MAKING THE MOVE TO JAVA 17

ALEX MOTLEY
@alexsmotley

CINDY HIGH
@CTHigh

ABOUT US

Alex Motley

Alex is a software engineer at IBM leading the WebSphere Application Server migration tools team. His team develops several migration tools to help with a variety of modernization scenarios including WebSphere traditional to Liberty, Java SE and EE versions, and on-premise to cloud. He has almost a decade of experience working with Java EE and Java application server technology. Alex enjoys hiking and skiing and lives in Rochester, MN.

Cindy High

Cindy is a software engineer with IBM and is the WebSphere Application Server migration tools architect focusing on application modernization and configuration migration. She has worked over 20 years with Java EE and application server technology. The tools we developed over the years for WebSphere can help you migrate your applications easier! Cindy enjoys biking and working on stained glass projects and lives in Raleigh, NC.

MAKING THE MOVE TO JAVA 17

WHY MIGRATE?

WHY MIGRATE?

Lots of new features!

Feature Lists (JDK 11, JDK 12, JDK 13,
JDK 14, JDK 15, JDK 16, JDK 17)

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 Algorith
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 of version 12 of the Java SE Platform as specified by by JSR 386 in the Java Community Process.

JDK 12 reached General Availability on 19 March 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

189: Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)
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)

JDK 14

JDK 14 is the open-source reference implementation of versio
Platform as specified by by JSR 389 in the Java Community Pr

JDK 14 reached General Availability on 17 March 2020. Produc
under the GPL are available from Oracle; binaries from other
shortly.

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

Features

305: Pattern Matching for instanceof (Preview)
343: Packaging Tool (Incubator)
345: NUMA-Aware Memory Allocation for G1
349: JFR Event Streaming
352: Non-Volatile Mapped Byte Buffers
358: Helpful NullPointerExceptions
359: Records (Preview)
361: Switch Expressions (Standard)
362: Deprecate the Solaris and SPARC Ports
363: Remove the Concurrent Mark Sweep (CMS) Garbage Collector
364: ZGC on macOS
365: ZGC on Windows
366: Deprecate the ParallelScavenge + SerialOld GC Combination
367: Remove the Pack200 Tools and API
368: Text Blocks (Second Preview)
370: Foreign-Memory Access API (Incubator)

JDK 15

JDK 15 is the open-source reference implementation of version 15 of the Java SE Platform, as specified by by JSR 390 in the Java Community Process.

JDK 15 reached General Availability on 15 September 2020. 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 tracke
Process, as amended by the JEP 2.0 proposal. The release was produ
JDK Release Process (JEP 3).

Features

339: Edwards-Curve Digital Signature Algorithm (EdDSA)
360: Sealed Classes (Preview)
371: Hidden Classes
372: Remove the Nashorn JavaScript Engine
373: Reimplement the Legacy DatagramSocket API
374: Disable and Deprecate Biased Locking
375: Pattern Matching for instanceof (Second Preview)
377: ZGC: A Scalable Low-Latency Garbage Collector
378: Text Blocks
379: Shenandoah: A Low-Pause-Time Garbage Collector
381: Remove the Solaris and SPARC Ports
383: Foreign-Memory Access API (Second Incubator)
384: Records (Second Preview)
385: Deprecate RMI Activation for Removal

JDK 16

JDK 16 is the open-source reference implementat
Platform, as specified by by JSR 390 in the Java C

JDK 16 reached General Availability on 16 March
under the GPL are available from Oracle; binaries
shortly.

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

Features

338: Vector API (Incubator)
347: Enable C++14 Language Features
357: Migrate from Mercurial to Git
369: Migrate to GitHub
376: ZGC: Concurrent Thread-Stack Processi
380: Unix-Domain Socket Channels
386: Alpine Linux Port
387: Elastic Metaspace
388: Windows/AArch64 Port
389: Foreign Linker API (Incubator)
390: Warnings for Value-Based Classes
392: Packaging Tool
393: Foreign-Memory Access API (Third Incubator)
394: Pattern Matching for instanceof
395: Records
396: Strongly Encapsulate JDK Internals by Default
397: Sealed Classes (Second Preview)

JDK 17

JDK 17 is the open-source reference implementation of version 17 of the Java SE Platform, as specified by by JSR 390 in the Java Community Process.

JDK 17 reached General Availability on 14 September 2021. 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

306: Restore Always-Strict Floating-Point Semantics
356: Enhanced Pseudo-Random Number Generators
382: New macOS Rendering Pipeline
391: macOS/AArch64 Port
398: Deprecate the Applet API for Removal
403: Strongly Encapsulate JDK Internals
406: Pattern Matching for switch (Preview)
407: Remove RMI Activation
409: Sealed Classes
410: Remove the Experimental AOT and JIT Compiler
411: Deprecate the Security Manager for Removal
412: Foreign Function & Memory API (Incubator)
414: Vector API (Second Incubator)
415: Context-Specific Deserialization Filters

WHY MIGRATE? - NEW JAVA 11 FEATURES

- var for local variables (JEP 286)
- New methods in classes (Collection, etc) (JEP 269)

```
-   List<String> myList = ArrayList<String>();
-   myList.add(“Georgia”);
-   myList.add(“Minnesota”);
-   myList.add(“Texas”);
-   myList = Collections.unmodifiableList(myList);
+   var myList = List.of(“Georgia”, “Minnesota”, “Texas”);
```

