White Paper

Java 8 SE End of Free Commercial Support

Colin Thomas

Manager of Software Engineering

April 5, 2019

Rev 1: April 25, 2019

ELDER RESEARCH

DATA SCIENCE ◆ MACHINE LEARNING ◆ AI

Table of Contents

Oracle free commercial support is ending for Java 8. Starting with Java 11, Oracle JDK will no longer be free for commercial use. This paper explores the Java ecosystem and alternatives to the use of Oracle JDK.

1.0 Java Overview

Java is a cross-platform computer programming language that is used to run server, desktop, mobile, and web-applications. It uses the class-based object-oriented paradigm in which "objects" are defined by a class and contain data (stored in fields) and references to code (called methods) used to interact with the data. It also supports concurrency, in which multiple computation threads can run at once. Recent versions have added support for functional programming paradigms that focus on treating computations as immutable evaluations of given arguments. Among other benefits, functional paradigms increase the predictability of behavior in concurrent systems, leading to more stable (and easier to maintain) code.

Java code is compiled to bytecode that runs on any Java Virtual Machine (JVM) implementation. The JVM acts as an intermediary that allows the bytecode to be run on any computer architecture without the need for recompiling from source. In addition to Java itself, other programming languages with differing feature sets and programming paradigms have been written or adapted to leverage the power of the JVM. Examples of these include Scala, Groovy, Clojure, and implementations of Python (Jython) and Ruby (JRuby).

Elder Research uses Java extensively for both internal service accelerators and products, such as AARDVARK/RADR, DARWIN, and Risk 360. Most of our products require Java to be installed on the host computer; however, some products (such as DARWIN and NESSUS) include an embedded JVM with the installer.

1.1 Java Platform

Oracle provides a reference JVM implementation called HotSpot, which is licensed under the GNU General Public License (GPL). Together with the Java Class Library (Java's standard library) and various utilities, HotSpot comprises the Java Runtime Environment (JRE), which is required for running any Java (or other JVM language) program. The Java Development Kit (JDK) contains the JRE as well as the compilers and debuggers that are required for developing and deploying Java applications. Oracle provides various JDKs targeted at various domains, such as:

- Java Standard Edition (SE) – the most commonly used version suitable for most PCs and servers
- Java Enterprise Edition (EE) – superset of Java SE with additional APIs for supporting enterprise applications

- Java Micro Edition – contains libraries focused on embedded and mobile devices

1.2 OpenJDK

OpenJDK is a free and open-source implementation of Java SE, maintained by Oracle and a community of developers who have signed the Oracle Contributor Agreement. Since Java 7, it is the official reference implementation of Java SE and is licensed under the GNU General Public License (GPL). As open source software, any entity can distribute builds of OpenJDK for use by end users. To ensure that a build is valid ("Java SE compatible"), it is subjected to a suite of tests called the Technology Compatibility Kit (TCK). Many vendors will provide tweaks such as additional features, performance and security improvements, or utilities. However, changes such as adding a new public method to an API will prevent certification.

1.3 Java 8

Java 8 SE is the current default download version of the JDK. Oracle will continue to offer public updates for personal use until at least the end of 2020; however, the last update for commercial users was released in January 2018. This means that commercial users will no longer receive security patches or other updates except via Oracle's paid My Oracle Support service. Due to the widespread use of Java and its relative complexity, security vulnerabilities that require patching are very common. As such, continued update support is extremely important – and likely required by many of our customers.

1.4 Java 11

Beginning with Java 8, Oracle adopted a new versioning system in which feature releases, with a new version number, would occur every 6 months and Long-Term-Support (LTS) releases would occur every 3 years, starting with Java 11 – which was released in September 2018. Update releases will occur every quarter and will include only security and bug fixes. In the lead up to Java 11's release, Oracle contributed the remaining language features to the OpenJDK Community. Thus, from Java 11 on, Oracle JDK and OpenJDK builds will be "essentially identical" (according to Oracle). A few cosmetic and packaging differences will remain – for example, Oracle JDK will include the Advanced Management Console, will distinguish itself when the `java --version` command is run, will require valid signed cryptographic provider certificates (which OpenJDK does not), and will include an installer (OpenJDK builds are offered as .zip and .tar.gz files).

