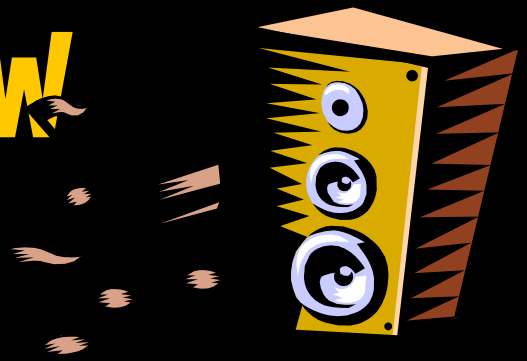
Sound in Java: A Brief Overview



jonas.kvarnstrom@liu.se – 2015

Sound in Java

jonkv@ida

- Early Java: Very simple sound API
  - Can play, loop or stop audio clips
  - Only handles **.AU audio files** – 8-bit sound, mono, 8 kHz
  - Can only be used in **applets** (in web pages), not in applications

- Java 1.2: Some **extensions**
  - Handles **.AU, .AIFF and .WAV files**, 8- and 16-bit, mono/stereo, 8-48 kHz
  - Mixes 64 channels
  - Software MIDI wavetable synthesizer
  - Can be used in **applets and applications** (finally!)

Sound in Java: Example

jonkv@ida

- Example: Playing sounds in an application
  - Use a static method in Applet(!):
    public static final AudioClip newAudioClip(URL url)

```
import java.applet.Applet;
import java.net.URL;

public class SoundPlayer {
   public static void main(String[] args) {
      URL url = new URL(args[0]);  // Or get a URL from a resource
      AudioClip clip = Applet.newAudioClip(url);
      clip.play();
      …
      clip.stop(); // If you want to stop before the sound/music ends
   }
}
```

The Java Sound API

jonkv@ida

- Much more in Java Sound (Java 1.3 and later)
  - Packages javax.sound.midi, javax.sound.sampled
  - Sampling, processing, mixing, and playing

```
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import java.net.URL;

public class SoundPlayer2 {
   public static void main(String[] args) throws ... {
      URL url = new URL(...);  // Or get a URL from a resource
      AudioInputStream audioIn = AudioSystem.getAudioInputStream(url);
      Clip clip = AudioSystem.getClip();
      clip.open(audioIn);
      clip.loop(5);
   }
}
```

The Java Sound API

jonkv@ida

- Much more functionality – and complexity
  - Brief example on the following pages
  - Borrowed from http://www.wikijava.org/wiki/Play_a_wave_sound_in_Java
  - See for example the Java Tutorial for more information: http://download.oracle.com/javase/tutorial/sound/index.html

Java Sound

jonkv@ida

- **private final** static int **EXTERNAL_BUFFER_SIZE** = 524288;
- // Assumes you have a URL to an audio file,
  // for example a Java resource URL
  public void play(URL waveURL)
  throws UnsupportedAudioFileException, IOException,
  LineUnavailableException
  {
  // Open the URL as an audio stream
  AudioInputStream audioInputStream =
  AudioSystem.getAudioInputStream(waveURL);

  // Obtain information about the AudioInputStream
  AudioFormat audioFormat = audioInputStream.getFormat();
  DataLine.Info info = **new** DataLine.Info(SourceDataLine.class, audioFormat);

  // Open an audio channel
  SourceDataLine dataLine = (SourceDataLine) AudioSystem.getLine(info);
  dataLine.open(audioFormat, **EXTERNAL_BUFFER_SIZE**);

```
// Start playing (asynchronously)...
dataLine.start();

// But we haven't provided any data yet!
// Incrementally push more data into the SourceDataLine, up to 512k at a time
int readBytes = 0;
byte[] audioBuffer = new byte[EXTERNAL_BUFFER_SIZE];

try {
  while (readBytes != -1) {
    readBytes = audioInputStream.read(audioBuffer, 0, audioBuffer.length);
    if (readBytes >= 0) dataLine.write(audioBuffer, 0, readBytes);
  }
} finally {
  // End of file – play whatever is still in the buffer, then close the audioChannel
  dataLine.drain();
  dataLine.close();
}
}
```