WHY MIGRATE? - NEW JAVA 17 FEATURES

Text Blocks (JEP 378)

```
-    String output = "Now you can more\n" +
-                    "easily write Strings\n" +
-                    "that span multiple lines.";
+    String output = """
+            Now you can more
+            easily write Strings
+            that span multiple lines.""";
```

WHY MIGRATE? - NEW JAVA 17 FEATURES

Helpful NullPointerExceptions (JEP 358)

```
- Exception in thread "main" java.lang.NullPointerException
-     at ...
+ Exception in thread "main" java.lang.NullPointerException:
+ Cannot invoke "String.concat(String)" because "MyClass.coffee"
+ is null
+     at ...
```

WHY MIGRATE? - NEW JAVA 17 FEATURES

Records (JEP 395)

```
-  public class Car {
-
-      private final String color;
-      private final String make;
-      private final String model;
-
-      public Car(String color, String make, String model) {
-          this.color = color;
-          this.make = make;
-          this.model = model;
-      }
-
-      // Getters for color, make, model
-      // equals, toString, and hashCode methods
-  }
+   public record Car (String color, String make, String model) {}
```

WHY MIGRATE? - NEW JAVA 17 FEATURES

Switch enhancements (JEP 361, JEP 406)

```
-    String result = "";
-    switch (day) {
-       case MONDAY:
-       case TUESDAY:
-       case WEDNESDAY:
-       case THURSDAY:
-       case FRIDAY:
-          result = "Weekday";
-          break;
-       case SATURDAY:
-       case SUNDAY:
-          result = "Weekend";
-          break;
-       default:
-          throw new IllegalStateException("Unknown day: " + day);
-     }
+    var result = switch (day) {
+       case MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY -> "Weekday";
+       case SATURDAY, SUNDAY -> "Weekend";
+       default -> throw new IllegalStateException("Unknown day: " + day);
+    };
```

WHY MIGRATE? - NEW JAVA 17 FEATURES

- instanceof Pattern Matching (JEP 394)

```
-    if (sport instanceof Baseball) {
-       Baseball baseball = (Baseball) sport;
-       baseball.pitch();
-    }
+    if (sport instanceof Baseball baseball) {
+       baseball.pitch();
+    }
```

Security Related Changes

- Crypto
- With Applet support removed, Security Manager deprecated for removal
- Strongly Encapsulated JDK Internals (JEP 403)
- Sealed classes (JEP 409)
- Hidden classes (JEP 371)

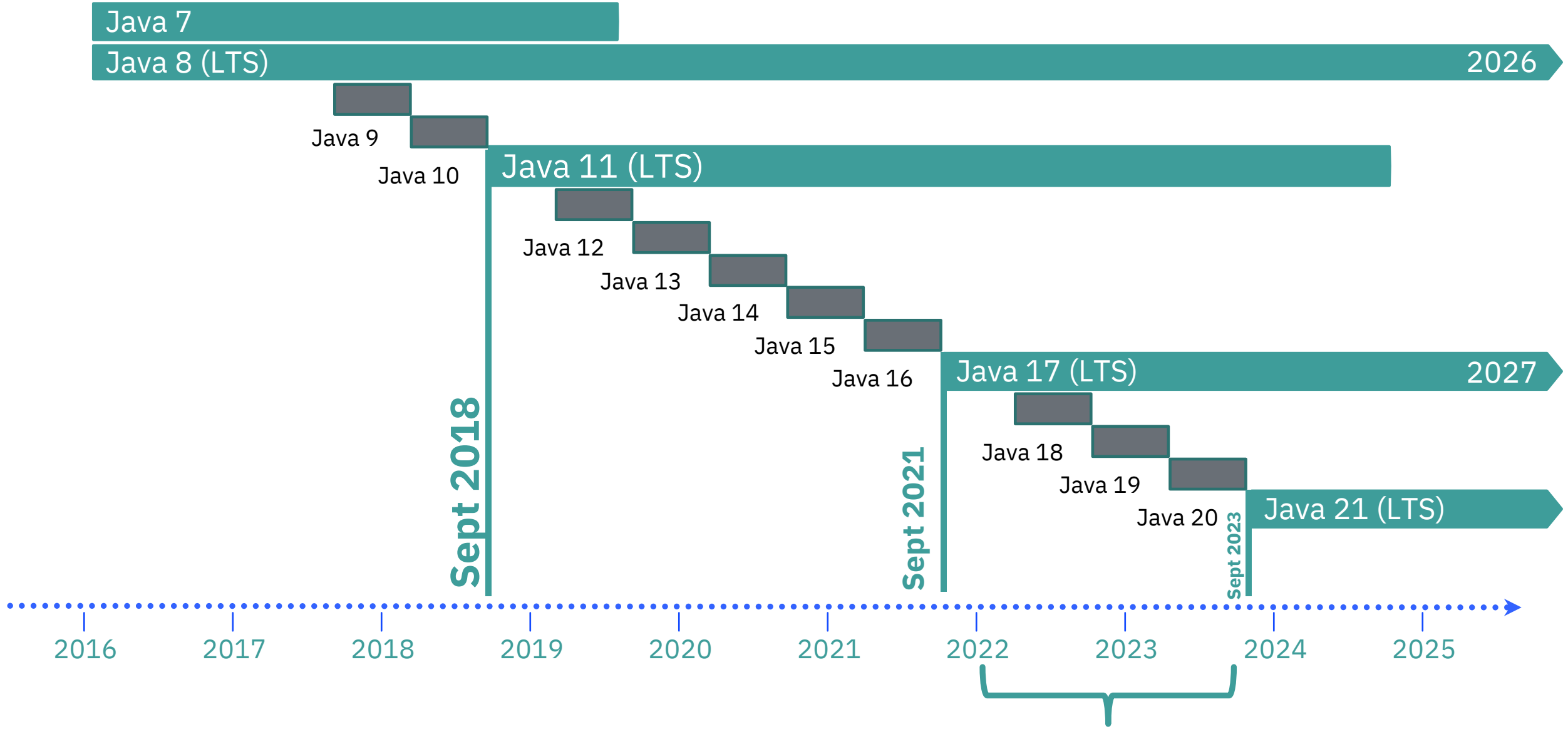
Library/tooling developers need to keep up