Oracle will continue to provide Java 11 SE under the GNU General Public License v2 for personal use; however, commercial use will require a commercial license as part of an Oracle product or service offering.

1.4.1 New Features

Java 9 – 11 include a variety of features that will be useful for ERI, including:

- Java Platform Module System: introduced in Java 9, the JPMS allows for more granular control of code package dependencies and seeks to reduce the occurrence of "JAR hell".

- Starting with Java 9, diamond operator can be omitted with anonymous inner classes. Additionally, Interfaces can contain private methods which will allow lengthy default methods to be split into more manageable chunks.

- Local Variable Type Inference: introduced in Java 10, this feature allows variables to be declared as `var` instead of the actual type (the compiler infers the proper type). Project Lombok offers this capability and has been used to good effect in ERI code; however, the inclusion of the feature in core Java will enable easier uptake.

- String updates:

    - Multiline String objects can be streamed intelligently (e.g. removing terminators, ignoring empty line after final terminator, splitting on \r and \n) using String::lines without using split. Additionally, String::lines is lazy and should be more performant than String::split("\R").

    - String objects can now be repeated simply with String::repeat.

    - String::strip (String::stripLeading, String::stripTrailing) offers a more accurate (using Character::isWhitespace) solution for stripping whitespace than String::trim.

- Reading and writing entire file contents is simplified with Files::readString and Files::writeString.

- Simplified conversion of List<T> to T[] without requiring streaming.

- Functional improvements:

    - (Java 9) Includes Optional::stream for easy conversion of Optional objects to a Stream of the contained object.

    - (Java 9) Adds immutable collection types (List, Set, and Map) and concise methods for creating small instances to the Collections API. (e.g. Set.of("One", "Two", "Three"). The new collections also enforce no null elements.

    - (Java 9) Stream now includes takeWhile and dropWhile for supporting functional approaches to while and do-while loops.

    - Provides Optional::isEmpty instead of requiring !opt.isPresent().

- Predicates.not() from Guava has been promoted to the standard Java library as Predicate.not().
- Pattern::asMatchPredicate that allows a regex to match an entire String as a Predicate.

1.4.2 Third Party Support

ERI uses a variety of third-party tools that require Java and may be affected by a transition to version 11.

1.4.2.1 Jenkins

In March, the Jenkins development team announced full support for Java 11 starting with the 2.164 (released on February 10, 2019) and LTS 2.164.1.

1.4.2.2 Maven

In order to use Maven with Java 11, it is recommended that as many plugins are updated as possible. The highest priorities are the compiler, surfire, and failsafe plugins (Figure 1). Additionally, dependencies should be updated to the latest stable versions and additional dependencies should be added to replace any classes that were removed from OpenJDK (for example, jaxb or javax.annotations may be required).

```
<build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.0</version>
        <configuration>
          <argLine>--illegal-access=permit</argLine>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-failsafe-plugin</artifactId>
        <version>2.22.0</version>
        <configuration>
          <argLine>--illegal-access=permit</argLine>
        </configuration>
      </plugin>
    </plugins>
</build>
```

Figure 1: Example maven compiler plugin configuration.

1.4.2.3 Scala

Since the JVM is backward compatible, the Scala language maintainers recommend compiling Scala code using Java 8. The team intends to offer experimental support for running the compiler on LTS versions of the JVM and "to the extent possible will include current LTS versions in [their] CI matrix and community builds" (Scala community blog). Certain Scala releases will increase the minimum supported JVM version to take advantage of new features. The Scala 2.12.x and 2.13.x series require Java 8. A bump to Java 11 is unlikely to occur in the 2.x series.

2.0 Impact and Alternatives

2.1 Java 8 to Java 11 Migration

Java 11 is the culmination of a significant change in the Java ecosystem. In addition to the licensing and cadence changes outlined previously, new paradigms and features have been introduced that will affect the Java 8 to 11 migration process.

