

1

Getting Started with JFugue

JFugue makes programming music easy. This chapter explains how to set up and get started with JFugue.

Downloading JFugue

Downloading JFugue is easy: go to <http://www.jfugue.org> and click on “Download”. Save the `jfugue.jar` file to your local system.

Personal Tip

When I download third-party libraries, I place them into a folder called “C:\Java Libraries”, where I extract the library’s compressed files, including source files and JavaDoc. When I need to use the jar file in a specific project, I copy the jar file into my project’s `lib` directory.

Running a Test Program

To be sure you are able to use JFugue after you download it, compile and run the following test program.

```
import org.jfugue.*;

public class MyMusicApp
{
    public static void main(String[] args)
```

Excerpt from *The Complete Guide to JFugue*

<http://www.jfugue.org/book.html>

```
{
    Player player = new Player();
    Pattern pattern = new Pattern("C D E F G A B");
    player.play(pattern);
    System.exit(0); // If using Java 1.4 or lower
}
}
```

To compile and run this program from a command prompt, follow the following steps (if you're using Eclipse, NetBeans, or another Java editor, you may jump ahead to the next section).

Step 1. To *compile* this program, enter this command at the command prompt, replacing %JFUGUE_DIR% with the directory into which you have placed `jfugue.jar`:

```
javac -classpath %JFUGUE_DIR%\jfugue.jar MyMusicApp.java
```

This will compile `MyMusicApp.java` and generate a `.class` file.

Step 2. To *run* the `.class` file, enter this line:

```
java -cp %JFUGUE_DIR%\jfugue.jar;. MyMusicApp
```

Be sure to copy this line exactly. The semicolon and period indicate where Java will find the `MyMusicApp` class – in the current (i.e., ".") directory.

Special Note for Mac Users

If you're using a Mac, replace the semicolon (;) with a colon (:).

You might wonder why it's necessary to put `System.exit(0);` at the end of the test program. Prior to Java 5.0 (also known as 1.5), the Java MIDI classes open a number of threads, but they're not all closed properly when the song is done playing (see Java bug [4735740](#)). This prevents the program from ending on its own, but it does not otherwise affect the execution of the program. Therefore, the `System.exit(0);` call is necessary to end the program when using versions of Java earlier than 5.0.

Using JFugue from an Integrated Development Environment

If you're using an Integrated Development Environment (IDE), like Eclipse or NetBeans, you'll need to include the JFugue jar file into your project. If you're using Eclipse (<http://www.eclipse.org>), go *Project > Properties*, select *Java Build Path*, select the *Libraries* tab, and click the "Add JARs..." or "Add External JARs..." button. Find `jfugue.jar` and add it to your project.

Personal Tip

For each of my projects, I create a `lib` directory, where I place third-party jar files.

To run the test program from Eclipse, right-click on the test program's filename and select *Run As...> Java Application*.

Deciding which version of JFugue to use

The latest version of JFugue is 4.0, and it is designed to work with Java 5.0 (a.k.a. Java 1.5) and later. A lot of MIDI bugs have been fixed in this version of Java. See <http://www.JFugue.org/download.html> for a link to the Java MIDI bugs fixed in Java 5.0.

If you're limited to an older version of Java, you can still use JFugue. JFugue version 2.1 works with Java versions 1.3 and 1.4. While JFugue 2.1 does not contain all of the latest features, you can still create music using most of the commonly used features of MusicStrings and Patterns. The JFugue download page at <http://www.JFugue.org/download.html> contains lists of changes between JFugue versions, so you can identify what features exist and do not exist in JFugue 2.1. Since JFugue 2.1 is no longer actively supported, the source files and JavaDoc are provided with the download.

Using MIDI Soundbanks

JFugue relies on Java's MIDI capabilities to produce music. Java MIDI uses the Java Sound engine, which in turn uses a soundbank to generate sounds using the synthesizer. A soundbank is a collection of audio samples for each instrument that are played by the synthesizer. A variety of soundbanks provided by Sun Microsystems are available for free download; some of these may provide richer sounds than the default soundbank that is packaged with the Java Runtime Environment (JRE).

In addition, there are third-party MIDI soundbanks that have incredibly rich sound. Many of these are available for purchase only. Try doing an online search for "midi sound bank" to see some examples.