...and of course: Java 8 and 11 will EOL

MAKING THE MOVE TO JAVA 17

WHAT IS MY TARGET JAVA VERSION?

JAVA TIMELINE

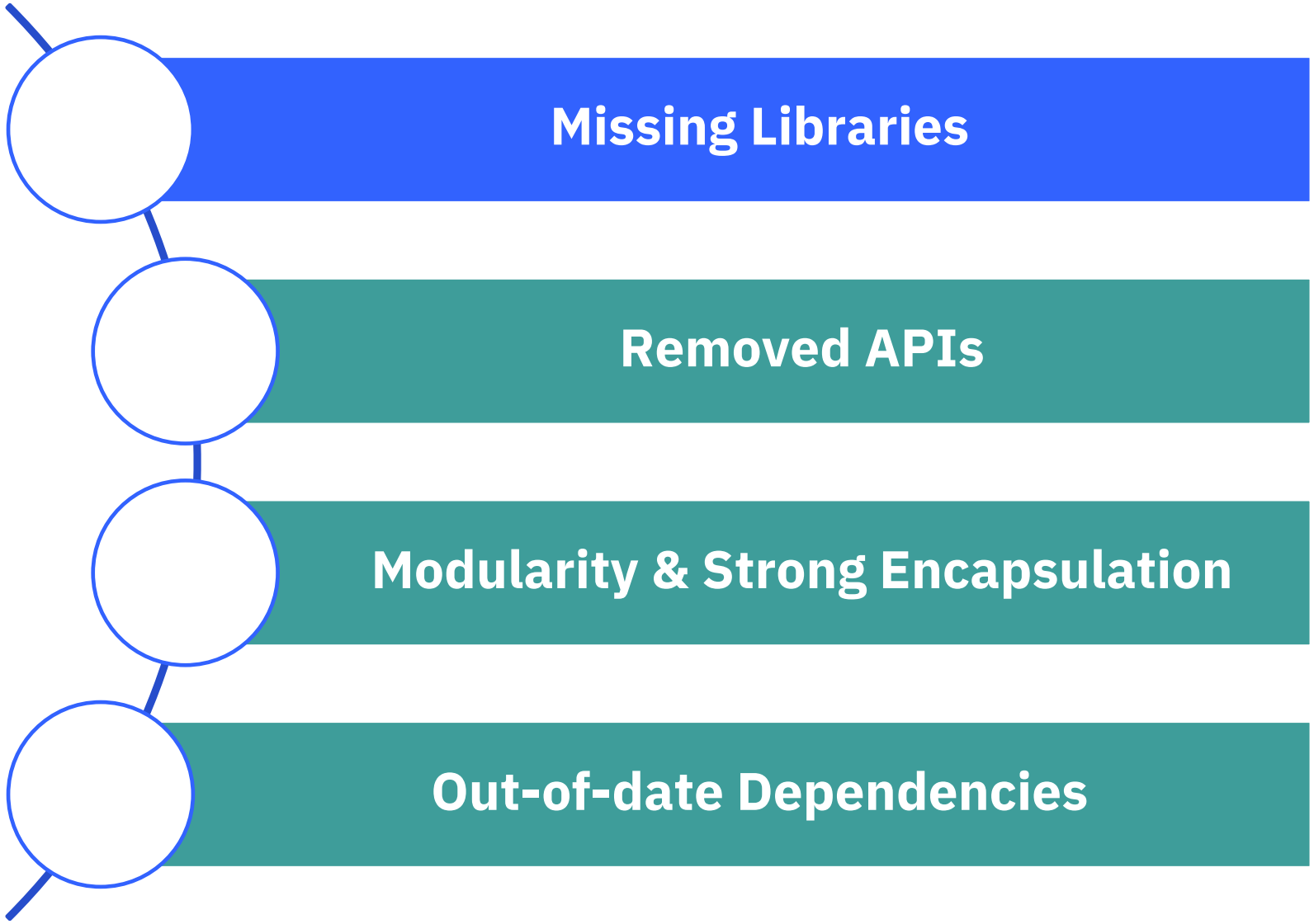


LTS: Long Term Support - Production

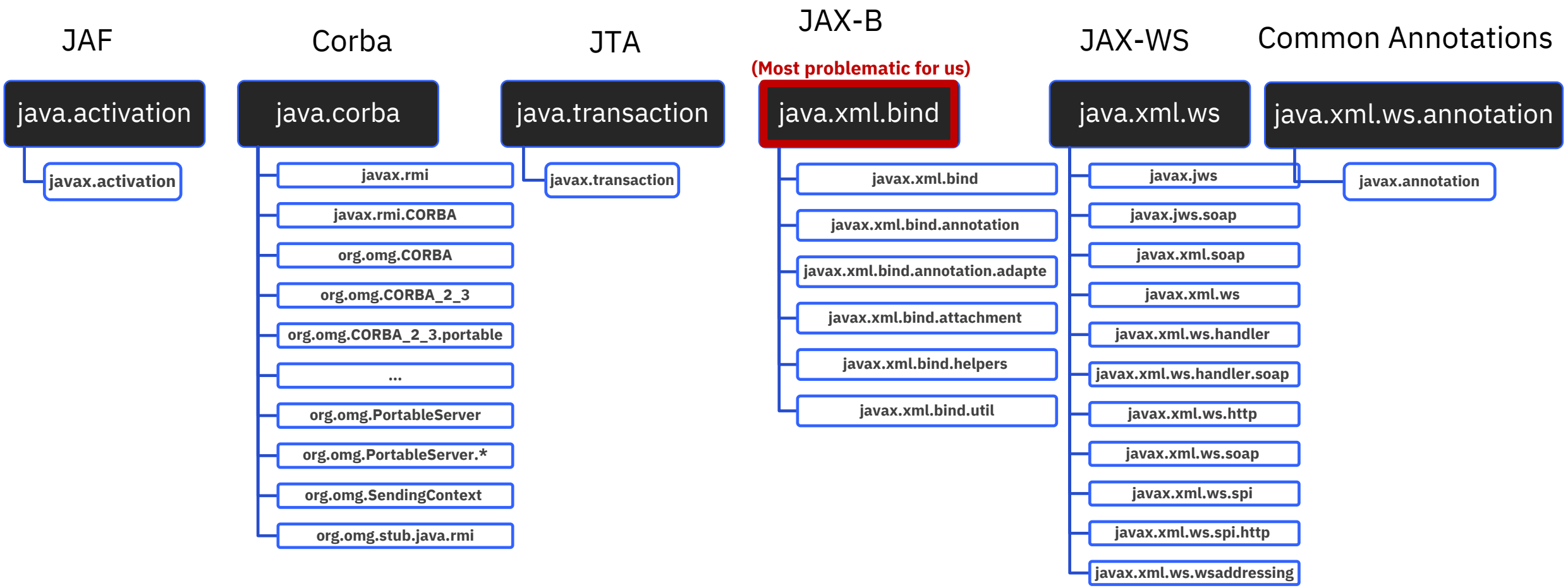
MAKING THE MOVE TO JAVA 17

WHAT ARE THE TOP MIGRATION ISSUES FOR JAVA 8/11 → JAVA 17+?

Top Migration Issues



Java 11+ (Java EE)



Option 1: Package your own dependencies

Option 2: Rely on the app server to provide them (Open Liberty with Java 11+)