2.1.1 Module System

There are several Java Platform Module System pitfalls that must be considered in the migration from Java 8 to 11, such as:

- The module system provides for strong encapsulation that hides non-public (and non-exported) classes from users of the module. In general, best practice is to only depend on public APIs; however, this best practice is be enforced by the compiler.
- Java SE 8 contained Java EE related code (such as the Java API for XML-Based Web Services, Java Architecture for XML Binding, and JavaBeans Activation Framework) that has been moved to modules and will not be seen on the classpath by default. In Java 11, the Java EE modules are removed completely and will need to be replaced with alternatives.
- Packages cannot be shared by two different modules. For non-modularized code, this should not cause an issue as all classes and packages will be added to the unnamed module. However, if a project depends on modularized code that contains the same package, classes from those packages in the non-modularized code will not be loaded.
- Casts to URLClassLoader (such as `(URLClassLoader)getClass().getClassLoader()`) will cause exceptions.
- The Java version string format has changed so may cause issues with code that checks Java version at runtime.

Many of these issues are likely edge cases; however, ERI's code base is large enough that some may be encountered. A particularly importing thing to note is that ERI code does not have to be modularized in order to upgrade to Java 11.

2.1.2 JavaFX

JavaFX, a desktop GUI library, was bundled with Oracle's Java 8 SE even though it was not actually part of the Java SE standard. It was never included in OpenJDK and with the alignment of Oracle JDK with OpenJDK, JavaFX has been moved to its own project, OpenJFX. Any projects that utilize JavaFX will need to be updated to account for this.

2.1.3 General Ecosystem

In general, migration is easiest when the entire ecosystem, from development environments to build tools, is updated at once. Given the size, complexity, and shared use of ERI's code base, wholesale updates are likely infeasible.

2.2 JDK Provider

There are several alternatives to consider with end of free commercial support for Oracle Java 8 SE:

1) Continue with commercial licensing from Oracle.
2) Switch to another commercial subscription.
3) Switch to a free OpenJDK build provider.

2.2.1 Continue to use Oracle Java SE

As of August 3, 2018, Java SE Desktop subscriptions start at $2.50/user and Server subscriptions start at $25/processor. Subscription management will also likely contribute significant additional costs. For some clients that are particularly leery of using open source software, a subscription might be a better option; however, for internal use, the cost outweighs the benefits.

2.2.2 Switch to an OpenJDK Build

OpenJDK will continue to be freely licensed under the GNU GPL license for both commercial and personal use. In addition to the build available directly from the OpenJDK Community, various vendors have offered alternative builds.

2.2.2.1 OpenJDK

The OpenJDK community website (https://jdk.java.net) offers binaries for Windows, macOS, and Linux via zip/tarball downloads (i.e. with no installer). These builds will likely match the Oracle JDK build most closely (including the 6 month release cadence), but may be more daunting to install. For each LTS version, it appears likely that there will be at least

four years of public updates stewarded by a particular organization (likely to be Red Hat for Java 11).

2.2.2.2 AdoptOpenJDK

AdoptOpenJDK is backed by a variety of major players, such as Amazon Azul, IBM, Red Hat, GoDaddy, and others. Binaries containing either the HotSpot or Eclipse OpenJ9 JVM are available for most platforms (https://adoptopenjdk.net/). Java 8 will be supported until at least September 2023 and Java 11 will be supported until at least September 2022. Of note is that AdoptOpenJDK's binaries (both those containing HotSpot and OpenJ9) pass the TCK tests; however, the community has not been able to reach an agreement with Oracle to run the TCK certification on OpenJ9 builds. The actual reasons for this are not clear, but there is suspicion in the community that this relates to OpenJ9's distribution under the Apache 2/Eclipse license (as opposed to the GNU GPL).

2.2.2.3 Azul Zulu

Zulu is offered by Azul Systems and is TCK certified and is free to download and use. Azul also has a commercial offering called Zulu Enterprise, which is licensed on a system basis ($13,200/year for up to 25 systems as of March 2019). For Zulu Enterprise, Azul offers an additional year of commercial support (beyond that offered by Oracle) for LTS Java releases. Additionally, they will offer Medium-Term Support (MTS) on every second non-LTS version of Java (e.g. Java 9, Java 13, etc.). This will allow switching to newer versions of Java while still enjoying support for longer than the 6-month version cadence.

