



Java is a trademark of Sun Microsystems, Inc.

NYJavaSIG

JavaOneSM

Making Music with the JavaTM Programming Language

Frank D Greco

NYJavaSIG

www.javasig.com

PAN-5388

Java and Music

Creating music with Java

- > Software developers and Musicians share traits
- > JavaSound™, JMF, Java 3D sound API, JavaFX™
 - Good or bad??
- > Panelists - demonstrations
 - Andrew Brown – jMusic
 - Robert Keller – Impro-Visor
 - Nick Didkovsky – JMSL
 - David Koelle – JFugue



Java is a trademark of Sun Microsystems, Inc.



JavaOneSM

Java and Music: Libraries for User- Developers

Andrew R. Brown
Queensland University of Technology
jMusic & SoundCipher libraries

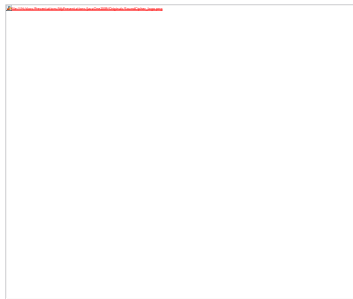
Java libraries for musicians

- > Musicians as users/developers
- > Developers as musicians
- > Feature for music libraries:
 - Discipline-friendly language
 - Music and sound integrated
 - Timing is everything
 - “Simple things should be simple”
 - Tutorials and documentation
 - Play everywhere
 - Creation leads to capture



JavaOneSM

Thank You



Andrew r. Brown
a.brown@qut.edu.au

<http://jmusic.ci.qut.edu.au>
<http://soundcipher.com>





Java is a trademark of Sun Microsystems, Inc.

JavaOneSM

Java Music Specification Language (JMSL): a Java API for Music

Nick Didkovsky
Algomusic.com

JMSL Features

Polymorphic Hierarchical Scheduler

- > Polymorphic (“Composable” Interface):
 - MusicJob: sleep, do something, repeat
 - MusicShape: ordered multidimensional numeric data
 - Collection: Composables that contain Composables
- > Hierarchical (parent/child relationships)
 - Parallel Collections (string quartet)
 - Sequential Collections (song form)
- > Scheduler:
 - Parent gets launch() command and a timestamp, schedules children

```
SequentialCollection seqCol = new SequentialCollection();  
seqCol.add(myMusicShape);  
seqCol.add(myMusicJob);  
seqCol.launch(JMSL.now());
```

JMSL Features

Instrument Interface

- > JMSL supports various music devices
 - JavaSound MIDI
 - MIDIShare (midishare.sourceforge.net)
 - JSyn: Java API for sound synthesis (www.softsynth.com)
 - Other

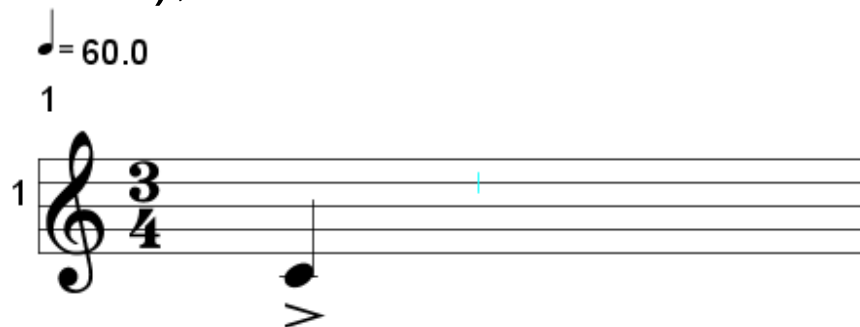
```
public double play(double timeStamp, double timeStretch, double[] perfData) {  
    double duration = perfData[0];  
    // do something with other perfData  
    return timeStamp + duration * timeStretch;  
}
```

JMSL Features

Notated music and non-notated music

- > Standard Music Notation supported
 - Create scores programmatically, or with GUI
 - Plug-ins for note transformation
 - Export to MusicXML, Lilypond, SCORE, Midifile

```
Score score = new Score(1, 800, 600);  
score.addMeasure(3, 4);  
Note n = score.addNote(1.0, 60, 0.5, 0.8);  
n.setMark(Note.MARK_ACCENT);
```





JavaOneSM

Thank You

Nick Didkovsky
nick@didkovsky.com

www.algomusic.com





Java is a trademark of Sun Microsystems, Inc.

