| | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-all.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | | |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/SimplePicture.html)   [**NEXT CLASS**](http://docs.google.com/SimpleTurtle.html) | [**FRAMES**](http://docs.google.com/index.html?SimpleSound.html)    [**NO FRAMES**](http://docs.google.com/SimpleSound.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#2jxsxqh) |

## Class SimpleSound

[java.lang.Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)  
 **SimpleSound**

**Direct Known Subclasses:** [Sound](http://docs.google.com/Sound.html)

public class **SimpleSound**extends [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true)

The SimpleSound class is an implementation of the Java Sound API specifically designed for use with students. http://java.sun.com/products/java-media/sound/index.html

This class allows for easy playback, and manipulation of AU, AIFF, and WAV files.

Code & ideas for this class related to playing and viewing the sound were borrowed from the Java Sound Demo: http://java.sun.com/products/java-media/sound/ samples/JavaSoundDemo/ Also, some code borrowed from Tritonus as noted. Copyright Georgia Institute of Technology 2004

**Author:** Ellie Harmon, ellie@cc.gatech.edu, Barbara Ericson ericson@mindspring.com

| **Field Summary** | |
| --- | --- |
| static int | [**MAX\_NEG**](http://docs.google.com/SimpleSound.html#MAX_NEG)            Constant for max negative value |
| static int | [**MAX\_POS**](http://docs.google.com/SimpleSound.html#MAX_POS)            Constant for max positive value |

| **Constructor Summary** | |
| --- | --- |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound())()            Constructs a SimpleSound of 3 seconds long. |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound(int))(int numFrames)            Constructs a SimpleSound of the specified length. |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound(int,%20boolean))(int sampleSizeInBits, boolean isBigEndian)            Constructs a simple sound with the given sample size in bits and type of endian (big or little) |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound(int,%20int))(int numFrames, int sampleRate)            Constructs a SimpleSound of the specified length. |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound(SimpleSound))([SimpleSound](http://docs.google.com/SimpleSound.html) sound)            Constructor that creates a new SimpleSound by copying a passed SimpleSound |
| [**SimpleSound**](http://docs.google.com/SimpleSound.html#SimpleSound(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) fileName)            Constructs a new SimpleSound from the given file. |