JAVA 11 +

Java Web Start → OpenWebStart

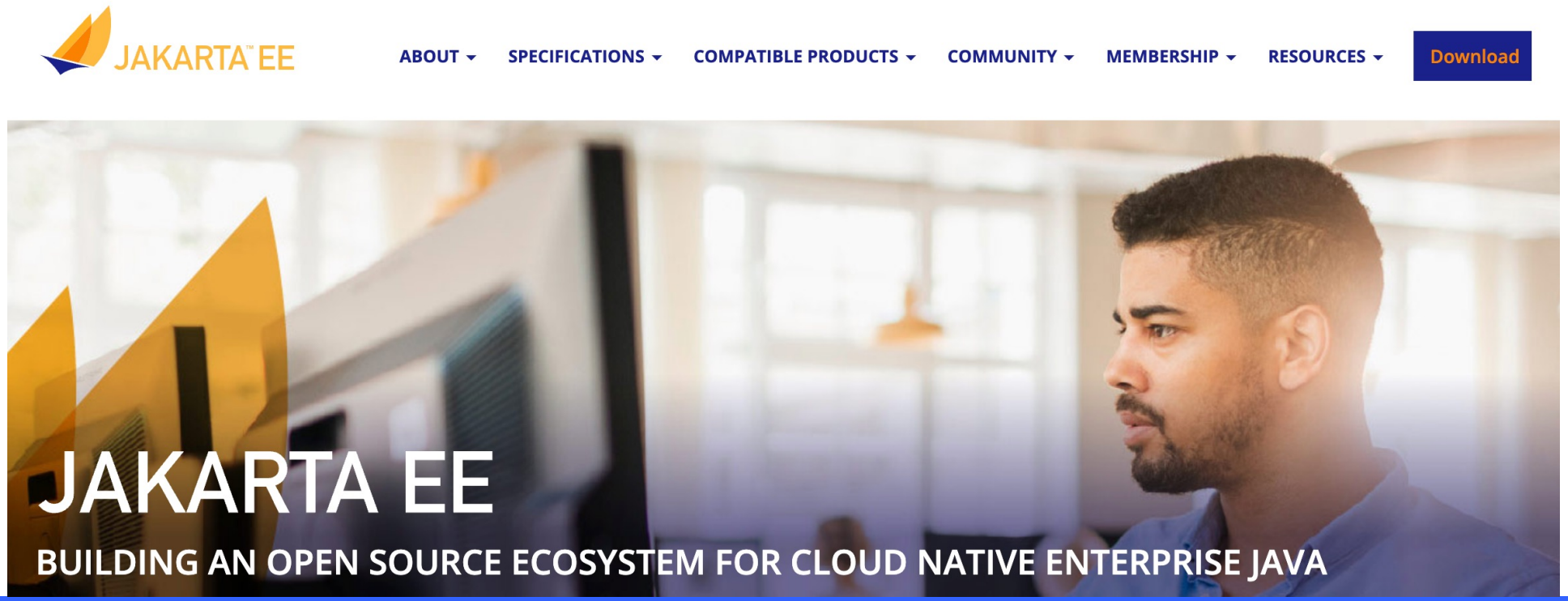
openwebstart.com

JavaFX → JavaFX

openjfx.io

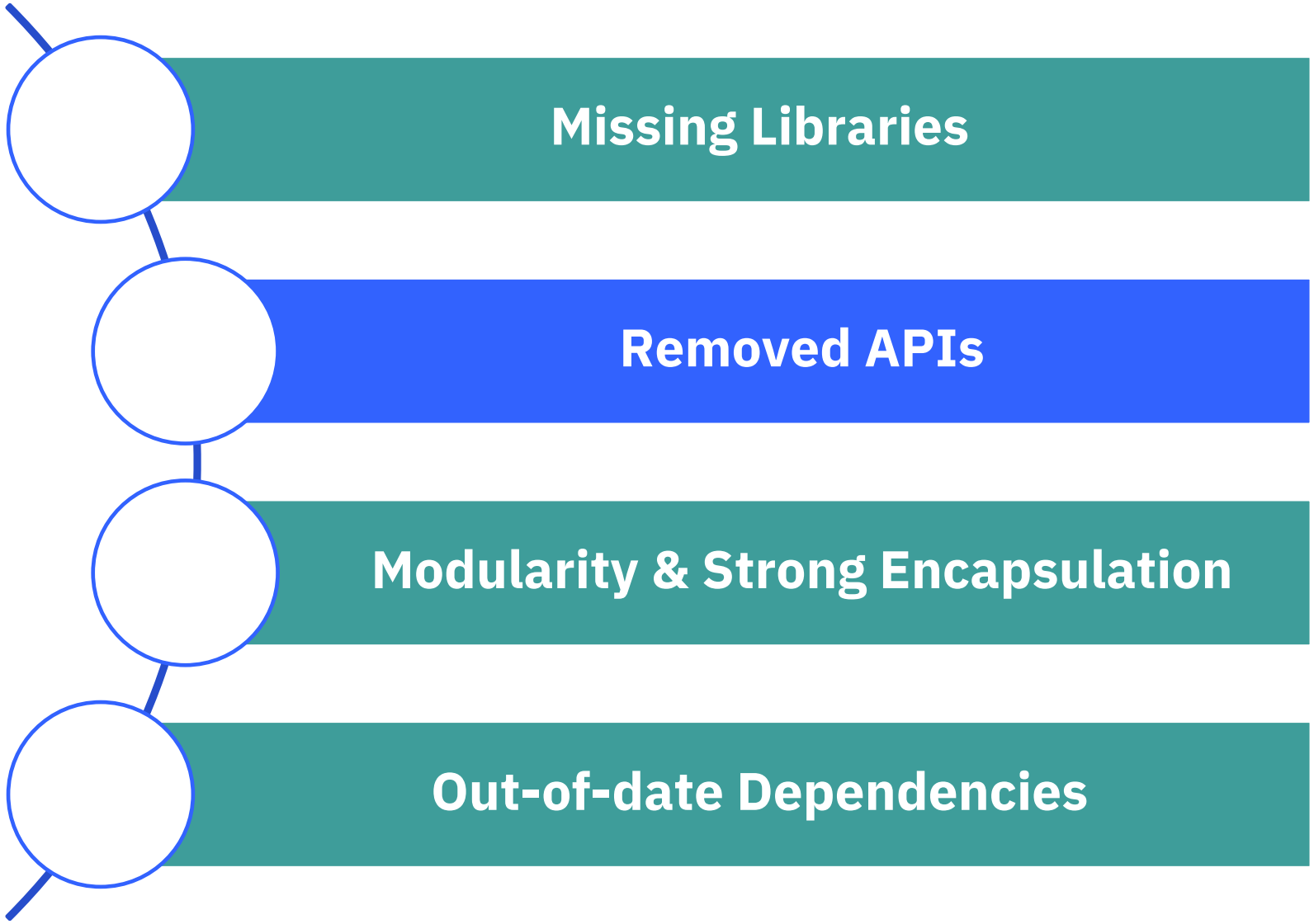
JAVA 17 +

Nashorn JavaScript Engine github.com/openjdk/nashorn



Java 11+: Java EE (Checkout jakarta.ee)

