Migrating Beyond Java 8

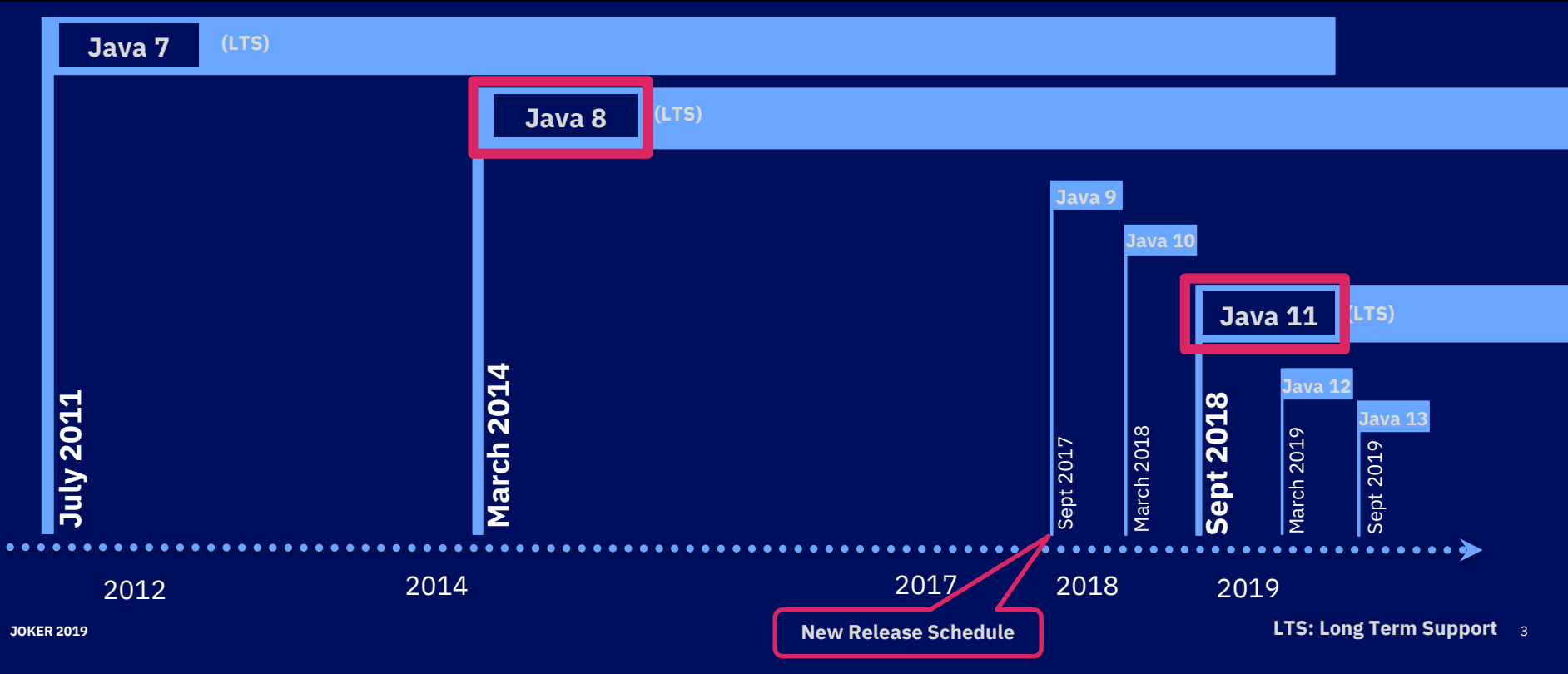
Identifying missing APIs and understanding new behavior

Dalia Abo Sheasha
WebSphere Migration Tools Development Lead
Email: dalia@us.ibm.com
Twitter: @DaliaShea

Agenda

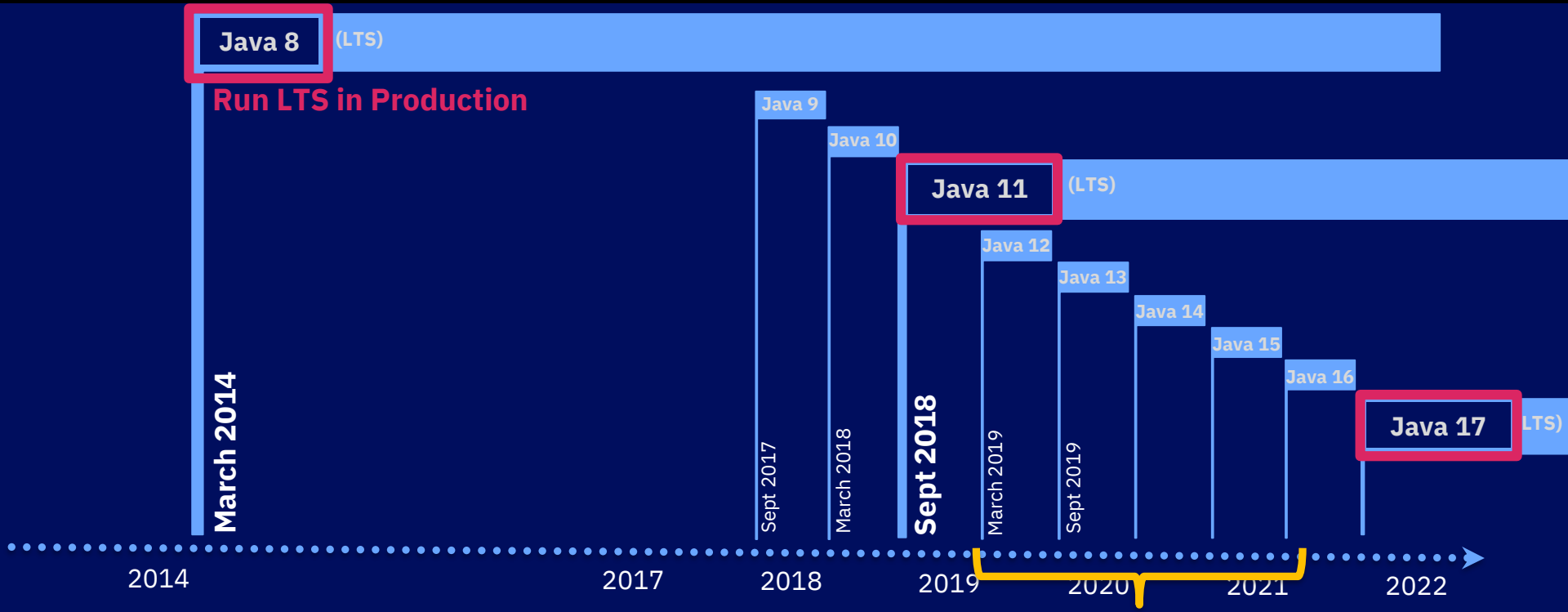
- Background
- Java 9..13: General Migration Concerns
- Java 8 -> Java 11: Breaking Changes
- Java 11 -> Java 13: Breaking Changes
- Migration Tools
  - Application Binary Scanner Tool
  - Application Source Scanner Tool
  - Jdeps