| **Method Summary** | |
| --- | --- |
| byte[] | [**asArray**](http://docs.google.com/SimpleSound.html#asArray())()            Method to return the byte array |
| void | [**blockingPlay**](http://docs.google.com/SimpleSound.html#blockingPlay())()            Plays the sound, then sleeps for how long the sound SHOULD last. |
| void | [**blockingPlayAtRateDur**](http://docs.google.com/SimpleSound.html#blockingPlayAtRateDur(double,%20double))(double rate, double durInFrames)            First, checks the value of durInFrames to make sure that it is not larger than Integer.MAX\_VALUE to guarrantee safe casting. |
| void | [**blockingPlayAtRateInRange**](http://docs.google.com/SimpleSound.html#blockingPlayAtRateInRange(float,%20int,%20int))(float rate, int startFrame, int endFrame)            Calls playAtRateInRange(rate, startFrame, endFrame, true) . |
| void | [**blockingPlayOld**](http://docs.google.com/SimpleSound.html#blockingPlayOld())()            Creates a new Playback thread, starts it, then waits for the entire sound to finish playing before it returns. |
| static void | [**convert**](http://docs.google.com/SimpleSound.html#convert(java.lang.String,%20java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) mp3File, [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) wavFile)            Method to convert a mp3 sound into a wav sound |
| void | [**explore**](http://docs.google.com/SimpleSound.html#explore())()            Method to open a sound viewer on a copy of this sound |
| [AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true) | [**getAudioFileFormat**](http://docs.google.com/SimpleSound.html#getAudioFileFormat())()            Method that returns the AudioFileFormat describing this simple sound. |
| byte[] | [**getBuffer**](http://docs.google.com/SimpleSound.html#getBuffer())()            Method that returns the byte array representation of this simple sound. |
| int | [**getChannels**](http://docs.google.com/SimpleSound.html#getChannels())()            Obtains the number of channels of this sound. |
| boolean | [**getDEBUG**](http://docs.google.com/SimpleSound.html#getDEBUG())()            Method to get the value of the debug flag |
| [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) | [**getFileName**](http://docs.google.com/SimpleSound.html#getFileName())()            Method that returns the name of the file this sound came from. |
| byte[] | [**getFrame**](http://docs.google.com/SimpleSound.html#getFrame(int))(int frameNum)            Returns an array containing all of the bytes in the specified frame. |
| int | [**getLeftSample**](http://docs.google.com/SimpleSound.html#getLeftSample(int))(int frameNum)            Obtains the left sample of the audio data contained at the specified frame. |
| int | [**getLength**](http://docs.google.com/SimpleSound.html#getLength())()            Method to return the length of the sound as the number of samples |
| int | [**getLengthInBytes**](http://docs.google.com/SimpleSound.html#getLengthInBytes())()            Obtains the length of this sound in bytes. |
| int | [**getLengthInFrames**](http://docs.google.com/SimpleSound.html#getLengthInFrames())()            Obtains the length of the audio data contained in the file, expressed in sample frames. |
| int | [**getNumSamples**](http://docs.google.com/SimpleSound.html#getNumSamples())()            Returns the number of samples in this sound |
| [Vector](http://java.sun.com/javase/6/docs/api/java/util/Vector.html?is-external=true) | [**getPlaybacks**](http://docs.google.com/SimpleSound.html#getPlaybacks())()            Method that returns the vector of playback threads currently active on this sound. |
| int | [**getRightSample**](http://docs.google.com/SimpleSound.html#getRightSample(int))(int frameNum)            Obtains the right sample of the audio data contained at the specified frame. |
| [SoundSample](http://docs.google.com/SoundSample.html) | [**getSample**](http://docs.google.com/SimpleSound.html#getSample(int))(int frameNum)            Method to create and return a SoundSample object for the given frame number |
| [SoundSample](http://docs.google.com/SoundSample.html)[] | [**getSamples**](http://docs.google.com/SimpleSound.html#getSamples())()            Method to create and return an array of SoundSample objects |
| int | [**getSampleValue**](http://docs.google.com/SimpleSound.html#getSampleValue(int))(int frameNum)            If this is a mono sound, obtains the single sample contained within this frame, else obtains the first (left) sample contained in the specified frame. |
| int | [**getSampleValueAt**](http://docs.google.com/SimpleSound.html#getSampleValueAt(int))(int index)            Method to get the sample at the passed index and handle any SoundExceptions |
| double | [**getSamplingRate**](http://docs.google.com/SimpleSound.html#getSamplingRate())()            Method to get the sampling rate of this sound |
