

Moving Towards JDK 12: Delivering New Java Features

© Copyright Azul Systems 2015

Simon Ritter

Deputy CTO, Azul Systems

azul.com



@speakjava

JDK 9: Big And Small Changes

JDK 9

- Process API Updates
- HTTP 2 Client
- Improve Contended Locking
- Unified JVM Logging
- Compiler Control
- Variable Handles
- Segmented Code Cache
- Smart Java Compilation, Phase 1
- The Modular JDK
- Modular Source Code
- Elide Deprecation Warnings on Import Statements
- Resolve Lint and Doclint Warnings
- Milling Project Coin
- Remove GC Combinations Deprecated in JDK 8
- Tiered Attribution for javac
- Process Import Statements Correctly
- Annotations Pipeline 2.0
- Datagram Transport Layer Security (DTLS)
- Modular Run-Time Image
- Simplified Doclet API
- jschell: The Java Shell (Read-Eval-Print Loop)
- New Version-String Scheme
- HTML5 Javadoc
- Javadoc Search
- UTF-8 Property Files
- Unicode 7.0
- Add More Diagnostic Commands
- Create PKCS12 Keystores by Default
- Remove Launch-Time JRE Version Selection

- Improve Secure Application Performance
- Generate Run-Time Compiler Tests Automatically
- Test Class-File Attributes Generated by javac
- Parser API for Nashorn
- Linux/AArch64 Port
- Multi-Release JAR Files
- Remove the JVM TI hprof Agent
- Remove the jhat Tool
- Improve JVM Compiler Space
- Network-Layer Flow Control and Negotiation Extension
- Validating Command-Line Arguments
- Leverage Constructive Grammar Annotations (CGA)
- Compiler-Over Platform-Specific
- Make GC the Default G1
- OCSP Stalling for TLS
- Store International Strings in UTF-8
- Multi-Resolution Image
- Use a Single Data Layout
- JavaFX UI Controls and CSS APIs for Localization
- Internationalized Strings
- Merge Selected Xerces 2.12 Fixes into JAXP
- BeanInfo Annotations
- Update JavaFX/Media to Newer Version of GStreamer
- HarfBuzz Font-Layout Engine
- Stack-Walking API
- Encapsulate Most Internal APIs
- Module System
- TIFF Image I/O
- HiDPI Graphics on Windows and Linux

- Platform Logging API and Service
- Marlin Graphics Renderer
- More Concurrency Updates
- Unicode 8.0
- XML Catalogs
- Convenience Factory Methods for Collections
- Reserved Stack Areas for Critical Sections
- Unified Class-File Format
- Platform-Specific Features
- DRBG and SecureRandom Implementations
- Enhanced Method Handles
- Modular Java Application Packaging
- Dynamic Linking of Libraries and Defined Object Models
- Enhanced Object Model
- Addition of WeakReferences Objects in G1
- Improve Test Failure Tracing and Shooting
- Indify String Concatenation
- HotSpot C++ Unit-Test Framework
- Linker: The Java Linker
- Enable the Java Linker
- New HotSpot System
- Spin-Wait Hints
- SHA-3 Hash Algorithms
- Disable SHA-1 Certificates
- Deprecate the Applet API
- Filter Incoming Serialization Data
- Implement Selected ECMAScript 6 Features in Nashorn
- Linux/s390x Port

Java Platform Module System (JPMS)

- The core Java libraries are now a set of modules (JEP 220)
 - 75 OpenJDK modules: 26 Java SE, 48 JDK
 - Oracle JDK: 14 additional JDK, 8 JavaFX, 2 Oracle specific
- Most internal APIs now encapsulated (JEP 260)
 - `sun.misc.Unsafe`
 - Some can be used with command line options

jlink: The Java Linker (JEP 282)

```
$ jlink --module-path $JDKMODS:$MYMODS \  
  --addmods com.azul.zapp --output myimage
```

```
$ myimage/bin/java --list-modules
```

```
java.base@9
```

```
java.logging@9
```

