

**Returns:**true the channel is muted, or false if not**See Also:**[setMute(boolean)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setMute(boolean))

### setSolo

void **setSolo**(boolean soloState)

Sets the solo state for this channel. If solo is true only this channel and other soloed channels will sound. If solo is false then only other soloed channels will sound, unless no channels are soloed, in which case all unmuted channels will sound. It is possible that the underlying synthesizer does not support solo channels. In order to verify that a call to setSolo was successful, use getSolo.

**Parameters:**soloState - new solo state for the channel**See Also:**[getSolo()](http://docs.google.com/javax/sound/midi/MidiChannel.html#getSolo())

### getSolo

boolean **getSolo**()

Obtains the current solo state for this channel. If the underlying synthesizer does not support solo on this channel, this method always returns false.

**Returns:**true the channel is solo, or false if not**See Also:**[setSolo(boolean)](http://docs.google.com/javax/sound/midi/MidiChannel.html#setSolo(boolean))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MidiChannel.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/sound/midi/MetaMessage.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/midi/MidiDevice.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/midi/MidiChannel.html)    [**NO FRAMES**](http://docs.google.com/MidiChannel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[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).