Top Migration Issues



JAVA 11

Removed Packages/Classes

- com.sun.awt.AWTUtilities
- com.sun.image.codec.jpeg.*
- com.sun.java.browser.plugin2.DOM
- com.sun.security.auth.callback.DialogCallbackHandler
- com.sun.security.auth.module.SolarisLoginModule
- com.sun.security.auth.module.SolarisSystem
- 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.SolarisNumericGroupPrincipal principal
+ com.sun.security.auth.UnixNumericGroupPrincipal principal
```

JAVA 11

Removed Packages/Classes

- com.sun.xml.internal.bind.*
- java.awt.dnd.peer.*
- java.awt.peer.*
- javax.security.auth.Policy
- sun.misc.BASE64Decoder
- sun.misc.BASE64Encoder
- sun.misc.Unsafe.defineClass
- sun.plugin.dom.DOMObject

```
- if(button.getPeer() != null) {
+ if(button.isDisplayable()) {
```

JAVA 11

Removed Methods

- java.lang.Runtime.getLocalizedInputStream(..)
- java.lang.Runtime.getLocalizedOutputStream(..)
- java.lang.Runtime.runFinalizersOnExit(..)
- java.lang.SecurityManager.checkAwtEventQueueAccess()
- java.lang.SecurityManager.checkMemberAccess(..)
- java.lang.SecurityManager.checkSystemClipboardAccess()
- 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()

```
- securityManager.checkAwtEventQueueAccess();
+ securityManager.checkPermission(permission);
```

JAVA 11

Removed Methods

- java.lang.System.runFinalizersOnExit(..)
- java.lang.Thread.destroy()
- java.lang.Thread.stop(java.lang.Throwable)
- 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.util.logging.LogManager.addPropertyChangeListener(..)
- java.util.logging.LogManager.removePropertyChangeListener(..)

"Finalizers are inherently problematic, and their use can lead to performance issues, deadlocks, hangs, and other problematic behavior." - JDK-8165641

JAVA 17

Removed Packages/Classes

- com.sun.awt.SecurityWarning
- java.security.acl.*
- java.util.jar.Pack200*
- jdk.net.SocketFlow*
- java.rmi.activation*

Removed Constants

- java.management.rmi.RMIConnectorServer.CREDENTIAL_TYPES

JAVA 17

Removed Methods

- java.io.FileInputStream.finalize()
- java.io.FileOutputStream.finalize()
- java.lang.Runtime.traceInstructions(boolean)
- java.lang.Runtime.traceMethodCalls(boolean)
- java.util.zip.ZipFile.finalize()
- java.util.zip.Inflater.finalize()
- java.util.zip.Deflater.finalize()
- netscape.javascript.JSObject.getWindow(Applet)
- sun.misc.Unsafe.defineAnonymousClass(Class<?> hostClass, byte[] data, Object[] cpPatches)
- javax.net.ssl.SSLSession.getPeerCertificateChain()**
- java.lang.thread.countStackFrames()**

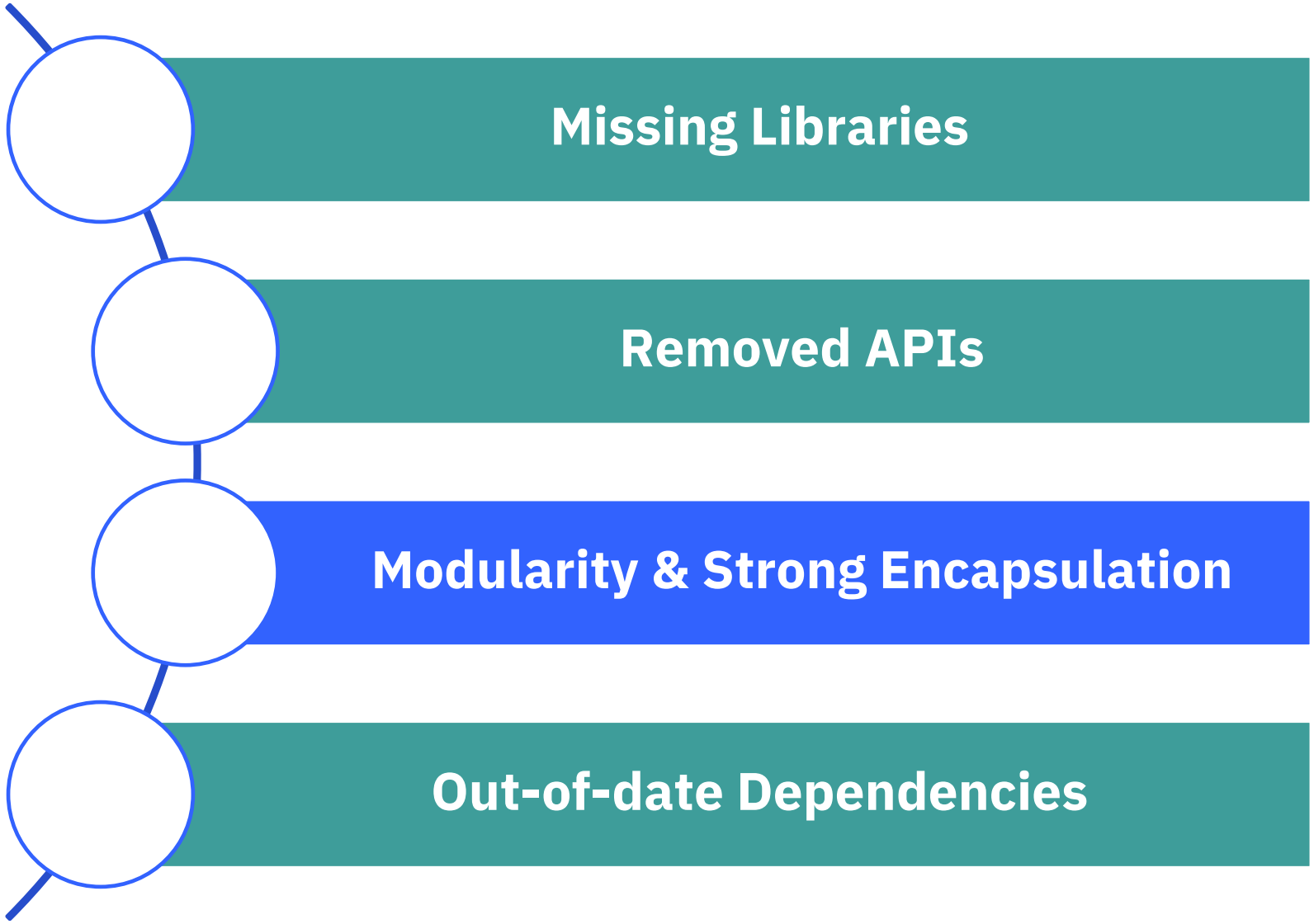
**APIs now throw UnsupportedOperationException

Removed Constructors

- javax.tools.ToolProvider()
- java.lang.invoke.ConstantBootstraps()
- java.lang.relect.Modifier()