2.2.2.4 Amazon – Corretto

Corretto is Amazon's build of OpenJDK that includes patches that Amazon has developed for running their own services. It is TCK certified and will include long-term support from Amazon. Corretto 8 will be supported until 2023 and Corretto 11 (released March 15, 2019) will be supported until 2024. Installation packages for Windows, macOS, and most of the major Linux distributions are available (including an official Docker image available from Docker Hub). Corretto is licensed under the GNU GPL and is free of charge.

3.0 Recommendations

Given the widespread use of Oracle Java 8 SE, commercial organizations such as ERI need to quickly decide how they will approach the change in Oracle's JDK licensing.

3.1 Recommendations for Developers

Given the extensive use of AWS and Amazon's general market presence, a switch to Amazon Corretto for development is recommended. Due to Corretto 8's support until 2023, an immediate switch to Java 11 is not necessary; however, transition plans should be decided upon by the end of 2019. Since version 11 is the next LTS, interim transitions to

versions 9 and 10 are not recommended; additionally, as modularization is not required for use of Java 11, conversion to modules should be avoided until any issues are resolved. Transition of any project should be delayed until its dependent libraries have been upgraded. At ERI, a possible scenario would be for the Software Architect and 1-2 other developers to have a day long Hack-a-thon in which those libraries and projects are tested on Java 11. Project managers for client-controlled code bases will work with their clients to come up with an appropriate transition plan/date.

3.2 Recommendations for Non-Developer Users

With the release of the first Java 8 update using the new license on April 16, 2019 (Update 211), all users using Java in any commercial context (this is likely everyone reading this whitepaper) should immediately uninstall Oracle Java and switch to one of the OpenJDK builds discussed above. Amazon Corretto 8 is recommended; however, any of the alternatives can be used without issue.

On Windows based systems, uninstalling all Oracle Java builds should be straightforward using the standard "Add/Remove Programs" utility.

On MacOS, discovering and removing all Oracle JVMs is less obvious. As discussed in 1.1, Oracle's JVM is branded HotSpot; as such, "HotSpot" will be contained in the output of the `java -version` terminal call (Figure 2). A bash snippet for finding and removing all Oracle Java distributions can be found in Figure 3.

```
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
```

Figure 2: Example `java -version` output.

```
#!/bin/bash

for j in $(ls /Library/Java/JavaVirtualMachines);
do
    if /Library/Java/JavaVirtualMachines/$j/Contents/Home/bin/java -version 2>&1 | grep -q
'HotSpot'; then
        echo $j' is an Oracle Java distribution and will be removed.';
        sudo mv /Library/Java/JavaVirtualMachines/$j/ ~/.Trash ;
    else
        echo $j' is not an Oracle Java HotSpot distribution and will be kept.';
    fi
done; \
echo 'All done.';
```

Figure 3: Bash snippet for removing Oracle JDK distributions (developed by William Proffitt). Note: no implied warranty – use at your own risk.

3.3 Recommendations for Client Systems

For client systems on which it has administrative rights or advisory influence, system owners should be notified of the change in licensing and be given an updated recommendation. Other clients should be notified of the change as a courtesy.

4.0 References

https://en.wikipedia.org/wiki/Java_(programming_language)

https://en.wikipedia.org/wiki/Java_(software_platform)

https://en.wikipedia.org/wiki/Java_version_history

https://en.wikipedia.org/wiki/Java_virtual_machine

https://en.wikipedia.org/wiki/List_of_Java_virtual_machines

https://en.wikipedia.org/wiki/Comparison_of_Java_virtual_machines

https://en.wikipedia.org/wiki/Java_Class_Library

https://en.wikipedia.org/wiki/Java_Classloader

https://en.wikipedia.org/wiki/Java_Development_Kit

https://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition

https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

https://en.wikipedia.org/wiki/Java_Platform_Module_System

https://en.wikipedia.org/wiki/HotSpot

https://en.wikipedia.org/wiki/OpenJDK

https://www.oracle.com/technetwork/java/javase/terms/license/index.html

https://www.oracle.com/assets/java-se-subscription-pricelist-5028356.pdf

https://www.oracle.com/technetwork/java/java-se-support-roadmap.html

https://www.oracle.com/technetwork/java/javaseproducts/overview/javasesubscriptionfaq-4891443.html

