JFugue

Java API for Music Programming

JFugue: Making Music With Java MIDI and Illustrating API Usability

David Koelle
Senior Software Engineer
Charles River Analytics Inc
http://www.cra.com
http://www.jfugue.org

Geertjan Wielenga
Technical Writer
NetBeans
http://www.netbeans.org
http://blogs.sun.com/geertjan

TS-1130

Goal

What we hope you'll take away

Learn about JFugue, an API for creating MIDI music, and learn how an easy-to-use API can make your projects successful.

Agenda

Explore JFugue

Enjoy Demos!

Create JFugue Clients

Examine API Usability

Agenda

Explore JFugue

Enjoy Demos!

Create JFugue Clients

Examine API Usability

What Is JFugue?

- An API for Programming Music in Java™ programming language
- Renders music in Java™ platform MIDI
  - … extensible to other formats (more later)
- Intended for multiple purposes
  - Define and play music at runtime
  - Experiment with changing and editing music
  - Inspire future programmers
- Without JFugue, programming music is hard!

Programming Music With Java Platform MIDI

```
// Play a Middle-C
Sequencer sequencer = MidiSystem.getSequencer();
Sequence sequence = sequencer.getSequence();
Track track = sequence.createTrack();
ShortMessage onMessage = new ShortMessage();
onMessage.setMessage(ShortMessage.NOTE_ON, 0, 60, 128);
MidiEvent noteOnEvent = new MidiEvent(onMessage, 0);
track.add(noteOnEvent);
ShortMessage offMessage = new ShortMessage();
offMessage.setMessage(ShortMessage.NOTE_OFF, 0, 60, 128);
MidiEvent noteOffEvent = new MidiEvent(offMessage, 200);
track.add(noteOffEvent);
sequencer.start();
try {
    Thread.sleep(track.ticks());
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}
```

Programming Music With JFugue

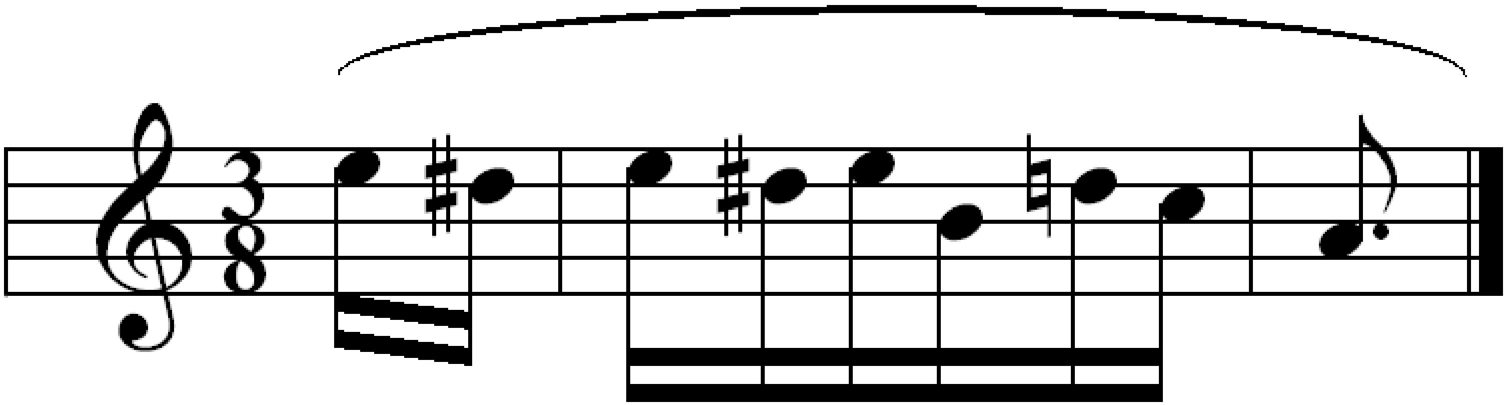
```
// Play a Middle-C

Player player = new Player();
player.play(“C”);
```

Programming Music With JFugue

```
// Play first 2 measures (and a bit) of “Für Elise”

Player player = new Player();
player.play(“E6s D#6s | E6s D#6s E6s B5s D6s C6s | A5i.”);
```