| [SoundExplorer](http://docs.google.com/SoundExplorer.html) | [**getSoundExplorer**](http://docs.google.com/SimpleSound.html#getSoundExplorer())()            Method that returns the SoundExplorer |
| boolean | [**isStereo**](http://docs.google.com/SimpleSound.html#isStereo())()            Method to check if a sound is stereo (2 channels) or not |
| void | [**loadFromFile**](http://docs.google.com/SimpleSound.html#loadFromFile(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) inFileName)            Resets the fields of this sound so that it now represents the sound in the specified file. |
| [AudioInputStream](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioInputStream.html?is-external=true) | [**makeAIS**](http://docs.google.com/SimpleSound.html#makeAIS())()            Creates an AudioInputStream for this sound from the buffer and the audioFileFormat. |
| void | [**play**](http://docs.google.com/SimpleSound.html#play())()            Creates a new Playback thread and starts it. |
| void | [**playAtRateDur**](http://docs.google.com/SimpleSound.html#playAtRateDur(double,%20double))(double rate, double durInFrames)            Checks the value of durInFrames to make sure that it is not larger than Integer.MAX\_VALUE to guarrantee safe casting. |
| void | [**playAtRateInRange**](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int))(float rate, int startFrame, int endFrame)            Calls playAtRateInRange(rate, startFrame, endFrame, false) . |
| void | [**playAtRateInRange**](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int,%20boolean))(float rate, int startFrame, int endFrame, boolean isBlocking)            Plays the specified segment of this sound at the given sample rate. |
| static void | [**playNote**](http://docs.google.com/SimpleSound.html#playNote(int,%20int,%20int))(int key, int duration, int intensity)            Method to play a note using MIDI |
| void | [**printError**](http://docs.google.com/SimpleSound.html#printError(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message)            Invokes printError(message, null) |
| void | [**printError**](http://docs.google.com/SimpleSound.html#printError(java.lang.String,%20java.lang.Exception))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message, [Exception](http://java.sun.com/javase/6/docs/api/java/lang/Exception.html?is-external=true) e)            Prints the given String to the "standard" error output stream, then prints a stack trace on the exception, and then exits the program. |
| void | [**removePlayback**](http://docs.google.com/SimpleSound.html#removePlayback(Playback))([Playback](http://docs.google.com/Playback.html) playbackToRemove)            Deletes the specified playback object from the Vector. |
| void | [**setAudioFileFormat**](http://docs.google.com/SimpleSound.html#setAudioFileFormat(javax.sound.sampled.AudioFileFormat))([AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true) newAudioFileFormat)            Changes the AudioFileFormat of this sound. |
| void | [**setBuffer**](http://docs.google.com/SimpleSound.html#setBuffer(byte%5B%5D))(byte[] newBuffer)            Changes the byte array that represents this sound. |
| void | [**setFrame**](http://docs.google.com/SimpleSound.html#setFrame(int,%20byte%5B%5D))(int frameNum, byte[] theFrame)            Changes the value of each byte of the specified frame. |
| void | [**setLeftSample**](http://docs.google.com/SimpleSound.html#setLeftSample(int,%20int))(int frameNum, int sample) |
| void | [**setRightSample**](http://docs.google.com/SimpleSound.html#setRightSample(int,%20int))(int frameNum, int sample) |
| void | [**setSampleValue**](http://docs.google.com/SimpleSound.html#setSampleValue(int,%20int))(int frameNum, int sample)            Changes the value of the sample found at the specified frame. |
| void | [**setSampleValueAt**](http://docs.google.com/SimpleSound.html#setSampleValueAt(int,%20int))(int index, int value)            Method to set the sample value at the passed index to the passed value |
| void | [**setSoundExplorer**](http://docs.google.com/SimpleSound.html#setSoundExplorer(SoundExplorer))([SoundExplorer](http://docs.google.com/SoundExplorer.html) soundExplorer)            Changes the explorer of this object. |
| [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) | [**toString**](http://docs.google.com/SimpleSound.html#toString())()            Obtains a string representation of this JavaSound. |
| void | [**write**](http://docs.google.com/SimpleSound.html#write(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) fileName)            Method to write this sound to a file |
| void | [**writeToFile**](http://docs.google.com/SimpleSound.html#writeToFile(java.lang.String))([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) outFileName)            Creates an audioInputStream from this sound, and then writes this stream out to the file with the specified name. |