```
- javax.net.ssl.SSLSession.getPeerCerficateChain();
+ javax.net.ssl.SSLSession.getPeerCerficates();
```

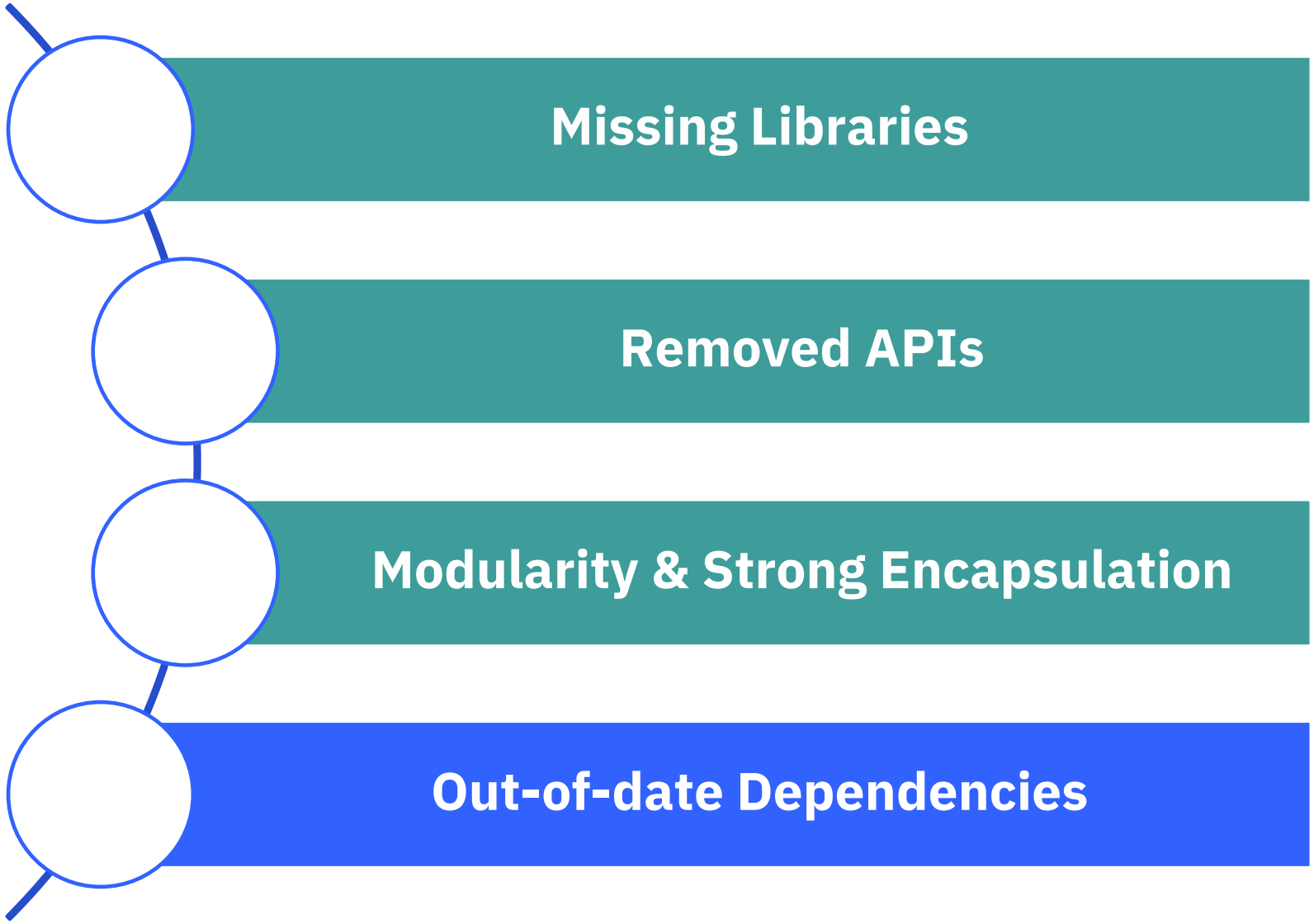
Top Migration Issues



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
Use find issues in your application	▪ Use JDeps or --illegal-access=warn|debug
If migrating to Java 11	▪ Use --illegal-access=permit
If migrating to Java 17+	▪ --add-opens launcher option ▪ Add-Opens jar Manifest Attribute

Top Migration Issues



OUT-OF-DATE DEPENDENCIES

Libraries (ASM, Mockito, Spring, Spring Boot, etc)

Build Tools (Maven, Gradle, etc)

IDE (Eclipse, IntelliJ, VS Code, etc)

Application servers (Open Liberty, JBoss, WebLogic, etc)

My advice: take this time to upgrade your dependencies to the latest supported version. Setup automation tools.

MAKING THE MOVE TO JAVA 17

WHAT ABOUT DEPRECATIONS?

KEEP AN EYE ON DEPRECATIONS

Some recently removed deprecations include:

- Java RMI Activation
- `Thread.destroy()` and `Thread.stop(Throwable t)`

Some APIs currently deprecated for removal:

- Java Security Manager
- Primitive wrapper class constructors

Full list: https://docs.oracle.com/en/java/javase/17/docs/api/deprecated-list.html#for-removal

MAKING THE MOVE TO JAVA 17

WHAT TOOLS CAN I USE TO MAKE MY MIGRATION EASIER?

USE CONTAINERS TO STAY UP TO DATE

- Use containers to easily keep your Java versions up to date
- This can be used to ensure you stay up to date with the latest fix packs, as well as test out new Java versions

TOOLS

Application Binary Scanner Tool

- Download: ibm.biz/WAMT4AppBinaries or Maven
- Command line tool (free personal and commercial use)
- Scans an application binary for migration issues and produces a report with identified problems, solutions and resource links
- Documentation

JDeps

- Command utility shipped with JDK (run with the target Java version – Java 11+)
- Migration Relevant Option: -jdkinternals
- Binary scanner will recommend running JDeps if it identifies internal APIs
- Documentation

Tools to automate dependency updates (various options)