The Magic of JFugue

Why JFugue makes music programming fun

- Simple and intuitive API—player.play()
- Innovative “Music String”
  - Seems to break object-oriented paradigm, but…
  - More convenient for specifying many notes

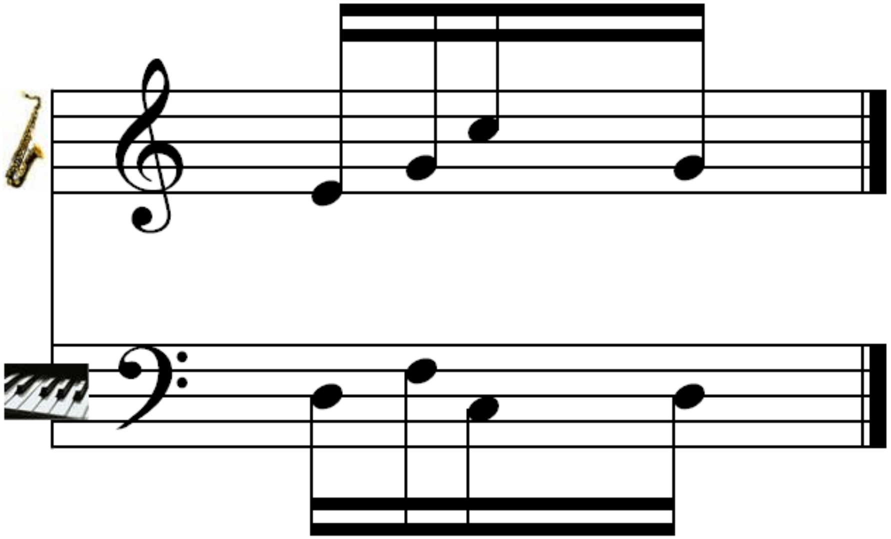
    song.add(new Note(Note.A_SHARP,6, Note.QUARTER));

    vs.

    play(“A#6q”);
  - Easy to specify all sorts of musical events
    - Notes, Durations, Instruments, Voices, Controller Events…
    - If it makes a sound in MIDI, you can represent it in JFugue