| **Methods inherited from class java.lang.**[**Object**](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) |
| --- |
| [clone](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#clone()), [equals](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#equals(java.lang.Object)), [finalize](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#finalize()), [getClass](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#getClass()), [hashCode](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#hashCode()), [notify](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#notify()), [notifyAll](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#notifyAll()), [wait](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#wait()), [wait](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#wait(long)), [wait](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#wait(long,%20int)) |

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

### MAX\_NEG

public static final int **MAX\_NEG**

Constant for max negative value

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#SimpleSound.MAX_NEG)

### MAX\_POS

public static final int **MAX\_POS**

Constant for max positive value

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#SimpleSound.MAX_POS)

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

### SimpleSound

public **SimpleSound**()

Constructs a SimpleSound of 3 seconds long.

### SimpleSound

public **SimpleSound**(int numFrames)

Constructs a SimpleSound of the specified length. This sound will simply consist of an empty byte array, and an AudioFileFormat with the following values:

* AudioFileFormat.Type.WAVE
* 22.05K sampling rate
* 16 bit sample
* 1 channel
* signed PCM encoding
* small-endian byte order

Note that no new sound file is created, we only represent the sound with a buffer and the AudioFileFormat. If a file is desired, then the method writeToFile(String filename) must be called on this newly created sound.

**Parameters:**numFrames - the number of samples in the sound**See Also:**[write(String filename)](http://docs.google.com/SimpleSound.html#write(java.lang.String))

### SimpleSound

public **SimpleSound**(int numFrames,  
 int sampleRate)

Constructs a SimpleSound of the specified length. This sound will simply consist of an empty byte array, and an AudioFileFormat with the following values:

* AudioFileFormat.Type.WAVE
* 22.05K sampling rate
* 16 bit sample
* 1 channel
* signed PCM encoding
* small-endian byte order

Note that no new sound file is created, we only represent the sound with a buffer and the AudioFileFormat. If a file is desired, then the method writeToFile(String filename) must be called on this newly created sound.

**Parameters:**numFrames - the number of samples in the sound**See Also:**[write(String filename)](http://docs.google.com/SimpleSound.html#write(java.lang.String))

### SimpleSound

public **SimpleSound**(int sampleSizeInBits,  
 boolean isBigEndian)

Constructs a simple sound with the given sample size in bits and type of endian (big or little)

### SimpleSound

public **SimpleSound**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) fileName)

Constructs a new SimpleSound from the given file.

**Parameters:**fileName - The File from which to create this sound.**See Also:**[loadFromFile(String filename)](http://docs.google.com/SimpleSound.html#loadFromFile(java.lang.String))

### SimpleSound

public **SimpleSound**([SimpleSound](http://docs.google.com/SimpleSound.html) sound)

Constructor that creates a new SimpleSound by copying a passed SimpleSound

**Parameters:**sound - the sound to copy

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

### getBuffer

public byte[] **getBuffer**()

Method that returns the byte array representation of this simple sound.

**Returns:**the sound represented as a byte array

### getAudioFileFormat

public [AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true) **getAudioFileFormat**()

Method that returns the AudioFileFormat describing this simple sound.

**Returns:**the AudioFileFormat describing this sound**See Also:**[AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true)

### getSamplingRate

public double **getSamplingRate**()

Method to get the sampling rate of this sound

**Returns:**the sampling rate in number of samples per second

### getSoundExplorer

public [SoundExplorer](http://docs.google.com/SoundExplorer.html) **getSoundExplorer**()

Method that returns the SoundExplorer

**Returns:**the sound explorer

### asArray

public byte[] **asArray**()

Method to return the byte array

**Returns:**an array of bytes which represents the simple sound**See Also:**[getBuffer()](http://docs.google.com/SimpleSound.html#getBuffer())

### getPlaybacks

public [Vector](http://java.sun.com/javase/6/docs/api/java/util/Vector.html?is-external=true) **getPlaybacks**()

Method that returns the vector of playback threads currently active on this sound.

**Returns:**the vector of playback threads for this simple sound

### getFileName

public [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) **getFileName**()

Method that returns the name of the file this sound came from. If this sound did not originate with a file, this value will be null.

**Returns:**the file name associated with this sound or null**See Also:**[loadFromFile(String fileName)](http://docs.google.com/SimpleSound.html#loadFromFile(java.lang.String))

### getDEBUG

public boolean **getDEBUG**()

Method to get the value of the debug flag

**Returns:**true if in debug mode else false

### setBuffer

public void **setBuffer**(byte[] newBuffer)

Changes the byte array that represents this sound.

**Parameters:**newBuffer - a byte array representation of the new sound we want this to represent.

### setAudioFileFormat

public void **setAudioFileFormat**([AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true) newAudioFileFormat)

Changes the AudioFileFormat of this sound.

**Parameters:**newAudioFileFormat - the new audioFileFormat that describes this sound.**See Also:**[AudioFileFormat](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioFileFormat.html?is-external=true)

### setSoundExplorer

public void **setSoundExplorer**([SoundExplorer](http://docs.google.com/SoundExplorer.html) soundExplorer)

Changes the explorer of this object.