Java Timeline (Current)



LTS: Long Term Support

Java Timeline (Future)



Java 9..13: New Features

JDK 9

The goal of this Project was to produce an open-source refe
of the Java SE 9 Platform as defined by JSR 379 in the Java C

JDK 9 reached General Availability on 21 September 2017. P
binaries under the GPL are available from Oracle; binaries fr
follow shortly.

The features and schedule of this release were proposed an
Process, as amended by the JEP 2.0 proposal.

Features

102: Process API Updates
110: HTTP 2 Client
143: Improve Contended Locking
158: Unified JVM Logging
165: Compiler Control
193: Variable Handles
197: Segmented Code Cache
199: Smart Java Compilation, Phase Two
200: The Modular JDK
201: Modular Source Code
211: Elide Deprecation Warnings on Import Statements
212: Resolve Lint and Doclint Warnings
213: Milling Project Coin
214: Remove GC Combinations Deprecated in JDK 8
215: Tiered Attribution for javac
216: Process Import Statements Correctly
217: Annotations Pipeline 2.0
219: Datagram Transport Layer Security (DTLS)
220: Modular Run-Time Images
221: Simplified Doclet API
222: jshell: The Java Shell (Read-Eval-Print Loop)
223: New Version-String Scheme
224: HTML5 Javadoc
225: Javadoc Search
226: UTF-8 Property Files
227: Unicode 7.0
228: Add More Diagnostic Commands
229: Create PKCS12 Keystores by Default
231: Remove Launch-Time JRE Version Selection
232: Improve Secure Application Performance
233: Generate Run-Time Compiler Tests Automatically
235: Test Class-File Attributes Generated by javac
236: Parser API for Nashorn
237: Linux/AArch64 Port
238: Multi-Release JAR Files
240: Remove the JVM TI hprof Agent

JDK 10

JDK 10 is the open-source reference implementation of the Java SE 10 Platform as defined by JSR 383 in the Java Community Process.

JDK 10 reached General Availability on 20 March 2018. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal.

Features

286: Local-Variable Type Inference
296: Consolidate the JDK Forest into a Single Repository
304: Garbage-Collector Int
307: Parallel Full GC for G1
310: Application Class-Dat
312: Thread-Local Handsh
313: Remove the Native-H
314: Additional Unicode La
316: Heap Allocation on Al
317: Experimental Java-Ba
319: Root Certificates
322: Time-Based Release V

JDK 11

JDK 11 is the open-source reference implementation of version 11 of the Java SE 11 Platform as specified by by JSR 384 in the Java Community Process.

JDK 11 reached General Availability on 25 September 2018. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal. The release was produced using the JDK Release Process (JEP 3).

Features

181: Nest-Based Access Control
309: Dynamic Class-File Constants
315: Improve Aarch64 Intrinsics
318: Epsilon: A No-Op Garbage Collector
320: Remove the Java EE and CORBA Modules
321: HTTP Client (Standard)
323: Local-Variable Syntax for Lambda Parameters
324: Key Agreement with Curve25519 and Curve448
327: Unicode 10
328: Flight Recorder
329: ChaCha20 and Poly1305 Cryptographic Algorithms
330: Launch Single-File Source-Code Programs
331: Low-Overhead Heap Profiling
332: Transport Layer Security (TLS) 1.3
333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
335: Deprecate the Nashorn JavaScript Engine
336: Deprecate the Pack200 Tools and API

JDK 12

JDK 12 is the open-source reference implementation
Platform as specified by by JSR 386 in the Java Comm

JDK 12 reached General Availability on 19 March 201
under the GPL are available from Oracle; binaries fr
shortly.

The features and schedule of this release were prop
Process, as amended by the JEP 2.0 proposal. The re
JDK Release Process (JEP 3).

Features

189: Shenandoah: A Low-Pause-Time Garbage C
230: Microbenchmark Suite
325: Switch Expressions (Preview)
334: JVM Constants API
340: One AArch64 Port, Not Two
341: Default CDS Archives
344: Abortable Mixed Collections for G1
346: Promptly Return Unused Committed Memory from G1

JDK 13

JDK 13 is the open-source reference implementation of version 13 of the Java SE Platform as specified by by JSR 388 in the Java Community Process.