More Fun With the Music String

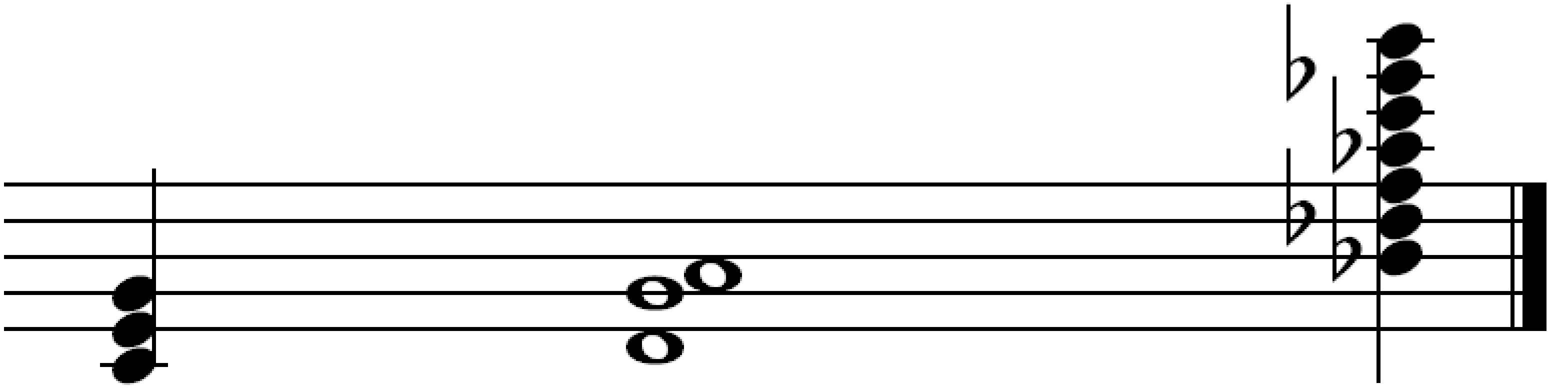
- Voices and Instruments



V0 I[Alto_Sax] E5s G5s C6s G5s
V1 I[Piano] D4s F4s C4s D4s

More Fun With the Music String

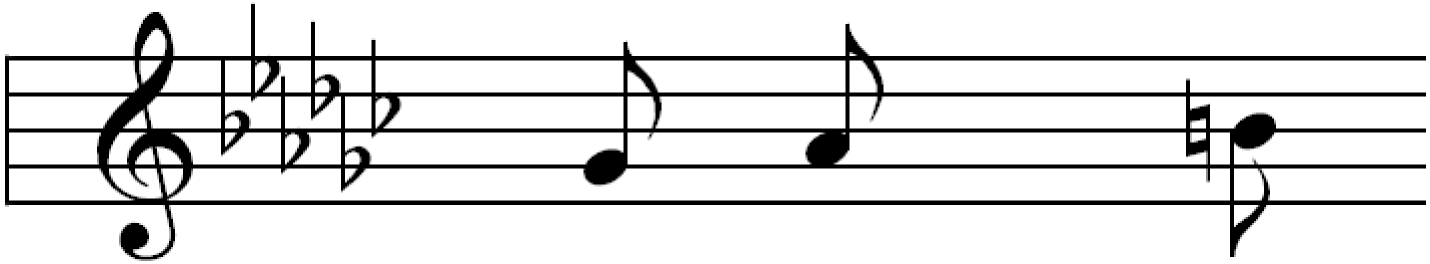
- Chords



Cmajq   Dsus4w   Bbmin13q

More Fun With the Music String

- Key Signatures



kGbmaj G5i A5i Bn5i

The G and A are automatically played as flats, the B has been declared natural

More Fun With the Music String

- Constants let you specify substitution values
  - To define: $*word*=*definition*
  - To use: [*word*]
- Example:
  - “$base=C [base]4q [base]majw”
  - Actually plays “C4q Cmajw”
  - Want to change all C notes to E? Just change $base
  - Instrument substitution: “$myFave=Piano I[myFave] C6q D6q”

More Fun With the Music String

- Pitch Bend
- Channel Pressure
- Polyphonic Pressure
- MIDI Controllers

Programming Music With JFugue

```
GrammarRewriter generator = new GrammarRewriter();

generator.setAxiom("T120 V0 I[Flute] Rq C5q " +
"V1 I[Tubular_Bells] Rq Rq Rq G6i+D6i V2 I[Piano] Cmajw E6q "+ "V3
I[Voice] E6q G6i+D6i V4 I[Choir] C5q E6q");

generator.addTransform("Cmajw", "Cmajw Fmajw");
generator.addTransform("Fmajw", "Rw Emajw");
generator.addTransform("Emajw", "Rw Fmajw");
generator.addTransform("C5q", "C5q G5q E6q C6q");
generator.addTransform("E6q", "G6q D6q F6i C6i D6q");
generator.addTransform("G6i+D6i", "Rq Rq G6i+D6i G6i+D6i Rq");

String music = generator.generate(3);
Pattern pattern = new Pattern(music);
Player player = new Player();
player.play(pattern);
```

What If You Could Manipulate Music?

How JFugue enables musical experimentation

- A Pattern is a fragment of music
- Patterns can be twisted, pulled, contorted…
  - PatternTransformer
  - Examples:
    - Duration Pattern Transformer
  - Bach wrote a song using a melody that was reversed and played on top of itself—The Crab Canon
    - Reverse Pattern Transformer
- PatternTransformers listen to the JFugue parser and create alternate patterns

Anonymous PatternTransfomer

```
// Lower the octave of each note in a pattern
//  (Number of notes in one octave = 12)

PatternTransformer octaveChanger = new PatternTransformer() {
    public void noteEvent(Note note) {
        byte currentValue = note.getValue();
        if (currentValue > 12) {
            note.setValue((byte)(currentValue - 12));
            returnPattern.addElement(note);
        }
    }
};

Pattern octaveLowerSong = octaveChanger.transform(song);
```

What Else Is Cool?

