| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Clip.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/sampled/BooleanControl.Type.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/sampled/CompoundControl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/sampled/Clip.html)    [**NO FRAMES**](http://docs.google.com/Clip.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | CONSTR | [METHOD](#3rdcrjn) |

## **javax.sound.sampled**

Interface Clip

**All Superinterfaces:** [DataLine](http://docs.google.com/javax/sound/sampled/DataLine.html), [Line](http://docs.google.com/javax/sound/sampled/Line.html)

public interface **Clip**extends [DataLine](http://docs.google.com/javax/sound/sampled/DataLine.html)

The Clip interface represents a special kind of data line whose audio data can be loaded prior to playback, instead of being streamed in real time.

Because the data is pre-loaded and has a known length, you can set a clip to start playing at any position in its audio data. You can also create a loop, so that when the clip is played it will cycle repeatedly. Loops are specified with a starting and ending sample frame, along with the number of times that the loop should be played.

Clips may be obtained from a [Mixer](http://docs.google.com/javax/sound/sampled/Mixer.html) that supports lines of this type. Data is loaded into a clip when it is opened.

Playback of an audio clip may be started and stopped using the start and stop methods. These methods do not reset the media position; start causes playback to continue from the position where playback was last stopped. To restart playback from the beginning of the clip's audio data, simply follow the invocation of [stop](http://docs.google.com/javax/sound/sampled/DataLine.html#stop()) with setFramePosition(0), which rewinds the media to the beginning of the clip.

**Since:** 1.3

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

| **Nested classes/interfaces inherited from interface javax.sound.sampled.**[**DataLine**](http://docs.google.com/javax/sound/sampled/DataLine.html) |
| --- |
| [DataLine.Info](http://docs.google.com/javax/sound/sampled/DataLine.Info.html) |

| **Field Summary** | |
| --- | --- |
| static int | [**LOOP\_CONTINUOUSLY**](http://docs.google.com/javax/sound/sampled/Clip.html#LOOP_CONTINUOUSLY)            A value indicating that looping should continue indefinitely rather than complete after a specific number of loops. |

| **Method Summary** | |
| --- | --- |
| int | [**getFrameLength**](http://docs.google.com/javax/sound/sampled/Clip.html#getFrameLength())()            Obtains the media length in sample frames. |
| long | [**getMicrosecondLength**](http://docs.google.com/javax/sound/sampled/Clip.html#getMicrosecondLength())()            Obtains the media duration in microseconds |
| void | [**loop**](http://docs.google.com/javax/sound/sampled/Clip.html#loop(int))(int count)            Starts looping playback from the current position. |
| void | [**open**](http://docs.google.com/javax/sound/sampled/Clip.html#open(javax.sound.sampled.AudioFormat,%20byte%5B%5D,%20int,%20int))([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format, byte[] data, int offset, int bufferSize)            Opens the clip, meaning that it should acquire any required system resources and become operational. |
| void | [**open**](http://docs.google.com/javax/sound/sampled/Clip.html#open(javax.sound.sampled.AudioInputStream))([AudioInputStream](http://docs.google.com/javax/sound/sampled/AudioInputStream.html) stream)            Opens the clip with the format and audio data present in the provided audio input stream. |
| void | [**setFramePosition**](http://docs.google.com/javax/sound/sampled/Clip.html#setFramePosition(int))(int frames)            Sets the media position in sample frames. |
| void | [**setLoopPoints**](http://docs.google.com/javax/sound/sampled/Clip.html#setLoopPoints(int,%20int))(int start, int end)            Sets the first and last sample frames that will be played in the loop. |
| void | [**setMicrosecondPosition**](http://docs.google.com/javax/sound/sampled/Clip.html#setMicrosecondPosition(long))(long microseconds)            Sets the media position in microseconds. |