JDK 13 reached General Availability on 17 September 2019. Production-ready binaries under the GPL are available from Oracle; binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the JEP Process, as amended by the JEP 2.0 proposal. The release was produced using the JDK Release Process (JEP 3).

Features

350: Dynamic CDS Archives
351: ZGC: Uncommit Unused Memory
353: Reimplement the Legacy Socket API
354: Switch Expressions (Preview)
355: Text Blocks (Preview)

Schedule

2019/06/13	Rampdown Phase One (fork from main line)
2019/07/18	Rampdown Phase Two
2019/08/08	Initial Release Candidate
2019/08/22	Final Release Candidate
2019/09/17	General Availability

Tons of Features!

- (JDK 9, JDK 10, JDK 11, JDK 12, JDK 13)
  - var for local variables (JEP 286)
  - Launch Single-File Source-Code Programs (JEP 330)
  - New methods in several classes (Optional, Collection, etc)
  - TLS 1.3
  - ...

Java 9..13: Modularity

- Java Platform Module System (JPMS)
  - Breaks code into modules containing packages
  - Applications must declare dependency on modules for access
  - Disruptive for Java SE application migrations
- “Kill switch” on by default (currently)
  - --illegal-access=permit

```
--illegal-access=<value>
                  permit or deny access to members of types in named modules
                  by code in unnamed modules.
                  <value> is one of "deny", "permit", "warn", or "debug"
                  This option will be removed in a future release.
```

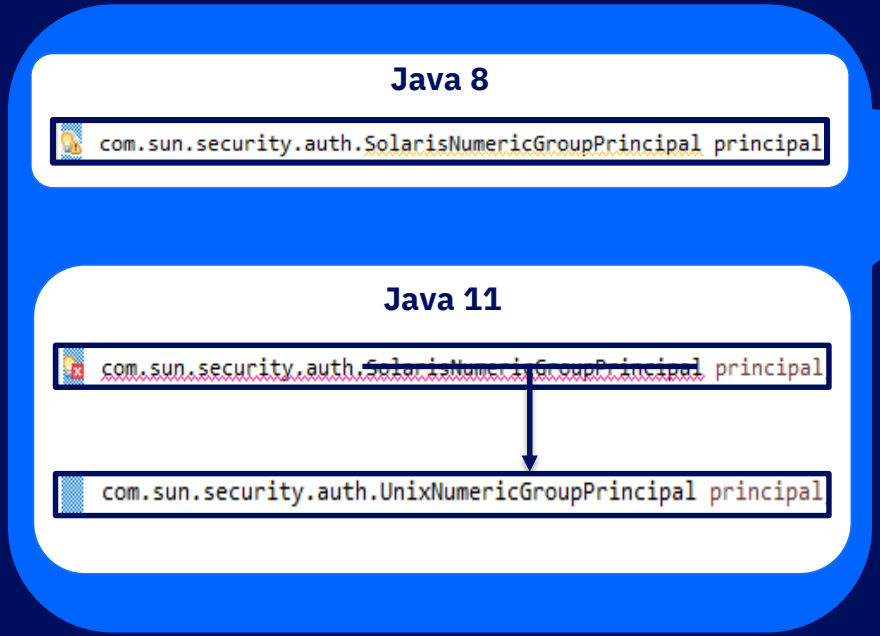
- **My Advice:** when first migrating to Java 11+, keep the “kill switch” on

Java 9..13: General Migration Concerns

- Missing packages/classes/methods
  - Migration Strategy is based on APIs
    - Package missing dependencies in your application
    - Rely on application servers to provide missing APIs
    - Change application code to call replacement APIs
- Minor behavior changes that come with Java updates
- Application **dependencies** may need to be updated to Java 11+ compatible versions

Java 8 -> Java 11

Java 8 -> Java 11: Removed Packages/Classes

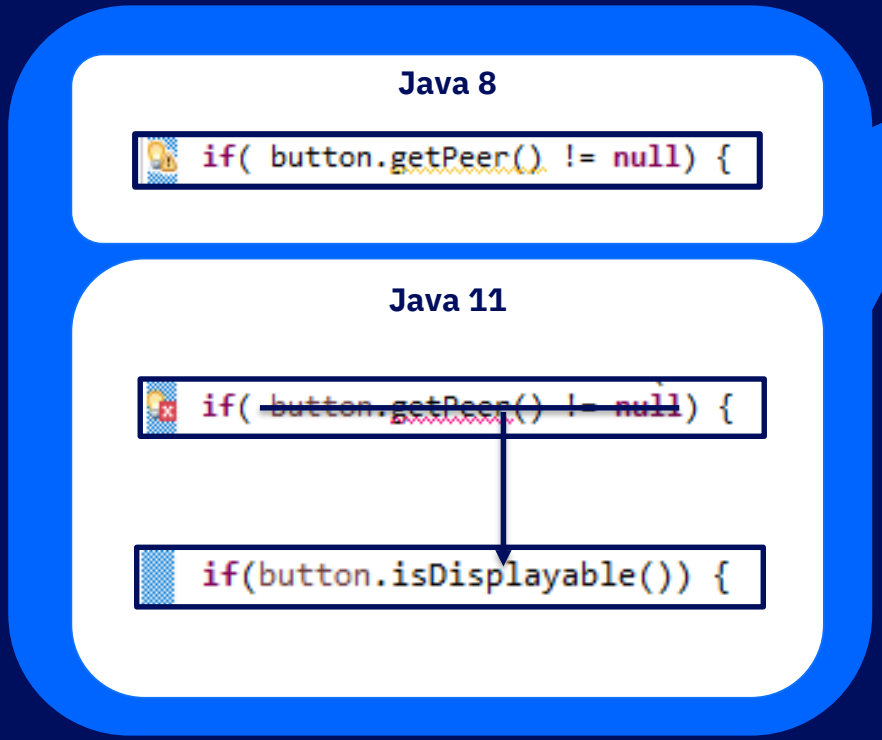


Removed Security classes:

- com.sun.security.auth.PolicyFile
- com.sun.security.auth.SolarisNumericGroupPrincipal
- com.sun.security.auth.SolarisNumericUserPrincipal
- com.sun.security.auth.SolarisPrincipal
- com.sun.security.auth.X500Principal
- com.sun.security.auth.callback.DialogCallbackHandler
- com.sun.security.auth.module.SolarisLoginModule
- com.sun.security.auth.module.SolarisSystem
- javax.security.auth.Policy

(Most classes Deprecated in Java 1.4!)

Java 8 -> Java 11: Removed Packages/Classes



Removed AWT APIs:

- java.awt.peer.*
- java.awt.dnd.peer.*
- com.sun.awt.AWTUtilities

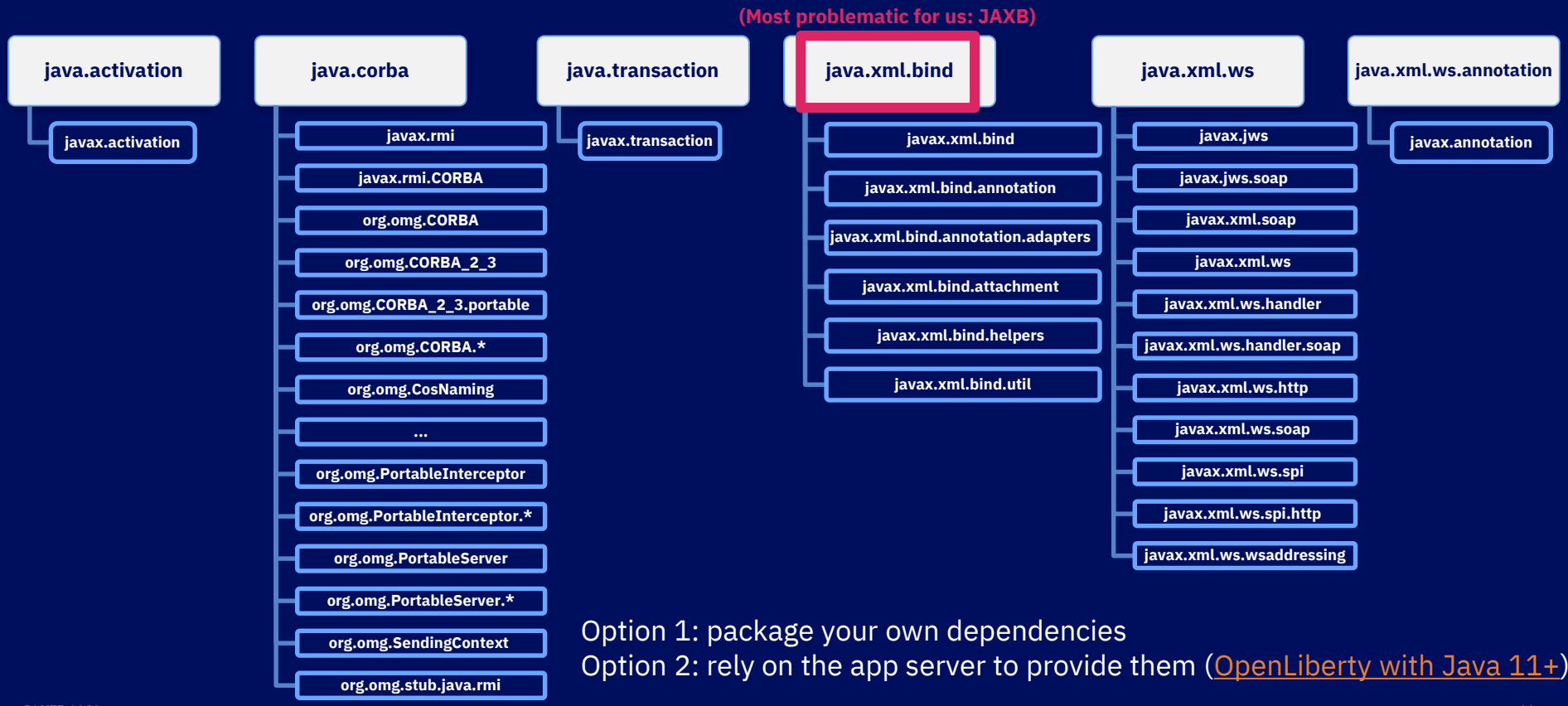
Removed DOM APIs:

- com.sun.java.browser.plugin2.DOM
- sun.plugin.dom.DOMObject

Removed Misc:

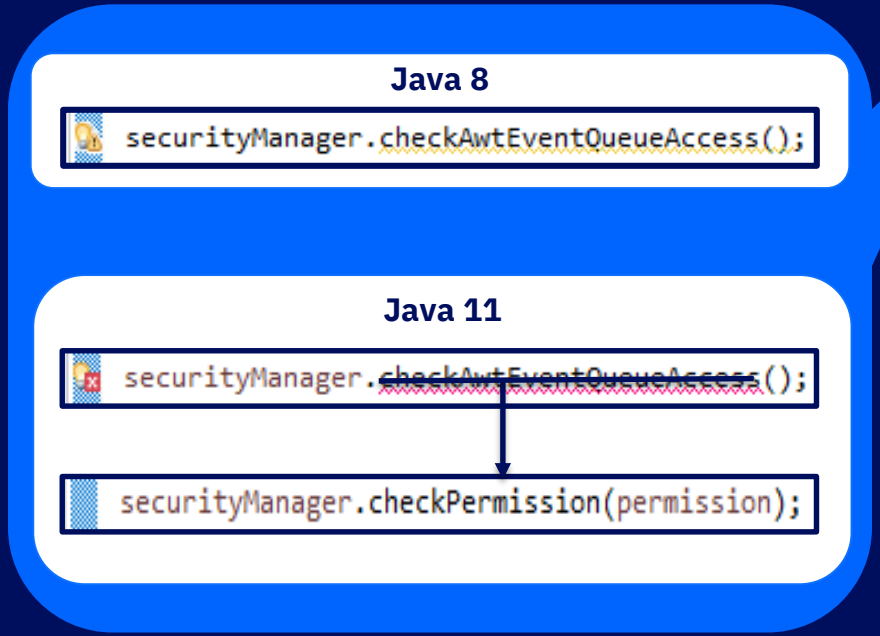
- com.sun.image.codec.jpeg.* (Replace with Java Image I/O)
- com.sun.xml.internal.bind.*

Java 8 -> Java 11: Removed Java EE Modules



Option 1: package your own dependencies
Option 2: rely on the app server to provide them (OpenLiberty with Java 11+)

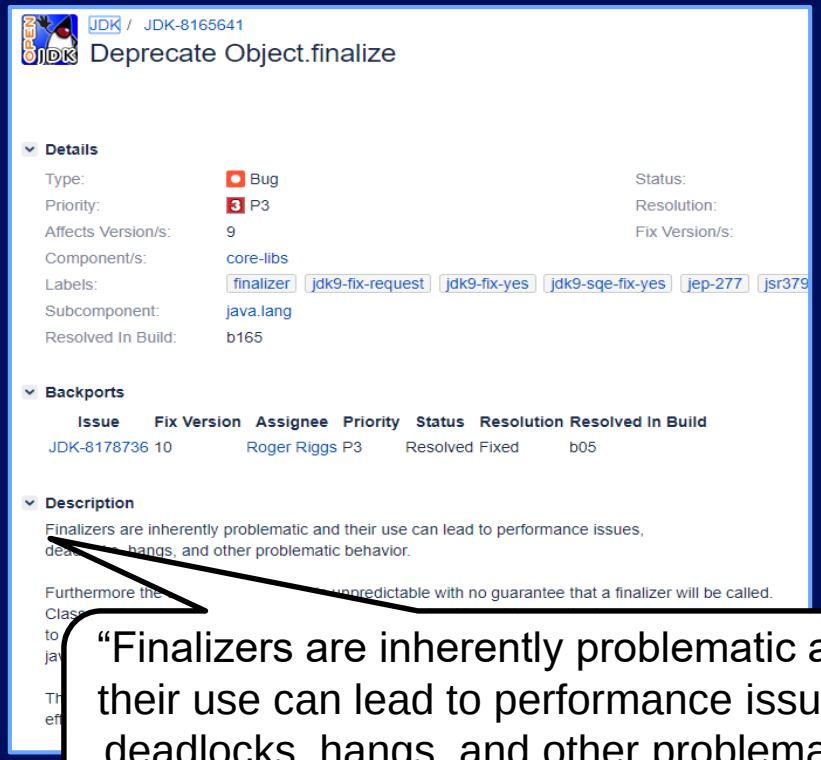
Java 8 -> Java 11: Removed Methods



Removed java.lang.SecurityManager Methods:

- java.lang.SecurityManager.checkAwtEventQueueAccess()
- java.lang.SecurityManager.checkSystemClipboardAccess()
- java.lang.SecurityManager.checkMemberAccess(..)
- java.lang.SecurityManager.checkTopLevelWindow(..)
- java.lang.SecurityManager.classDepth(..)
- java.lang.SecurityManager.classLoaderDepth()
- java.lang.SecurityManager.currentClassLoader()
- java.lang.SecurityManager.currentLoadedClass()
- java.lang.SecurityManager.getInCheck()
- java.lang.SecurityManager.inClass(..)
- java.lang.SecurityManager.inClassLoader()

Java 8 -> Java 11: Removed Methods



"Finalizers are inherently problematic and their use can lead to performance issues, deadlocks, hangs, and other problematic behavior."

Removed java.lang.Thread Methods:

- java.lang.Thread.destroy()
- java.lang.Thread.stop(java.lang.Throwable)

Removed java.lang.Runtime Methods:

- java.lang.Runtime.getLocalizedInputStream(..)
- java.lang.Runtime.getLocalizedOutputStream(..)
- java.lang.Runtime.runFinalizersOnExit(..)

Removed java.lang.System Method:

- java.lang.System.runFinalizersOnExit(..)

Java 8 -> Java 11: Removed Methods

Removed java.util.logging.LogManager Methods:

- java.util.logging.LogManager.addPropertyChangeListener(..)
- java.util.logging.LogManager.removePropertyChangeListener(..)

Removed java.util.jar.Pack200.Packer/Unpacker Methods:

- java.util.jar.Pack200.Packer.addPropertyChangeListener(..)
- java.util.jar.Pack200.Packer.removePropertyChangeListener(..)
- java.util.jar.Pack200.Unpacker.addPropertyChangeListener(..)
- java.util.jar.Pack200.Unpacker.removePropertyChangeListener(..)