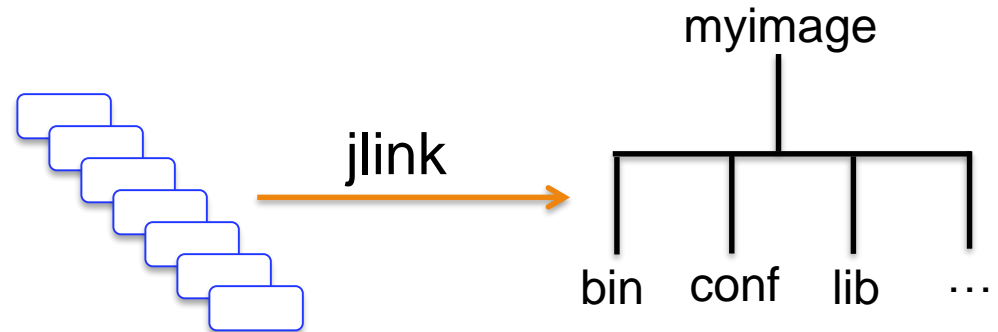
```
java.sql@9
```

```
java.xml@9
```

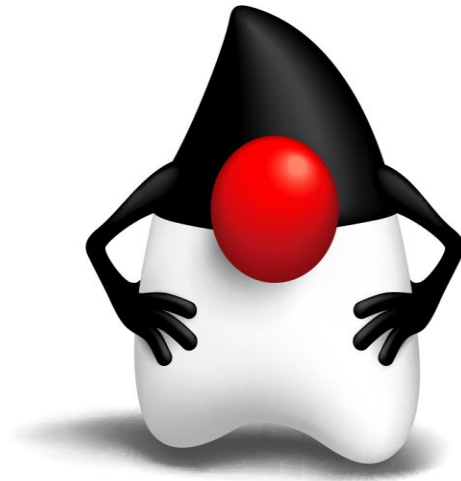
```
com.azul.zapp@1.0
```

```
com.azul.zoop@1.0
```

```
com.azul.zeta@1.0
```



Moving Java Forward Faster



OpenJDK: New Release Model

- A new version of the JDK will be released every six months
 - March and September
 - Started last year with JDK 10 and JDK 11
 - Continuing this year with JDK 12 and JDK 13
- OpenJDK development will be more agile
 - Previous target was a release every two years
- Features will be included only when ready
 - Targeted for a release when feature complete

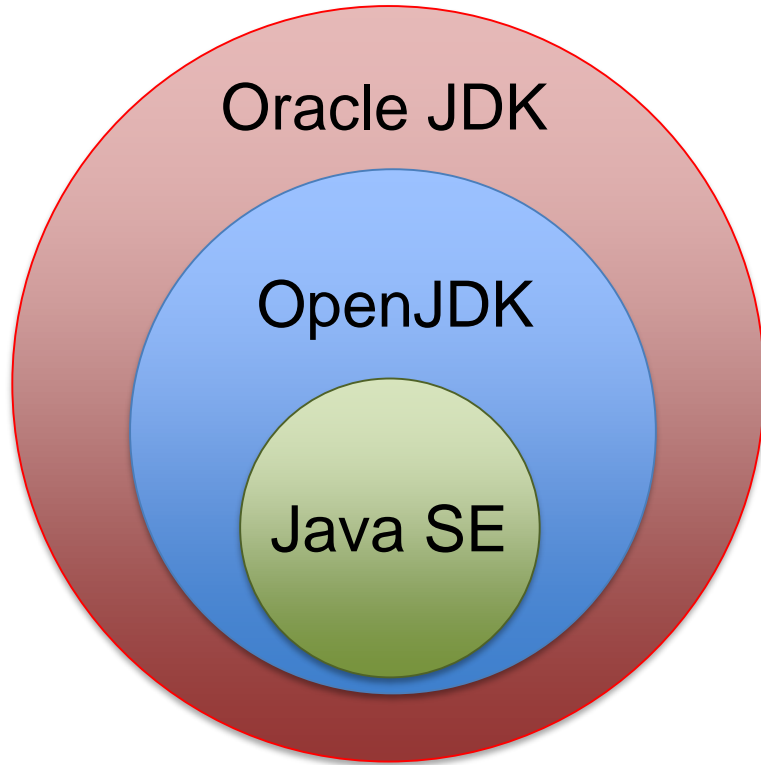
Long Term Support Releases

- Long term support for all releases is not practical
 - One Long Term Support (LTS) release every three years
- JDK 8 has been classified as an LTS release
 - Last JDK 8 public update last week for commercial users!
 - Non-commercial users get updates until December 2020
- JDK 11 is an LTS release
 - JDK 9 and JDK 10 are feature releases
 - Updated only until next release