**Parameters:**soundExplorer - the new SoundExplorer to use**See Also:**[SoundExplorer](http://docs.google.com/SoundExplorer.html)

### makeAIS

public [AudioInputStream](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioInputStream.html?is-external=true) **makeAIS**()

Creates an AudioInputStream for this sound from the buffer and the audioFileFormat.

**Returns:**an AudioInputStream representing this sound.**See Also:**[AudioInputStream](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioInputStream.html?is-external=true)

### printError

public void **printError**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message)

Invokes printError(message, null)

**Throws:** [SoundException](http://docs.google.com/SoundException.html) - Will throw under every circumstance. This way we can catch the exception in JES.**See Also:**[printError(String message, Exception e)](http://docs.google.com/SimpleSound.html#printError(java.lang.String,%20java.lang.Exception))

### printError

public void **printError**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) message,  
 [Exception](http://java.sun.com/javase/6/docs/api/java/lang/Exception.html?is-external=true) e)

Prints the given String to the "standard" error output stream, then prints a stack trace on the exception, and then exits the program. If the String is null, then nothing happens, the method just returns. If the Exception is null, then it prints the String and then exits the program.

**Parameters:**message - A description of the errore - The exception, if any, that was caught regarding the error **Throws:** [SoundException](http://docs.google.com/SoundException.html) - Will throw under every circumstance. This way we can catch the exception in JES.

### isStereo

public boolean **isStereo**()

Method to check if a sound is stereo (2 channels) or not

**Returns:**true if in stereo else false

### write

public void **write**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) fileName)

Method to write this sound to a file

**Parameters:**fileName - the name of the file to write to

### writeToFile

public void **writeToFile**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) outFileName)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Creates an audioInputStream from this sound, and then writes this stream out to the file with the specified name. If no file exists, one is created. If a file already exists, then it is overwritten. This does not check the extension of the fileName passed in to make sure it agrees with the AudioFileFormat.Type of this sound.

**Parameters:**outFileName - The name of the file to write this sound to **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if any error is encountered while writing to the file.

### loadFromFile

public void **loadFromFile**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) inFileName)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Resets the fields of this sound so that it now represents the sound in the specified file. If successful, the fileName ariable is updated such that it is equivalent to inFileName.

**Parameters:**inFileName - the path and filename of the sound we want to represent. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if any problem is encountered while reading in from the file.

### play

public void **play**()

Creates a new Playback thread and starts it. The thread is guarranteed to finish playing the sound as long as the program doesn't exit before it is done. This method does not block, however. So, if you invoke play() multiple times in a row, sounds will simply play on top of eachother - "accidental mixing"

**See Also:**[Playback](http://docs.google.com/Playback.html)

### blockingPlayOld

public void **blockingPlayOld**()

Creates a new Playback thread, starts it, then waits for the entire sound to finish playing before it returns. This method is guarranteed to play the entire sound, and does not allow for any "accidental mixing"

**See Also:**[Playback](http://docs.google.com/Playback.html)

### blockingPlay

public void **blockingPlay**()

Plays the sound, then sleeps for how long the sound SHOULD last.

### playAtRateDur

public void **playAtRateDur**(double rate,  
 double durInFrames)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Checks the value of durInFrames to make sure that it is not larger than Integer.MAX\_VALUE to guarrantee safe casting. Also checks the value of rate to make sure that it is not larger than Float.MAX\_VALUE before casting.

**Parameters:**rate - a double representing the change in sampleRate (==frameRate) for playing back this sounddurInFrames - a double representing how much of this sound we want to play. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if there are problems playing the sound.**See Also:**[playAtRateInRange(float rate, int startFrame, int endFrame, boolean isBlocking)](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int,%20boolean))

### blockingPlayAtRateDur

public void **blockingPlayAtRateDur**(double rate,  
 double durInFrames)  
 throws [SoundException](http://docs.google.com/SoundException.html)

First, checks the value of durInFrames to make sure that it is not larger than Integer.MAX\_VALUE to guarrantee safe casting. Simmilarly, checks the value of rate to make sure that it is not larger than FLoat.MAX\_VALUE before casting.