| **Methods inherited from interface javax.sound.sampled.**[**DataLine**](http://docs.google.com/javax/sound/sampled/DataLine.html) |
| --- |
| [available](http://docs.google.com/javax/sound/sampled/DataLine.html#available()), [drain](http://docs.google.com/javax/sound/sampled/DataLine.html#drain()), [flush](http://docs.google.com/javax/sound/sampled/DataLine.html#flush()), [getBufferSize](http://docs.google.com/javax/sound/sampled/DataLine.html#getBufferSize()), [getFormat](http://docs.google.com/javax/sound/sampled/DataLine.html#getFormat()), [getFramePosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getFramePosition()), [getLevel](http://docs.google.com/javax/sound/sampled/DataLine.html#getLevel()), [getLongFramePosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getLongFramePosition()), [getMicrosecondPosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getMicrosecondPosition()), [isActive](http://docs.google.com/javax/sound/sampled/DataLine.html#isActive()), [isRunning](http://docs.google.com/javax/sound/sampled/DataLine.html#isRunning()), [start](http://docs.google.com/javax/sound/sampled/DataLine.html#start()), [stop](http://docs.google.com/javax/sound/sampled/DataLine.html#stop()) |

| **Methods inherited from interface javax.sound.sampled.**[**Line**](http://docs.google.com/javax/sound/sampled/Line.html) |
| --- |
| [addLineListener](http://docs.google.com/javax/sound/sampled/Line.html#addLineListener(javax.sound.sampled.LineListener)), [close](http://docs.google.com/javax/sound/sampled/Line.html#close()), [getControl](http://docs.google.com/javax/sound/sampled/Line.html#getControl(javax.sound.sampled.Control.Type)), [getControls](http://docs.google.com/javax/sound/sampled/Line.html#getControls()), [getLineInfo](http://docs.google.com/javax/sound/sampled/Line.html#getLineInfo()), [isControlSupported](http://docs.google.com/javax/sound/sampled/Line.html#isControlSupported(javax.sound.sampled.Control.Type)), [isOpen](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [open](http://docs.google.com/javax/sound/sampled/Line.html#open()), [removeLineListener](http://docs.google.com/javax/sound/sampled/Line.html#removeLineListener(javax.sound.sampled.LineListener)) |

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

### LOOP\_CONTINUOUSLY

static final int **LOOP\_CONTINUOUSLY**

A value indicating that looping should continue indefinitely rather than complete after a specific number of loops.

**See Also:**[loop(int)](http://docs.google.com/javax/sound/sampled/Clip.html#loop(int)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.sound.sampled.Clip.LOOP_CONTINUOUSLY)

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

### open

void **open**([AudioFormat](http://docs.google.com/javax/sound/sampled/AudioFormat.html) format,  
 byte[] data,  
 int offset,  
 int bufferSize)  
 throws [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html)

Opens the clip, meaning that it should acquire any required system resources and become operational. The clip is opened with the format and audio data indicated. If this operation succeeds, the line is marked as open and an [OPEN](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#OPEN) event is dispatched to the line's listeners.

Invoking this method on a line which is already open is illegal and may result in an IllegalStateException.

Note that some lines, once closed, cannot be reopened. Attempts to reopen such a line will always result in a [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html).

**Parameters:**format - the format of the supplied audio datadata - a byte array containing audio data to load into the clipoffset - the point at which to start copying, expressed in *bytes* from the beginning of the arraybufferSize - the number of *bytes* of data to load into the clip from the array. **Throws:** [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html) - if the line cannot be opened due to resource restrictions [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the buffer size does not represent an integral number of sample frames, or if format is not fully specified or invalid [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the line is already open [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the line cannot be opened due to security restrictions**See Also:**[Line.close()](http://docs.google.com/javax/sound/sampled/Line.html#close()), [Line.isOpen()](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [LineListener](http://docs.google.com/javax/sound/sampled/LineListener.html)

### open

void **open**([AudioInputStream](http://docs.google.com/javax/sound/sampled/AudioInputStream.html) stream)  
 throws [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html),  
 [IOException](http://docs.google.com/java/io/IOException.html)

Opens the clip with the format and audio data present in the provided audio input stream. Opening a clip means that it should acquire any required system resources and become operational. If this operation input stream. If this operation succeeds, the line is marked open and an [OPEN](http://docs.google.com/javax/sound/sampled/LineEvent.Type.html#OPEN) event is dispatched to the line's listeners.

Invoking this method on a line which is already open is illegal and may result in an IllegalStateException.

Note that some lines, once closed, cannot be reopened. Attempts to reopen such a line will always result in a [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html).

