Getting Started with Java Development in FreeBSD/GhostBSD

By **Ashik Salahudeen**

This is a short introduction to setting up a Java development environment on your computer running FreeBSD 12.0 or other desktop operating systems that derive from it — such as GhostBSD or Trident desktop.

What Is Java Anyway?

Java is a general-purpose, object-oriented programming language developed at Sun Microsystems. It was first released in 1996. Its primary motto was "Write once, run everywhere," which makes it possible for the developers to write Java code, compile it into byte code, and be able to execute it on any operating system that had a Java runtime. So, someone can write code on Windows, but execute it on FreeBSD or GNU/Linux servers. Over time, Java has become extremely popular, and is a very reliable choice for writing server-side applications as well as Android applications.

Java on the FreeBSD Platform

Oracle Corporation bought Sun and is now owner of the official Java runtime and development kit implementations. They provide implementations for Windows, Mac OSX, and Linux operating systems. The official reference implementation is open-sourced under GPLv2 (with a linking exception), and hence it is possible to have OpenJDK implementations for FreeBSD.

The current long-term support version of Java is Java 11, released in September 2018. There are no builds for FreeBSD yet. The previous stable release was Java 8 released in March 2014 and builds for this version exist. The rest of this article is based on Java 8 openjdk.

Getting Started: Install the Java Development Kit

To write applications in Java, you will need a Java Development Kit that provides a compiler, a runtime, and a standard library.

Thankfully, this is available as a pre-built package, and for development purposes, the binary package is sufficient. Install this using the following command (as root):

```
pkg install openjdk8
```

This should not take a long time, and you may verify that the installation is successful by invoking that Java compiler from a terminal.

```
$ javac -version
javac 1.8.0_181
```

Editing Java Code

You can edit Java code using a plain text editor, but because of how the language operates, it gets cumbersome very soon. Almost all Java programmers use an IDE to edit their code. There are several editors available and some of them are commercial. The following is a list of recommended IDEs:

Intellij IDEA Community Edition

IntelliJ is an excellent IDE, available as a binary package and is released under the Apache20 license. Install it using:

```
pkg install intellij
```

The makers of this IDE, Jetbrains, also make a commercial (paid) version called IntelliJ Ultimate, which has more features. For all practical purposes, the community edition should suffice.

Netbeans

Netbeans is another well-built IDE that has been around for a long time. It is also released under the Apache20 license and is available as a pre-built package. Install it using:

```
pkg install netbeans
```

Eclipse

Eclipse is another popular IDE, available under the Eclipse Public License. It is also available as a pre-built package. Install it using:

```
pkg install eclipse
```

Some Generic Configuration to Make Things Easier

In general, setting some environment variables/configurations will make Java development easier. The following is a brief list:

Java Home

This variable specifies the location of your JDK/JRE. It is useful to set this in your environment variable via the .profile file. Add the following line:

```
JAVA_HOME=/usr/local/openjdk8
```

Maven Home

Maven is a popular project management tool for Java projects. It is a pure Java application and is available via the binary packages. The project gets updated often though, so if you want to get the latest Maven binary, it is still best to download it from the project site directly. If you choose to do this, set up the location to Maven's installation directory using the environment variable M2_HOME. This path is the location of "bin" directory in Maven's installation.

```
M2_HOME=/path/to/apache-maven-3.x.0/
```

Intellij – Fix Font Rendering

If the fonts in the IDE look fuzzy, you can fix the problem by passing in the following arguments to the launcher script. Intellij makes it easy via the "Help > Edit Custom VM Options" menu. Paste the following in the text file that opens up. Close the IDE and launch it again.

```
-Dsun.java2d.renderer=sun.java2d.
 marlin.MarlinRenderingEngine
-Dawt.useSystemAAFontSettings=on
-Dswing.aatext=true
-Dsun.java2d.xrender=true
```

Netbeans – Fix Font Rendering

Netbeans' font rendering can be fixed by passing the same arguments, but it does not provide an easy way to do this. As root (or sudo) open "/usr/local/netbeans-8.2/etc/netbeans.conf" in a text editor. If you are on a newer version of Netbeans, just change "8.2" in the above to that version. Find the line that sets the variable "netbeans_default_options" and append the following to the end of the option string. It is one long line.

```
-J-Dswing.aatext=true -J-Dawt.use
SystemAAFontSettings=on -J-Dsun.
java2d.renderer=sun.java2d.marlin.
MarlinRenderingEngine -J-Dsun.java2d.
xrender=true
```

Summary

These things should get you set up for developing Java applications, but this article does not go into detail about writing a Java program or customizing the IDEs or various options you can pass to the JVM when you run it. You can contact me on aashiks@gmail.com if you have queries. •

ASHIK SALAHUDEEN *has been working with computers for about 18 years, mostly with Unix-like operating systems and open source tech. He runs the engineering team at Accordium and works with other FOSS communities during his spare time—primarily the Indic language computing group, Swathanthra Malayalam Computing.*