More amazing things you can do in JFugue

- Microtonal music
  - JFugue automatically adjusts pitch bend to change/make microtonal adjustments
- Rhythms
  - JFugue lets you bang on your keyboard like a set of drums
- Follow along with or anticipate MIDI events
  - You’ll see this in the demo!

Microtones in JFugue

```
MicrotoneHelper microtone = new MicrotoneHelper();
microtone.put(“Be", 400.00);
microtone.put(“Bf", 405.50);
microtone.put(“Bt", 415.67);
microtone.put(“Bv", 429.54);

Pattern p = new Pattern("[Be]q [Bf]q [Bt]q [Bv]q");
new Player().play(microtone.convertPattern(p));
```

Rhythms in JFugue

```
Rhythm rhythm = new Rhythm();
rhythm.addSubstitution('O', "[ACOUSTIC_BASS_DRUM]s");
rhythm.addSubstitution('o', "[ACOUSTIC_SNARE]s");
rhythm.addSubstitution('\'', "[CLOSED_HI_HAT]s");
rhythm.addSubstitution('`', "[OPEN_HI_HAT]s");
rhythm.addSubstitution('.', "Rs");

rhythm.setLayer(1, "O.OO...O.OO....O");
rhythm.setLayer(2, "....o.......o...");
rhythm.setLayer(3, "'.`.'.`.'.`.'.`.");

Pattern pattern = rhythm.getPattern();
pattern.repeat(4);

Player player = new Player();
player.play(pattern);
```

JFugue and MIDI Devices

Interact with MIDI keyboard and synthesizers

- Send music to an external device
- Listen to music from an external device

Sending Music to a MIDI Device

```
// Send music to keyboard - without JFugue
MidiDevice.Info[] info = MidiSystem.getMidiDeviceInfo();

// Need to figure out which info[] to use - more lines, need user input!

MidiDevice device = MidiSystem.getMidiDevice(info[x]);
if (!(device.isOpen())) {
 device.open();
}

Receiver receiver = device.getReceiver();
Sequence sequence = MidiSystem.getSequence(midifile);

// Sort all of the MidiEvents in sequence by time - 30/40 more lines
MidiEvent[] events = // sequence sorted by time

// Dole out event messages according to elapsed time
long elapsedTime = 0;
for (int i = 0; i < events.length; i++) {
  MidiEvent event = events[i];
  MidiMessage message = event.getMessage();
  long timestamp = event.getTick();
  long deltaTime = timestamp - elapsedTime;
  elapsedTime = timestamp;
  try {
    // Need to figure out tempoFactor - another 10 lines!
    Thread.sleep((int)(deltaTime * tempoFactor));
  } catch (InterruptedException ex) {
    Thread.currentThread().interrupt();
  }
}
receiver.send(message, -1);
receiver.close();
device.close();
```

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

Sending Music to a MIDI Device

```
// Send music to keyboard – with JFugue

try
{
    MidiOutDevice device = new MidiOutDevice();
    sequence = MidiSystem.getSequence(midifile);
   // OR: sequence = player.getSequence(pattern);
    device.sendSequence(sequence);
}
catch (MidiUnavailableException e) { /* handle this */ }
catch (InvalidMidiDataException e) { /* handle this */ }
catch (IOException e) {/* handle this */ }
```

Parsers and Renderers

Reading and writing to limitless formats

- JFugue has a clear architectural design
  - **Parsers** convert some format into musical events
  - **Renderers** turn musical events into something meaningful
- Examples
  - Parsers: MusicStringParser, MidiParser
  - Renderers: MidiRenderer, MusicStringRenderer

Parsers and Renderers in JFugue

```
// General Example
XxxxParser parser = new XxxxParser();
XxxxRenderer renderer = new XxxxRenderer();
parser.addParserListener(renderer);
parser.parse(whatever object the parser can parse);
```

Parsers and Renderers in JFugue

```
// Specific: Convert MIDI into a JFugue MusicString
MidiParser parser = new MidiParser();
MusicStringRenderer renderer = new MusicStringRenderer();
parser.addParserListener(renderer);
parser.parse(MidiSystem.getSequence(file));
```

```
// Wishlist: Convert MusicXML Format into Sheet Music
// (neither parser/renderer currently exists)
MusicXmlParser parser = new MusicXmlParser();
SheetMusicRenderer renderer = new SheetMusicRenderer();
parser.addParserListener(renderer);
parser.parse(new File(“music.xml”));
```

Agenda

Explore JFugue

Enjoy Demos!