Author on a Soapbox

MIDI is often ridiculed as producing dry, unemotional, dinky music – but in reality, MIDI is simply a format for communicating musical events between electronic musical instruments. The lack of freely available, widespread, symphonic-quality soundbanks is what makes many developers think that MIDI is not up to modern standards of music. Interestingly, this is a belief held more commonly by software developers than musicians. Fortunately, there are ways around this supposed limitation, as you'll see in the sections ahead.

Downloading Soundbanks

Soundbanks provided by Sun Microsystems can be downloaded from <http://java.sun.com/products/java-media/sound/soundbanks.html>. This page offers three soundbanks:

Minimal (0.35 MB)

This soundbank is packaged by default with Java SDK Standard Edition versions 1.2.2 and higher. It is the smallest soundbank available, and its sound samples are of slightly less quality than those found in the midsize soundbank.

Midsize (1.09 MB)

This soundbank shipped with Java2 versions 1.2 and 1.2.1.

Deluxe (4.92 MB)

This soundbank contains higher-quality sound samples.

Installing the Java Media Soundbanks

Installing a soundbank is as simple as moving the file you've downloaded to the correct directory.

First, download and unzip the soundbank you are interested in. You will see a file with a ".gm" extension.

On Windows computers, move this file to

`C:\Program Files\JavaSoft\JRE\<version>\lib\audio`. If there is no audio directory, create it. In addition, if you are using a Java SDK that you've downloaded, also copy the soundbank file to `<jdk-install-dir>\jre\lib\audio`.

On Linux or Solaris machines, move the soundbank file to

`<install-dir>/jre/lib/audio`. If the audio directory does not exist, create it.

Java will automatically use the highest-quality soundbank available, so if there is an existing soundbank file in the audio directory, you don't have to delete or rename it.

After you have moved your soundbank to the correct directory, be sure to exit any running Java programs. When you start them up again, they will use the new soundbanks.

Excerpt from *The Complete Guide to JFugue*
<http://www.jfugue.org/book.html>

In The Complete Guide to JFugue,
this chapter continues with the following section:

Using Gervill to Load Soundbanks

When Sun Microsystems released Java under an open source license, there were some interesting implications related to closed-source, licensed libraries that were used by the JDK. One of these libraries is the audio synthesis library, which is proprietary and cannot be released as open source software. In addition, the current audio synthesis engine used in Java can only use GM soundbank files, which is an unpublished, proprietary format that is not used as commonly as some other soundbank formats, such as SoundFont from Creative Technology Ltd. or Downloadable Sounds (DLS) from the MIDI Manufacturers Association Incorporated.

In response to this limitation, a project known as the Audio Synthesis Engine Project was started (see <http://openjdk.java.net/projects/audio-engine>) . The goal of this project is to create a new, open source version of Java's MIDI synthesizer.

Gervill is a software synthesizer created as a proposal for the Audio Synthesis Engine Project. It is open source, and is available at <https://gervill.dev.java.net>. It is also very easy to use; here are the steps to get Gervill working with your JFugue program (or any Java program that uses MIDI):

Read more in The Complete Guide to JFugue <http://www.jfugue.org/book.html>

Detailed Table of Contents of **The Complete Guide to JFugue**

<http://www.jfugue.org/book.html>

| | |
|---|-----------|
| Table of Contents | 7 |
| Detailed Table of Contents | 9 |
| Forward | 13 |
| Getting Started with JFugue | 15 |
| Downloading JFugue | 15 |
| Running a Test Program | 15 |
| Using JFugue from an Integrated Development Environment | 16 |
| Deciding which version of JFugue to use | 17 |
| Using MIDI Soundbanks | 17 |
| Downloading Soundbanks | 18 |
| Installing the Java Media Soundbanks | 18 |
| Using Gervill to Load Soundbanks | 19 |
| Using the JFugue MusicString | 21 |
| Introducing the MusicString | 21 |
| Learning the Parts of the MusicString | 22 |
| Notes, Rests, and Chords | 22 |
| Sharps, Flats, and Naturals | 23 |
| Octave | 23 |
| Chords | 24 |
| Chord Inversions | 25 |
| Duration | 26 |
| Triplets and Other Tuplets | 27 |
| Ties | 28 |
| Attack and Decay Velocities | 29 |
| Notes played in Melody and Harmony | 29 |
| Measure | 30 |
| Key Signature | 31 |
| Instrument | 31 |
| Voice | 34 |
| MIDI Percussion Track | 34 |
| Layer | 35 |
| Tempo | 36 |
| Pitch Wheel | 36 |
| Channel Pressure | 37 |
| Polyphonic Pressure | 37 |
| Controller Events | 37 |
| Constants | 40 |
| Timing Information | 41 |
| MusicString Style | 41 |
| JFugue Elements: Using Objects instead of MusicStrings | 42 |
| Getting Assistance with Notes | 45 |
| Transcribing Sheet Music to JFugue MusicString | 46 |
| Working with Patterns | 49 |
| What is a Pattern? | 49 |
| Using Patterns as Musical Building Blocks | 50 |