Java 8 -> Java 11: Other Changes

Technology Removal:

- Java Web Start (openwebstart.com)
- JavaFX (openjfx.io)

Other Removals:

- JDK-internal APIs (sun.misc.BASE64Encoder/Decoder)
- @jdk.Exported
- netdoc protocol handler

Behavior changes:

- java.lang.Class.getAnnotation()
- java.nio.channels.DatagramChannel.send()

And more...

- Oracle JDK 11 Migration Guide

Java 11 -> Java 13

Java 11 -> Java 12

```
// Use try-with-resources to cleanup your resources
try(FileInputStream fis = new FileInputStream(file)) {
     // Use your resource....
} catch(IOException e) {
     e.printStackTrace();
}
// Resources will be cleaned up when try block finishes execution
```

- Removed Methods:
  - java.io.FileInputStream.finalize()
  - java.io.FileOutputStream.finalize()
  - java.util.zip.ZipFile.finalize()
  - java.util.zip.Inflater.finalize()
  - java.util.zip.Deflater.finalize()
- Removed Class:
  - com.sun.awt.SecurityWarning
- Behavior changes
  - user.timezone system property
  - java.util.Properties.loadFromXML(java.io.InputStream)

Migration Guide: Oracle JDK 12 Migration Guide

Java 12 -> Java 13: Changes

- Removed java.lang.Runtime Methods:
  - java.lang.Runtime.traceInstructions(boolean)
  - java.lang.Runtime.traceMethodCalls(boolean)
- Removed System Property:
  - awt.toolkit system property

Migration Guide: Oracle JDK 13 Migration Guide

Java 8 -> Java 11+
Where do I start?

Application Binary Scanner Tool

- Download: http://ibm.biz/WAMT4AppBinaries
  - Quarterly release schedule
- Scans an application for potential migration issues for a variety of scenarios
  - **Java SE versions**
    - **Oracle/IBM Java 5 → Java 11+**
  - On-premise → Cloud
  - WebSphere traditional → Liberty
  - Others…
- Produces a migration report describing any issues found in the application, solutions and resource links

Application Binary Scanner Tool

- Command line tool (binaryAppScanner.jar)
  - Run the tool with Java 6 or later
- Specify the source and target environment using command line options
  - Java SE migration options

    --analyzeJavaSE

    --sourceJava=[ibm5|oracle5|ibm6|oracle6|ibm7|oracle7|**ibm8|oracle8**|java11|java12]

    --targetJava=[ibm7|oracle7|ibm8|oracle8|**java11|java12**]
  - Java SE help command: java -jar binaryAppScanner.jar --help --analyzeJavaSE
- Accepts .ear/.war files (and .jar files with --analyzeJavaSE option)
  - Uses ASM to scan .class files for problematic package, class and methods
  - No application server required

Application Binary Scanner Tool

- Excludes third-party packages by default
  - Packages excluded by default: com.ibm, com.informix, com.mchange, com.microsoft, com.sybase, com.sun, java, javax, net, oracle, org,sqlj, _ibmjsp
  - Use the --includePackages or --excludePackages options to override the default scanned packages.
- Produces an HTML report by default. JSON report when --format=json is specified.
- Explore other command options using: java -jar binaryAppScanner.jar --help
- Documentation

Application Binary Scanner Tool Supported Languages

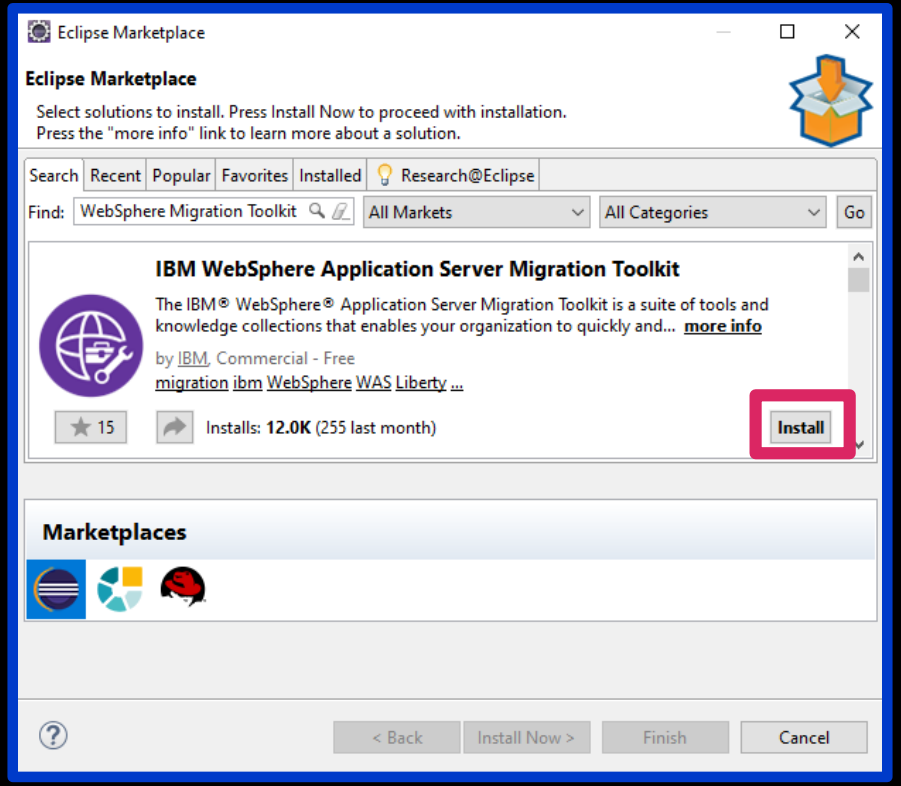
- Chinese - Simplified
  -Duser.language=zh -Duser.country=CN
- Chinese - Traditional
  -Duser.language=zh -Duser.country=TW
- Czech
  -Duser.language=cs
- English (default)
  -Duser.language=en
- Czech
  -Duser.language=cs
- French
  -Duser.language=fr
- German
  -Duser.language=de
- Hungarian
  -Duser.language=hu
- Italian
  -Duser.language=it
- Japanese
  -Duser.language=ja
- Korean
  -Duser.language=ko
- Polish
  -Duser.language=pl
- Portuguese - Brazillian
  -Duser.language=pt -Duser.country=BR
- Romanian
  -Duser.language=ro
- **Russian**
  -Duser.language=ru
- Spanish
  -Duser.language=es