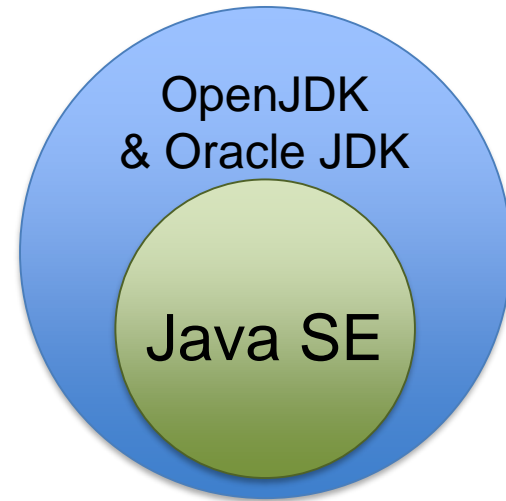
Oracle JDK Binary

- Traditional Oracle branded binary (java.oracle.com)
 - Oracle Binary Code License (FoU restrictions)
- New OpenJDK binary (jdk.java.net)
 - GPLv2 with CPE license (no restrictions)
 - Security and bug fix updates only for six months
 - Only until next JDK release
 - Two scheduled updates

Converged Binaries



JDK 10 and earlier



JDK 11 and later

Converged Binaries (JDK 11)

- Some closed-source parts of the JDK will be open-sourced
 - Flight recorder
 - Mission control
 - Others
- Other closed-source parts will be removed
 - Browser Plugin
 - Java Web Start
 - JavaFX

JDK 9 Onwards And Compatibility

"Clean applications that just depend on java.se
should just work" - Oracle

JDK 9: The Clean Up Starts

- JDK 9 was a significant change for Java
 - Deprecated APIs were removed for the first time
 - Six methods and one class
 - JDK 10 removed 1 package, 6 classes, 9 methods and 1 field
 - Redundant features eliminated
 - jhat tool, JVM TI hprof agent
 - Numerous deprecated GC options removed
- JDK 10 and 11 have continued this work
- More features will be removed in the future
 - CMS GC, Nashorn and Pack200 all deprecated. Others?

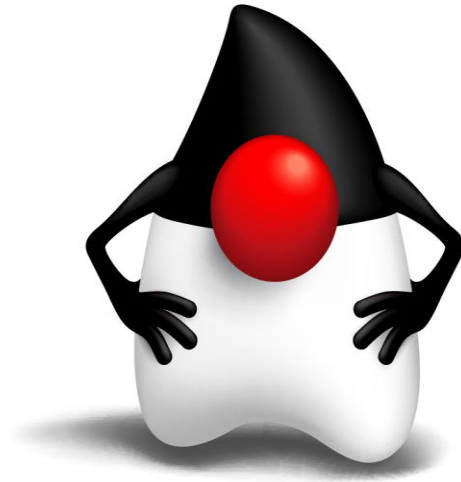
Compatibility Not Guaranteed

- New versions of Java may include breaking changes
 - Anything for removal will be deprecated first
 - Minimum of one release warning
 - Could be only six months

Java Updates

- Oracle JDK binary has LTS versions (JDK 11, 17, etc.)
 - Oracle OpenJDK does not
- Oracle JDK 11 was released under a different license
 - Free for development and testing
 - Requires a Java SE subscription for use in production
- Oracle OpenJDK binaries only updated for six months
- JDK 8 can be used indefinitely for free
 - But without any further security patches and bug fixes

JDK 10



Local Variable Type Inference (JEP 286)

- Java gets var

```
var userList = new ArrayList<String>(); // infers ArrayList<String>
var stream = list.stream();           // infers Stream<String>
```

```
for (var name : userList) {           // infers String
    ...
}
```

```
for (var i; i < 10; i++) {           // infers int
    ...
}
```


var: Clearer try-with-resources

```
try (InputStream inputStream = socket.getInputStream();
    InputStreamReader inputStreamReader =
        new InputStreamReader(inputStream, UTF_8);
    BufferedReader bufferedReader = new BufferedReader(inputStreamReader)) {
    // Use bufferedReader
}
```

var: Clearer try-with-resources

```
try (var inputStream = socket.getInputStream();  
    var inputStreamReader = new inputStreamReader(is, UTF_8);  
    var bufferedReader = new BufferedReader(isr)) {  
    // Use bufferedReader  
}
```

var: Reserved Type (Not Keyword)

```
var var = new ValueAddedReseller(); ✓
```

```
public class var {  
    public var(String x) {  
        ...  
    }  
}
```



```
public class Var {  
    public Var(String x) {  
        ...  
    }  
}
```



JDK 10: JEPs

- JEP 307: Parallel Full GC for G1
- JEP 310: Application Class-Data Sharing
- JEP 317: Experimental Java-based JIT compiler (Graal)
- JEP 319: Root Certificates
- JEP 296: Consolidate JDK forests into single repo