Create JFugue Clients

Examine API Usability

DEMO

Seeing (or Hearing) JFugue in Action

Agenda

Explore JFugue

Enjoy Demos!

Create JFugue Clients

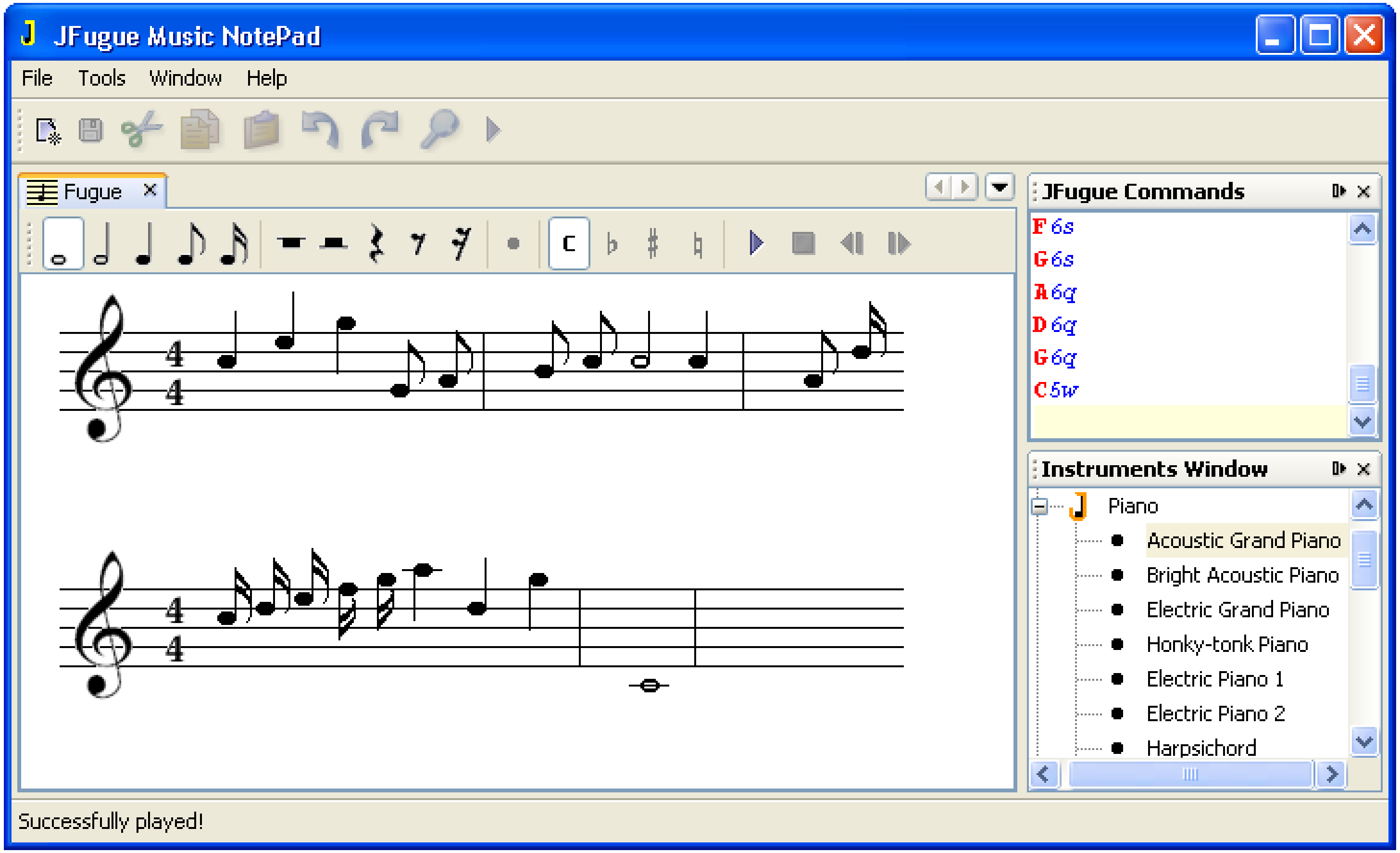
Examine API Usability

JFugue Provides Functionality…

…so a client **only** needs to provide a user interface

- To generate JFugue music strings
- To invoke the playing of JFugue music strings
- To invoke the saving of JFugue music strings
- To invoke the loading of MIDI files