https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-1.html

https://docs.oracle.com/en/java/javase/11/migrate/index.html

https://blogs.oracle.com/java-platform-group/oracle-jdk-releases-for-java-11-and-later

https://blogs.oracle.com/java-platform-group/oracle-java-se-releases-faq

https://blogs.oracle.com/java-platform-group/update-and-faq-on-the-java-se-release-cadence

https://www.java.com/en/download/faq/distribution.xml

https://aws.amazon.com/corretto/

https://aws.amazon.com/corretto/faqs/

https://aws.amazon.com/blogs/opensource/amazon-corretto-no-cost-distribution-openjdk-long-term-support/

https://www.azul.com/downloads/zulu/

https://www.azul.com/products/zulu-enterprise/

https://www.azul.com/products/azul_support_roadmap/

https://adoptopenjdk.net/

https://adoptopenjdk.net/quality.html

https://adoptopenjdk.net/support.html

https://adoptopenjdk.net/about.html

https://blog.adoptopenjdk.net/

https://openjdk.java.net/faq/

https://openjdk.java.net/contribute/

http://openjdk.java.net/jeps/286

https://jenkins.io/blog/2019/03/11/let-s-celebrate-java-11-support/

https://docs.scala-lang.org/overviews/jdk-compatibility/overview.html

https://www.eclipse.org/openj9/

https://blog.codefx.org/java/java-9-migration-guide/

https://blog.codefx.org/java/java-11-migration-guide/

https://blog.codefx.org/java/java-module-system-tutorial/

https://blog.codefx.org/java/java-11-gems/

https://www.baeldung.com/jvm-languages

https://www.baeldung.com/oracle-jdk-vs-openjdk

https://www.baeldung.com/project-jigsaw-java-modularity

https://www.baeldung.com/new-java-9

https://www.baeldung.com/java-9-stream-api

https://blog.joda.org/2018/09/do-not-fall-into-oracles-java-11-trap.html

https://blog.joda.org/2018/09/time-to-look-beyond-oracles-jdk.html

https://blog.joda.org/2018/08/java-is-still-available-at-zero-cost.html

https://blog.joda.org/2018/09/from-java-8-to-java-11.html

https://dev.karakun.com/java/2018/06/25/java-releases.html

https://winterbe.com/posts/2018/08/29/migrate-maven-projects-to-java-11-jigsaw/

https://jaxenter.com/oracle-jdk-builds-openjdk-builds-difference-149318.html

https://jaxenter.com/jdk-11-java-ee-modules-140674.html

https://medium.com/@javachampions/java-is-still-free-2-0-0-6b9aa8d6d244

https://medium.com/codefx-weekly/no-free-java-lts-version-b850192745fb

https://blog.devexperts.com/oracle-jdk-vs-openjdk-builds-comparison/

https://dzone.com/articles/immutable-collections-in-java-9

https://www.geeksforgeeks.org/differences-jdk-jre-jvm/

https://aboullaite.me/10-new-features-in-java-10/

https://www.reddit.com/r/java/comments/9odebq/the_openjdk_community_tck_license_agreement_octla/

www.elderresearch.com

ELDER RESEARCH

DATA SCIENCE • MACHINE LEARNING • AI

Headquarters
300 W. Main Street
Suite 301
Charlottesville, VA 22903

434.973.7673

Arlington, VA Office
2101 Wilson Boulevard
Suite 900
Arlington, VA 22201

855.973.7673

Linthicum, MD Office
839 Elkridge Landing
Suite 215
Linthicum, MD 21090

855.973.7673

Raleigh, NC Office
14 E. Peace
Suite 302
Raleigh, NC 27604

855.973.7673