JavaOneSM

Java meets Jazz

Robert M. Keller
Harvey Mudd College
Claremont, California 91711

Impro-Visor

Tool to assist musicians learning to improvise

- > Conventional wisdom:
 - Transcribe solos of great masters from recordings

- > Proposed alternate wisdom:
 - Compose one's own solos off-line
 - Don't play exactly as composed (improvise!)
 - Student learns more about the tune
 - More exploration encouraged
 - Student owns the result

Impro-Visor Provides

- > Musical knowledge (chord spellings, scales, etc.)
- > Suggestions
 - Database of cells, idioms, licks, quotes
 - Instant generation of new licks (melodic material over chords)
- > AI Elements
 - Style extraction (Derive styles from MIDI + chords)
 - Grammar learning (for lick generation)

Impro-Visor Provides

- > Visual feedback (“in” vs “out” notes)
- > Aural feedback (chord accompaniment on entry)
- > Auto-generated accompanying rhythm section
 - piano, bass, drums

Impro-Visor Screenshot

The screenshot displays the Impro-Visor software interface for a 12-Bar Blues piece. The window title is "Impro-Visor: 12-Bar Blues". The menu bar includes File, Edit, View, Play, Utilities, Window, Preferences, and Help. The toolbar contains various icons for file operations, editing, and playback. The playback controls show a playback location from 0:00 to 0:16, a looping button set to 2, a mute button, a volume slider, a tempo of 180.0 BPM, and transpose/tracker delay settings. A "Textual Entry" field is present with a "Clear" button. The main score area is titled "Chorus 1" and "12-Bar Blues". It shows a musical score in 4/4 time with a swing style. The score is divided into three lines of four bars each. The first line contains measures 1-4 with chords F13, Bb13, Bo7, F13, Cm9, and F13b9. The second line contains measures 5-8 with chords Bb13, Bo7, F13, and D7#5#9. The third line contains measures 9-12 with chords Gm9, C13b9, F13, D7#5#9, Gm9, and C13b9. A blue box highlights the first four notes of measure 10.

Pure-Text Lead-Sheet Files

```

Editor For: 12-Bar Blues
-----
Leadsheet to Editor

(title 12-Bar Blues)
(composer )
(show )
(year )
(comments )
(meter 4 4)
(key -1)
(tempo 180.0)
(volume 127)
(playback-transpose 0)
(bass-instrument 33)
(bass-volume 31)
(drum-volume 48)
(chord-volume 37)
(breakpoint 54)
(layout)
(style swing
  (swing 0.67)
  (comp-swing 0.67)
  (bass-high g-)
  (bass-low g---)
  (bass-base c--)
  (chord-high a)
  (chord-low b--)
  (chord-base c- e- g-)
)
(part
  (type chords)
  (title )
  (composer )
  (instrument 0)
  (volume 65)
  (key -1)
)

```

Meta-Data

```

(section (style swing))
F13 | Bb13 Bo7 | F13 | Cm9 F13b9 |
Bb13 | Bo7 | F13 | D7#5#9 |
Gm9 | C13b9 | F13 D7#5#9 | Gm9 C13b9 |

(part
  (type melody)
  (title )
  (composer )
  (instrument 11)
  (volume 85)
  (key -1)
  (stave treble)
)
r4 f+4 d+8 c+8 a8 g8

f8 d8 f4 ab4 r2

f+8 c#+8 d+8 c+8 a8 g8

f8 d8 f4 eb4 r4+8

d8 f8 g8 ab4 d4

f4+8 ab8 r8 r2

f8 c+8 bb8 a8 c+8 d+4

eb+4 f+8 d+8 r2+8

bb8 d+8 b1+2+8

r1+1