- Use an auto-upgrade dependency tool to keep your dependencies up-to-date
- Configure the tools to update based on various settings (frequency, etc)
- Example: Dependabot

Eclipse Transformer

- Transform application binaries or source code to use the new package names in Jakarta EE 9+
- Documentation

DEMO

NEXT STEPS

Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Evaluate your applications using the tools.	Try running your application on Java 17. If your app runs successfully, you can skip the next two steps.	Debug and fix your application code or build files. Repeat steps 1-3.	Update your dependencies and tools as needed.	Run all your application tests.	Celebrate!

HELPFUL LINKS

- Our blog on adding the Java 17 feature to the migration tools (written instructions on running binary scanner): https://developer.ibm.com/tutorials/migration-to-java-11-made-easy/
- Binary Scanner and Eclipse IDE Plugin Source Scanner downloads: ibm.biz/wamtdownloads
  - Documentation: https://www.ibm.com/docs/wamt
- Link to the java17demo GitHub repo: https://github.com/alexsm82/java17demo
- Oracle Java SE Support Roadmap: https://www.oracle.com/java/technologies/java-se-support-roadmap.html
- Java Version Almanac: https://javaalmanac.io/
- Andy Guibert's blog on OpenLiberty Java 11+ support: https://openliberty.io/blog/2019/02/06/java-11.html
- Blog on running the binary scanner with Maven: https://community.ibm.com/community/user/wasdevops/blogs/alex-motley1/2022/04/12/applications-pipelines-and-migrations

IBM SESSIONS

- Tuesday 10:00AM Fast JVM Startup with Checkpoint and Restore – Tobi Ajila | JVM Platform
- Tuesday 2:15PM Better, Stronger, Faster Java in the Cloud - Jarek Gawor and Tobi Ajila |Main Room (Java)
- 3:15 Book signing – Cloud Native Application Development with MicroProfile and Open Liberty and Birthday Cake
- Wednesday 10:00AM Making the Move to Java 17 – Alex Motley and Cindy High | Main Room (Java)
- Wednesday 1:15PM Relook at Microservices – Emily Jiang | Main Room (Java)

QUESTIONS?

@alexsmotley

@CTHigh