Open Sourced JFugue Music Notepad



F 6s
G 6s
A 6q
D 6q
G 6q
C 5w

- Piano
  - Acoustic Grand Piano
  - Bright Acoustic Piano
  - Electric Grand Piano
  - Honky-tonk Piano
  - Electric Piano 1
  - Electric Piano 2
  - Harpsichord

Successfully played!

DEMO

JFugue Music NotePad

https://nbjfuguesupport.dev.java.net/

Agenda

Explore JFugue

Enjoy Demos!

Create JFugue Clients

Examine API Usability

What Is API Usability?

- Designing an interface for the user
  - Like usability design for graphical interfaces…
  - …but the users are other developers…
  - …so it’s easy to relate!
- “Interface” = your API
- “User” = other developers
- API Usability is the intersection of user-centered design and excellent coding practices

API Usability Tips

…illustrated through JFugue

- Start with the end in mind
  - Think to yourself: What do I want to accomplish?
- Develop examples as you develop your API
  - Example: JFugue’s Rhythm class

```
Rhythm rhythm = new Rhythm();
rhythm.addSubstitution('O', "[ACOUSTIC_SNARE]q");
rhythm.setLayer(1, "oo'O' oo'O' oo'O' oo'O' “);
Player player = new Player();
player.play(rhythm);
```

API Usability Tips

…illustrated through JFugue

- Make conceptually easy things simple to do
  - Player player = new Player();
  - player.play(“musical notes”);
- Create a compact API
  - Require the user to type as few lines as possible
    - song.add(pattern, 2); // Add the pattern twice
  - Don’t flood the API with unnecessary methods
    - Player had a “allNotesOff” method… thought I needed it, I was wrong

API Usability Tips

…illustrated through JFugue

- Be absolutely correct
  - If people are relying on your API, it must work!
  - Be available for comments and bugs
- Construct complete objects only
  - Don't rely on methods that the user must call after the construct your object…because they won't
- Catch errors right away
- Be verbose in reporting errors
  - Exception in thread "main" org.jfugue.JFugueException: The word DBF has no definition; Check the spelling, or define the word before using it

API Usability Tips

…illustrated through JFugue

- Follow Joshua Bloch's "Effective Java"
- Tips for evolving APIs
  - Once you release an API, people will rely on it
  - If you change the API, change the major version number of your release
  - Provide documentation for converting between versions
- Finally: The success of your API project also depends on your presentation
  - Webpage, communications, etc.

Summary

- JFugue lets you do wonderful things with music
- JFugue Music NotePad lets you build music graphically, and turn it into JFugue strings
- A usable API is important towards getting a programming library adopted and enjoyed

For More Information

- Java platform and music
  - Paul Lamere’s “Search Inside the Music”, TS-1548
- JFugue
  - JFugue—http://www.jfugue.org
  - The Complete Guide to JFugue
- Music NotePad
  - Music Notepad—https://nbjfuguesupport.dev.java.net
  - Geertjan’s blog—http://blogs.sun.com/geertjan
- API usability
  - Joshua Bloch’s Effective Java session and book
  - Dave Koelle’s website–http://www.DaveKoelle.com

Q&A

David Koelle

Geertjan Wielenga

JFugue:
Making Music With Java MIDI and Illustrating API Usability

David Koelle
Senior Software Engineer
Charles River Analytics Inc
http://www.cra.com
http://www.jfugue.org

Geertjan Wielenga
Technical Writer
NetBeans
http://www.netbeans.org
http://blogs.sun.com/geertjan

TS-1130