Excerpt from The Complete Guide to JFugue
<http://www.jfugue.org/book.html>

| | |
|--|-----------|
| Using Patterns to Construct Music | 51 |
| Observing Changes to a Pattern with a PatternListener | 53 |
| Maintaining Properties within a Pattern | 53 |
| Loading and Saving Patterns | 54 |
| Transforming Patterns with PatternTransformer | 55 |
| PatternTransformers Included with with JFugue | 56 |
| How to Create a PatternTransformer | 57 |
| How to Use a PatternTransformer | 59 |
| PatternTransformers In Action | 59 |
| Using a ParserListener to Analyze a Pattern | 60 |
| Working with MIDI Patterns | 62 |
| The JFugue Player..... | 63 |
| Playing Music | 63 |
| Starting a Player with a Known Sequencer or Synthesizer | 64 |
| Pausing, Rewinding, and Forwarding the Player | 64 |
| The Streaming Player | 65 |
| How to Simulate a Pipe Organ | 67 |
| Throttling the Delivery of New Fragments | 68 |
| The Anticipator: Know Upcoming Events Before They Happen | 68 |
| Working with MIDI Files..... | 71 |
| Understanding MIDI Support in JFugue | 71 |
| Playing MIDI Files | 72 |
| Creating MIDI Files | 72 |
| Converting MIDI to JFugue MusicStrings | 72 |
| Using JFugue with MIDI Devices..... | 75 |
| Why Communicate with External Devices? | 75 |
| Setting Up Communication with External Devices | 75 |
| JFugue's Device Classes | 76 |
| Using the Intelligent Device Resolver | 76 |
| Sending Music to a MIDI Device | 77 |
| Listening to Music from a MIDI Device | 78 |
| Troubleshooting Your Connections | 79 |
| Rhythms, Intervals, and Microtones..... | 81 |
| Rhythm | 81 |
| Interval Notation | 85 |
| Combining Rhythm and Interval Notation | 86 |
| Microtonal Music | 88 |
| The Architecture of JFugue..... | 89 |
| Parsers and ParserListeners (or Renderers) | 89 |
| JFugue Supports MusicXML | 90 |
| Recombine Parsers and Renderers Endlessly | 90 |
| Creating a New Parser | 90 |
| Creating a New Renderer | 91 |
| Ideas for New Parsers and Renderers | 92 |
| Working with MusicStringParser | 92 |
| Adding a new JFugue Element | 92 |
| Exploring JFugue's "Extras" Package..... | 97 |
| FilePlayer | 97 |
| Midi2JFugue | 98 |

| | |
|--|------------|
| <u>JFugue by Example</u> | 99 |
| The Quintessential Music Program | 99 |
| How to Save Music as a MIDI file | 99 |
| How to Load and Play a MIDI file | 100 |
| How to Save a Pattern | 100 |
| How to Load a Pattern | 100 |
| How to Load a MIDI file and convert it into a JFugue MusicString | 101 |
| How to Combine Patterns | 101 |
| How to Repeat a Pattern | 101 |
| How to Create an Anonymous ParserListener | 101 |
| How to Create Your Own Parser | 102 |
| How to Create Your Own Renderer | 102 |
| How to Connect a Parser to a Renderer | 103 |
| How to Parse MIDI and Render a MusicString | 103 |
| How to Parse a MusicString and Render MIDI | 103 |
| How to Create a Rhythm | 103 |
| How to Use Interval Notation | 104 |
| How to Combine Intervals and Rhythms | 104 |
| How to Use Microtone Notation | 105 |
| How to Send MIDI to an External Device | 106 |
| How to Send a Pattern to an External Device | 106 |
| How to Listen for Music from an External Device | 106 |
| How to Use JFugue with Loaded Soundbanks | 107 |
| <u>Creative Applications with JFugue</u> | 109 |
| JFugue Drum Circle | 109 |
| Lindenmayer System (L-System) Music | 114 |
| <u>Conclusion</u> | 117 |