Demo: Application Binary Scanner Tool

Demo: Application Binary Scanner Tool (Russian)

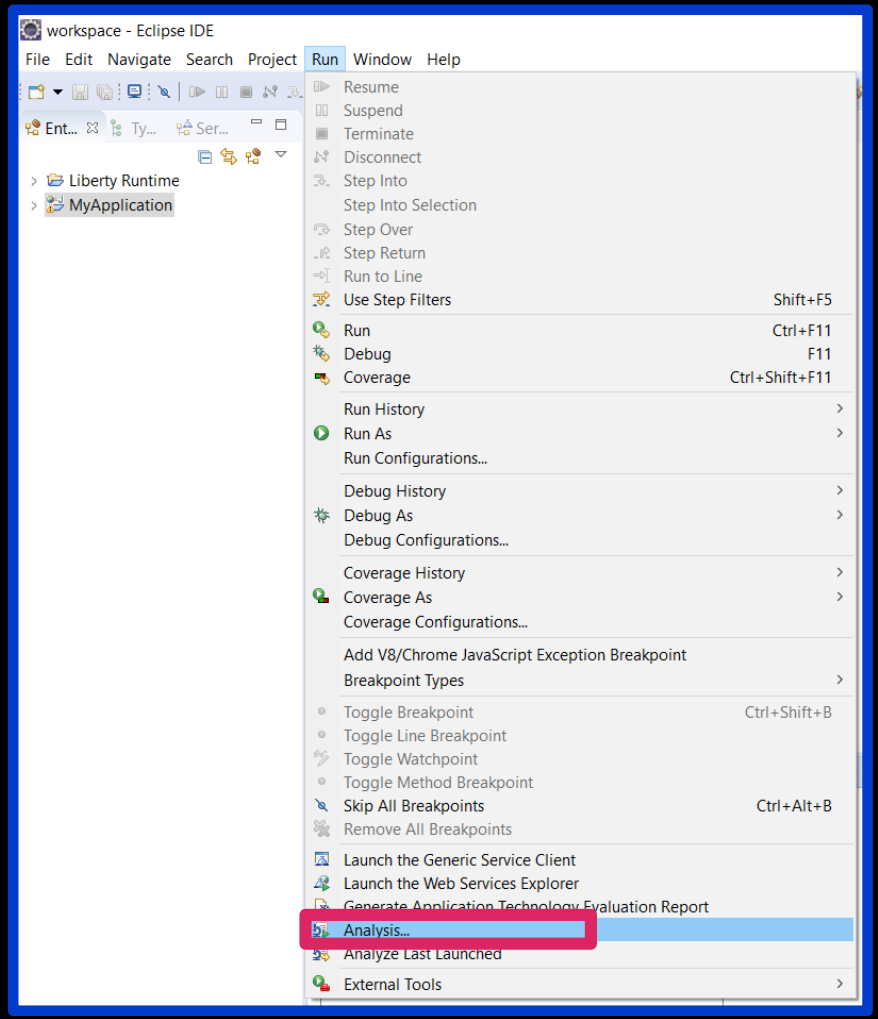
Application Source Scanner Tool (Eclipse Plugin)

- Download site: http://ibm.biz/WAMT4Eclipse
  - Eclipse Marketplace
  - Quarterly release schedule
  - Documentation (See Chapter 3-4 for installation details)
- Install in Eclipse Oxygen (4.7) and later
- Scans an application **source code** for potential migration issues
  - **Oracle/IBM Java 6 → Java 11+**



Application Source Scanner Tool (Eclipse Plugin)

- After installing Eclipse Plugin, a new Analysis... option is added under the Run menu
- Select projects to scan from a list of imported Eclipse projects.
- Specify the source and target environment from a drop-down list in Eclipse
- A migration issues list is generated after analysis
  - Click-and-go to affected code line
  - Preview help on migration issues within Eclipse



Demo: Application Source Scanner Tool

Java class dependency analyzer (Jdeps)

- Command utility shipped with JDK
- Various Options
  - Migration Relevant Option: -jdkinternals
  - Accepts .jars, .class files, directories
- Binary scanner and source scanner tools will recommend jdeps if applicable
- Limitations
  - Flags only a subset of removed APIs
  - No method removal flagging
  - Java version dependent (i.e. running Jdeps shipped with Java 8 is different than Java 11)
- Documentation