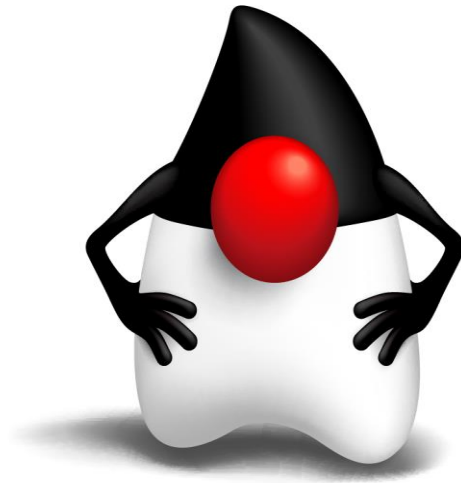
JDK 10: JEPs

- JEP 316: Heap allocation on alternative devices (Intel)
- JEP 313: Remove javah tool
- JEP 304: Garbage Collector Interface (Red Hat)
- JEP 312: Thread-Local Handshakes

JDK 10: APIs

- 73 New APIs
 - `List, Set, Map.copyOf(Collection)`
 - Collectors
 - `toUnmodifiableList`
 - `toUnmodifiableMap`
 - `toUnmodifiableSet`
 - `Optional.orElseThrow()`

JDK 11



JDK 11

- 17 JEPs
- 3 from outside Oracle
 - JEP 318: Epsilon garbage collector (Red Hat)
 - JEP 315: Improve Aarch64 intrinsics (Red Hat)
 - JEP 331: Low overhead heap profiling (Google)

323: Extend Local-Variable Syntax

- Local-variable syntax for lambda parameters

```
list.stream()  
  .map(s -> s.toLowerCase())  
  .collect(Collectors.toList());
```

```
list.stream()  
  .map((var s) -> s.toLowerCase())  
  .collect(Collectors.toList());
```

```
list.stream()  
  .map(@NotNull var s) -> s.toLowerCase()  
  .collect(Collectors.toList());
```

327: Unicode 10 Support

- 8,518 new characters (seriously)
 - Bitcoin symbol
 - Nishu
 - Soyombo, Zanabazar Square
- The long awaited (?) Colbert emoji



330: Launch Single File Source Code

- JDK 10 has three modes for the Java launcher
 - Launch a class file
 - Launch the main class of a JAR file
 - Launch the main class of a module
- JDK 11 adds a forth
 - Launch a class declared in a source file

```
$ java Factorial.java 4
```

Single File Source Code Shebang

```
#!/$JAVA_HOME/bin/java --source 11
public class Factorial {
    public static void main(String[] args) {
        int n = Integer.parseInt(args[0]);
        int r = (n == 0) ? 0 : 1;
        for (int i = 1; i <= n; i++)
            r *= i;
        System.out.println("n = " + n + ", n! = " + r);
    }
}
```

```
$ ./Factorial 4
n = 4, n! = 24
```

Other JDK 11 JEPs

- 181: Nest-based Access Control
- 309: Dynamic Class-file constants
- 321: HTTP client
- 324: Key Agreement with Curve25519 and Curve448
- 329: ChaCha20 and Poly1305 Cryptographic Algorithms
- 332: Transport Layer Security (TLS) 1.3
- 333: ZGC: Experimental low-latency collector
- 335: Deprecate the Nashorn JavaScript Engine
- 336: Deprecate the Pack200 Tools and API

New APIs

- New I/O methods
 - `InputStream nullInputStream()`
 - `OutputStream nullOutputStream()`
 - `Reader nullReader()`
 - `Writer nullWriter()`
- Optional
 - `isEmpty()` // Opposite of `isPresent`
- Character
 - `toString(int)` // Unicode codepoint

New APIs

- New String methods
 - `isBlank()`
 - `Stream lines()`
 - `String repeat(int)`
 - `String strip()`
 - `String stripLeading()`
 - `String stripTrailing()`

New APIs

- Predicate not(Predicate)

```
lines.stream()  
    .filter(s -> !s.isBlank())
```

```
lines.stream()  
    .filter(Predicate.not(String::isBlank))
```

```
lines.stream()  
    .filter(not(String::isBlank))
```