```

Chords

Melody

```

-----
Editor to Leadsheet

```

Implementation

- > Java 1.5 (on MacOSx, Windows, Linux)
- > Adapted version of jMusic circa June 2005
- > All sound is MIDI
- > Netbeans IDE
- > Musical information is open-format text files
 - Readable S-expressions (no XML)
 - Exports MIDI
- > GPL

Acknowledgment

Students at Harvey Mudd College and NSF REU

- > Stephen Jones Steve Gomez (Dartmouth)
- > Aaron Wolin Brandy McMenemy (Carlton)
- > Martin Hunt Jon Gillick (Wesleyan U.)
- > David Morrison Kevin Tang (Cornell U.)
- > Sayuri Soejima Chad Waters (Winthrop U.)
- > Emma Carlson John Goodman (U.K.)
- > Stephen Lee

- > Support by Mellon Foundation, Baker Foundation,
National Science Foundation



JavaOneSM

Thank You

Robert M. Keller
keller@cs.hmc.edu
www.impro-visor.com



Java is a trademark of Sun Microsystems, Inc.

JavaOneSM

Teaching and Tabbing Guitar with JFrets

Matt Warman
Creator of JFrets
Senior Consultant
STARBASE Consulting Inc.



What Is JFrets?

- Teaches guitar in an interactive desktop tool
- Displays notes, chords, and scales
- Plays sounds to aid in learning process
- Provides tutorials and exercises
- Ability to create and save guitar tablature
- Guitar tablature playback

JFrets Capabilities

- Creates songs in tab or note format
- Provides a Metronome
- Provides a Scale Player
- Contains a guitar tuner with various tunings
- Displays left-handed chords
- Prints tabs or songs

Audio Streaming Code

```
try
{
    File audio = new File(getClass().getResource("/org/jfrets/sounds/" + file).toURI());
    float sample = 128000;
    AudioInputStream ais = AudioSystem.getAudioInputStream(audio);
    AudioFormat af = ais.getFormat();
    AudioFormat target = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, af.getSampleRate(), 16, af.getChannels(),
    af.getChannels() * 2, af.getSampleRate(), false);
    AudioInputStream decode = AudioSystem.getAudioInputStream(target, ais);
    DataLine.Info info = new DataLine.Info(SourceDataLine.class, target);
    SourceDataLine line = (SourceDataLine) AudioSystem.getLine(info);
    line.open(target);
    if (line != null)
    {
        byte[] data = new byte[4096];
        line.start();
        int bytesRead;
        while ((bytesRead = decode.read(data, 0, data.length)) != -1)
        {
            line.write(data, 0, bytesRead);
        }
        line.drain();
        line.stop();
        line.close();
        decode.close();
    }
} catch (IOException e) {
    /* handle this exception */
} catch (LineUnavailableException e) {
    /* handle this exception */
} catch (URISyntaxException e) {
    /* handle this exception */
} catch (UnsupportedAudioFileException e) {
    /* handle this exception */
}
}
```

**Don't worry,
you're not supposed
to be able to read
this.**

**And you certainly
shouldn't have to
program it.**

JFugue Code

```
Player player = new Player();
String patternString =
    getMidiType((String) this.voiceBox.getSelectedItem())
    + noteName;
patternString = "T" + bpm + " " + patternString;
Pattern pattern = new Pattern(patternString);
player.play(pattern);
```



JavaOneSM

Thank You

Matt Warman
mattwrock@gmail.com
<https://jfrets.dev.java.net>



JavaOneSM

Thank You

NYJavaSIG

Frank D. Greco
NYJavaSIG
fgreco@javasig.com

www.javasig.com