**Parameters:**rate - a double representing the change in sampleRate (==frameRate) for playing back this sounddurInFrames - a double representing how much of this sound we want to play **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if there are problems playing the sound.**See Also:**[playAtRateInRange(float range, int startFrame, int endFrame, boolean isBlocking)](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int,%20boolean))

### playAtRateInRange

public void **playAtRateInRange**(float rate,  
 int startFrame,  
 int endFrame)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Calls playAtRateInRange(rate, startFrame, endFrame, false) .

**Parameters:**rate - a float representing the change in sampleRate (==frameRate) for playing back this soundstartFrame - an int representing the frame at which we want to begin playing the soundendFrame - an int representing the frame at which want to stop playing the sound **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if there are problems playing the sound.**See Also:**[playAtRateInRange(float range, int startFrame, int endFrame, boolean isBlocking)](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int,%20boolean))

### blockingPlayAtRateInRange

public void **blockingPlayAtRateInRange**(float rate,  
 int startFrame,  
 int endFrame)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Calls playAtRateInRange(rate, startFrame, endFrame, true) .

**Parameters:**rate - a float representing the change in sampleRate (==frameRate) for playing back this soundstartFrame - an int representing the frame at which we want to begin playing the soundendFrame - an int representing the frame at which want to stop playing the sound **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if there are problems playing the sound.**See Also:**[playAtRateInRange(float range, int startFrame, int endFrame, boolean isBlocking)](http://docs.google.com/SimpleSound.html#playAtRateInRange(float,%20int,%20int,%20boolean))

### playAtRateInRange

public void **playAtRateInRange**(float rate,  
 int startFrame,  
 int endFrame,  
 boolean isBlocking)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Plays the specified segment of this sound at the given sample rate. Then it saves the old fields (buffer and audioFileFormat) of this sound into temporary variables, and setting the fields of this sound to modified values. Then it creates a Playback thread on this sound (with the modified values) and starts the thread. The values for buffer and audioFileFormat are restored to their original values before the method returns.

**Parameters:**rate - The change in the sampleRate (==frameRate) for playing back this sound. The old SampleRate is multiplied by this value. So, if rate = 2, the sound will play twice as fast (half the length), and if rate = .5, the sound will play half as fast (twice the length).startFrame - The index of the frame where we want to begin playendFrame - The index of the frame where we want to end playisBlocking - If true, this method waits until the thread is done playing the sound before returning. If false, it simply starts the thread and then returns. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if there are any problems playing the sound.

### removePlayback

public void **removePlayback**([Playback](http://docs.google.com/Playback.html) playbackToRemove)

Deletes the specified playback object from the Vector. This should only be called from within the run() method of an individual playback thread.

**See Also:**[Playback.run()](http://docs.google.com/Playback.html#run())

### getFrame

public byte[] **getFrame**(int frameNum)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Returns an array containing all of the bytes in the specified frame.

**Parameters:**frameNum - the index of the frame to access **Returns:**the array containing all of the bytes in frame frameNum **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frame number is invalid.

### getLengthInFrames

public int **getLengthInFrames**()

Obtains the length of the audio data contained in the file, expressed in sample frames.

**Returns:**the number of sample frames of audio data in the file

### getNumSamples

public int **getNumSamples**()

Returns the number of samples in this sound

**Returns:**the number of sample frames

### getSample

public [SoundSample](http://docs.google.com/SoundSample.html) **getSample**(int frameNum)

Method to create and return a SoundSample object for the given frame number

**Returns:**a SoundSample object for this frame number

### getSamples

public [SoundSample](http://docs.google.com/SoundSample.html)[] **getSamples**()

Method to create and return an array of SoundSample objects

**Returns:**the array of SoundSample objects

### getSampleValueAt

public int **getSampleValueAt**(int index)

Method to get the sample at the passed index and handle any SoundExceptions

**Parameters:**index - the desired index **Returns:**the sample value

### getSampleValue

public int **getSampleValue**(int frameNum)  
 throws [SoundException](http://docs.google.com/SoundException.html)

If this is a mono sound, obtains the single sample contained within this frame, else obtains the first (left) sample contained in the specified frame.