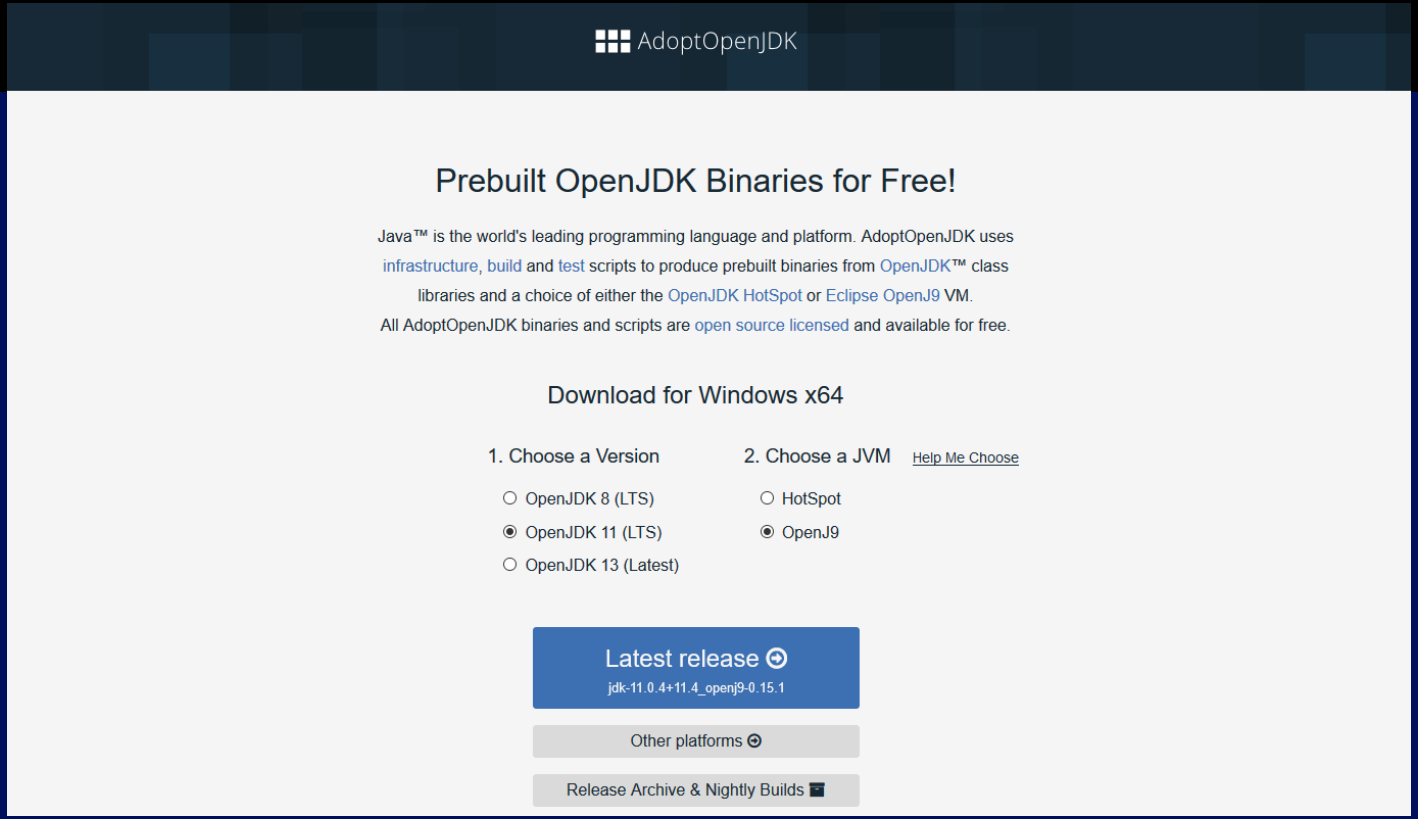
What should I do after I leave this session?

1. Download the Application Binary scanner: http://ibm.biz/WAMT4AppBinaries
2. Follow the prompts to install the jar.
3. Run the binary scanner against your application binaries. Here is my command:

   java -jar binaryAppScanner.jar C:\Apps\MyApplication.war --analyzeJavaSE --sourceJava=ibm8 --targetJava=java11

   - Command help:
     - **java -jar binaryAppScanner.jar --help --analyzeJavaSE**
     - **java -jar binaryAppScanner.jar --help**
4. Review the report and read the help for each flagged issue to resolve them.
5. After resolving all issues identified, run your application with Java 11+.
6. (optional) Let me know if the tools helped….or didn't! (@DaliaShea)

Looking for a free Java 11 build? Checkout AdoptOpenJDK!



https://adoptopenjdk.net/

References/Resources

- My Blog on adding the Java 11+ feature to the migration tools (written instructions on running binary scanner): https://developer.ibm.com/tutorials/migration-to-java-11-made-easy/
- Video of how to migrate to Java 11 using the source scanner: https://youtu.be/m-l9eu4AAq4
- Binary Scanner (full link): https://developer.ibm.com/wasdev/downloads/#asset/tools-Migration_Toolkit_for_Application_Binaries
  - Tool documentation and videos - see the « additional information » section
- Source Scanner (full link): https://developer.ibm.com/wasdev/downloads/#asset/tools-WebSphere_Application_Server_Migration_Toolkit
  - Tool documentation and videos - see the « additional information » section
- Oracle Java SE Support Roadmap: https://www.oracle.com/technetwork/java/java-se-support-roadmap.html
- Migrating from Oracle Java to AdoptOpenJDK: https://adoptopenjdk.net/MigratingtoAdoptOpenJDKfromOracleJava.pdf
- Andy Guibert's blog on OpenLiberty Java 11+ support: https://openliberty.io/blog/2019/02/06/java-11.html

Questions?

Dalia Abo Sheasha
Email: dalia@us.ibm.com
Twitter: @DaliaShea