**Parameters:**stream - an audio input stream from which audio data will be read into the clip **Throws:** [LineUnavailableException](http://docs.google.com/javax/sound/sampled/LineUnavailableException.html) - if the line cannot be opened due to resource restrictions [IOException](http://docs.google.com/java/io/IOException.html) - if an I/O exception occurs during reading of the stream [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the stream's audio format is not fully specified or invalid [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if the line is already open [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if the line cannot be opened due to security restrictions**See Also:**[Line.close()](http://docs.google.com/javax/sound/sampled/Line.html#close()), [Line.isOpen()](http://docs.google.com/javax/sound/sampled/Line.html#isOpen()), [LineListener](http://docs.google.com/javax/sound/sampled/LineListener.html)

### getFrameLength

int **getFrameLength**()

Obtains the media length in sample frames.

**Returns:**the media length, expressed in sample frames, or AudioSystem.NOT\_SPECIFIED if the line is not open.**See Also:**[AudioSystem.NOT\_SPECIFIED](http://docs.google.com/javax/sound/sampled/AudioSystem.html#NOT_SPECIFIED)

### getMicrosecondLength

long **getMicrosecondLength**()

Obtains the media duration in microseconds

**Returns:**the media duration, expressed in microseconds, or AudioSystem.NOT\_SPECIFIED if the line is not open.**See Also:**[AudioSystem.NOT\_SPECIFIED](http://docs.google.com/javax/sound/sampled/AudioSystem.html#NOT_SPECIFIED)

### setFramePosition

void **setFramePosition**(int frames)

Sets the media position in sample frames. The position is zero-based; the first frame is frame number zero. When the clip begins playing the next time, it will start by playing the frame at this position.

To obtain the current position in sample frames, use the [getFramePosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getFramePosition()) method of DataLine.

**Parameters:**frames - the desired new media position, expressed in sample frames

### setMicrosecondPosition

void **setMicrosecondPosition**(long microseconds)

Sets the media position in microseconds. When the clip begins playing the next time, it will start at this position. The level of precision is not guaranteed. For example, an implementation might calculate the microsecond position from the current frame position and the audio sample frame rate. The precision in microseconds would then be limited to the number of microseconds per sample frame.

To obtain the current position in microseconds, use the [getMicrosecondPosition](http://docs.google.com/javax/sound/sampled/DataLine.html#getMicrosecondPosition()) method of DataLine.

**Parameters:**microseconds - the desired new media position, expressed in microseconds

### setLoopPoints

void **setLoopPoints**(int start,  
 int end)

Sets the first and last sample frames that will be played in the loop. The ending point must be greater than or equal to the starting point, and both must fall within the the size of the loaded media. A value of 0 for the starting point means the beginning of the loaded media. Similarly, a value of -1 for the ending point indicates the last frame of the media.

**Parameters:**start - the loop's starting position, in sample frames (zero-based)end - the loop's ending position, in sample frames (zero-based), or -1 to indicate the final frame **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the requested loop points cannot be set, usually because one or both falls outside the media's duration or because the ending point is before the starting point

### loop

void **loop**(int count)

Starts looping playback from the current position. Playback will continue to the loop's end point, then loop back to the loop start point count times, and finally continue playback to the end of the clip.

If the current position when this method is invoked is greater than the loop end point, playback simply continues to the end of the clip without looping.

A count value of 0 indicates that any current looping should cease and playback should continue to the end of the clip. The behavior is undefined when this method is invoked with any other value during a loop operation.

If playback is stopped during looping, the current loop status is cleared; the behavior of subsequent loop and start requests is not affected by an interrupted loop operation.

**Parameters:**count - the number of times playback should loop back from the loop's end position to the loop's start position, or [LOOP\_CONTINUOUSLY](http://docs.google.com/javax/sound/sampled/Clip.html#LOOP_CONTINUOUSLY) to indicate that looping should continue until interrupted

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Clip.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/sampled/BooleanControl.Type.html)   [**NEXT CLASS**](http://docs.google.com/javax/sound/sampled/CompoundControl.html) | [**FRAMES**](http://docs.google.com/index.html?javax/sound/sampled/Clip.html)    [**NO FRAMES**](http://docs.google.com/Clip.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#tyjcwt) | CONSTR | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | CONSTR | [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).