**Parameters:**frameNum - the index of the frame to access **Returns:**an integer representation of the bytes contained within the specified frame **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frame number is invalid.

### getLeftSample

public int **getLeftSample**(int frameNum)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Obtains the left sample of the audio data contained at the specified frame.

**Parameters:**frameNum - the index of the frame to access **Returns:**an int representation of the bytes contained in the specified frame. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frameNumber is invalid

### getRightSample

public int **getRightSample**(int frameNum)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Obtains the right sample of the audio data contained at the specified frame.

**Parameters:**frameNum - the index of the frame to access **Returns:**an int representation of the bytes contained in the specified frame. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frameNumber is invalid, or the encoding isn't supported.

### getLengthInBytes

public int **getLengthInBytes**()

Obtains the length of this sound in bytes. Note, that this number is not neccessarily the same as the length of this sound's file in bytes.

**Returns:**the sound length in bytes

### getLength

public int **getLength**()

Method to return the length of the sound as the number of samples

**Returns:**the length of the sound as the number of samples

### getChannels

public int **getChannels**()

Obtains the number of channels of this sound.

**Returns:**the number of channels (1 for mono, 2 for stereo), or AudioSystem.NOT\_SPECIFIED**See Also:**[AudioSystem.NOT\_SPECIFIED](http://java.sun.com/javase/6/docs/api/javax/sound/sampled/AudioSystem.html?is-external=true#NOT_SPECIFIED)

### setFrame

public void **setFrame**(int frameNum,  
 byte[] theFrame)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Changes the value of each byte of the specified frame.

**Parameters:**frameNum - the index of the frame to changetheFrame - the byte array that will be copied into this sound's buffer in place of the specified frame. **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frameNumber is invalid.

### setSampleValueAt

public void **setSampleValueAt**(int index,  
 int value)

Method to set the sample value at the passed index to the passed value

**Parameters:**index - the indexvalue - the new value

### setSampleValue

public void **setSampleValue**(int frameNum,  
 int sample)  
 throws [SoundException](http://docs.google.com/SoundException.html)

Changes the value of the sample found at the specified frame. If this sound has more than one channel, then this defaults to setting only the first (left) sample.

**Parameters:**frameNum - the index of the frame where the sample should be changedsample - an int representation of the new sample to put in this sound's buffer at the specified frame **Throws:** [SoundException](http://docs.google.com/SoundException.html) - if the frameNumber is invalid, or another problem is encountered

### setLeftSample

public void **setLeftSample**(int frameNum,  
 int sample)  
 throws [SoundException](http://docs.google.com/SoundException.html)

**Throws:** [SoundException](http://docs.google.com/SoundException.html)

### setRightSample

public void **setRightSample**(int frameNum,  
 int sample)  
 throws [SoundException](http://docs.google.com/SoundException.html)

**Throws:** [SoundException](http://docs.google.com/SoundException.html)

### explore

public void **explore**()

Method to open a sound viewer on a copy of this sound

### playNote

public static void **playNote**(int key,  
 int duration,  
 int intensity)

Method to play a note using MIDI

**Parameters:**key - the piano key to playduration - how long to play the noteintensity - how hard to strike the note from (0-127)

### convert

public static void **convert**([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) mp3File,  
 [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) wavFile)

Method to convert a mp3 sound into a wav sound

**Parameters:**mp3File - wavFile -

### toString

public [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) **toString**()

Obtains a string representation of this JavaSound.

**Overrides:**[toString](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true#toString()) in class [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html?is-external=true) **Returns:**a String representation of this JavaSound.

| | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-all.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | | |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/SimplePicture.html)   [**NEXT CLASS**](http://docs.google.com/SimpleTurtle.html) | [**FRAMES**](http://docs.google.com/index.html?SimpleSound.html)    [**NO FRAMES**](http://docs.google.com/SimpleSound.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#2jxsxqh) |