JDK 11: Modules Removed

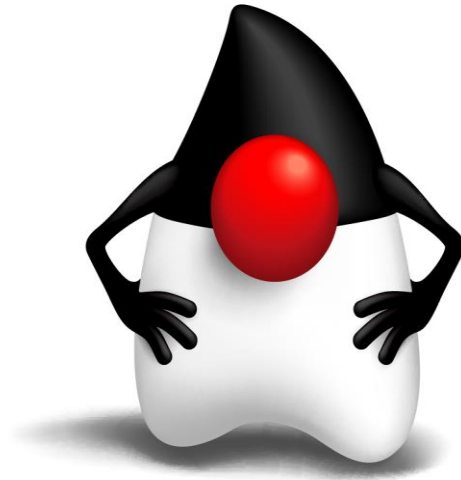
- The `java.se.ee` aggregator-module has been removed
 - `java.corba`
 - `java.transaction`
 - `java.activation`
 - `java.xml.bind`
 - `java.xml.ws`
 - `java.xml.ws.annotation`

Command Line -XX Flags

- Big changes
- JDK 9
 - Removed 187, added 123
- JDK 10
 - Removed 36, added 26
- JDK 11
 - Removed 27, added 53

chriswhocodes.com/hotspot_option_differences.html

What Will Be in JDK 12?



JEP 325: Switch Expressions (Preview)

```
int numLetters;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException("Huh?: " + day); };
```

JEP 325: Switch Expressions

```
int numLetters = switch (day) {  
    case MONDAY, FRIDAY, SUNDAY -> 6;  
    case TUESDAY -> 7;  
    case THURSDAY, SATURDAY -> 8;  
    case WEDNESDAY -> 9;  
    default -> throw new IllegalStateException("Huh?: " + day);  
};
```

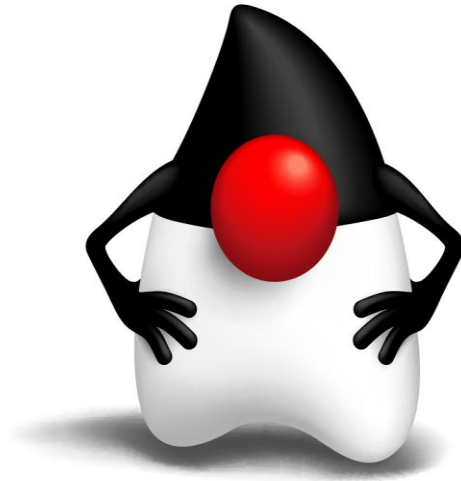
JEPs

- 189: Shenandoah GC (Experimental)
- G1 GC updates
 - 344: Abortable mixed collections
 - 346: Return unused committed memory
- 334: JVM constant API
- 230: Microbenchmark suite
- 341: Default CDS archive

New APIs

- Collectors
 - `teeing(Collector, Collector, BiFunction)`
- Class
 - `describeConstable`
- `CompletableFuture/CompletionStage`
 - Five new methods for exceptions in `CompletionStage`

Longer Term JDK Futures



Project Amber

- Simplifying Java language syntax
- JEP 302: Lambda leftovers
 - Single underscore for unused parameters
- JEP 326: Raw string literals
 - Use single backquote
 - ``c:\Users\simon``
 - ````A string with a `` in it````

JEP 305: Pattern Matching

- Type test and switch statement support to start

```
String formatted;
switch (obj) {
    case Integer i: formatted = String.format("int %d", i); break;
    case Byte b:    formatted = String.format("byte %d", b); break;
    case Long l:   formatted = String.format("long %d", l); break;
    case Double d: formatted = String.format("double %f", d);
break;
    case String s: formatted = String.format("String %s", s); break
    default:       formatted = obj.toString();
}
```

Project Valhalla

- Java has:
 - Primitives: for performance
 - Objects: for encapsulation, polymorphism, inheritance, OO
- Problem is where we want to use primitives but can't
 - `ArrayList<int>` won't work
 - `ArrayList<Integer>` requires boxing and unboxing, object creation, heap overhead, indirection reference

Project Valhalla

- Value types
- "Codes like a class, works like a primitive"
 - Can have methods and fields
 - Can implement interfaces
 - Can use encapsulation
 - Can be generic
 - Can't be mutated
 - Can't be sub-classed

Project Loom

- Further work on making concurrent programming simpler
 - Threads are too heavyweight
- Loom will introduce fibres
 - JVM level threads (remember green threads?)
 - Add continuations to the JVM
 - Use the ForkJoinPool scheduler
 - Much lighter weight than threads
 - Less memory
 - Close to zero overhead for task switching

Azul's Zulu Java



Zulu Java

- Azul's binary distribution of OpenJDK
 - Passes all TCK tests
- JDK 6, 7, 8, 9, 10 and 11 available
- Wide platform support:
 - Intel 64-bit Windows, Mac, Linux
 - Intel 32-bit Windows and Linux
 - ARM 32 and 64-bit
 - PowerPC

www.azul.com/downloads/zulu

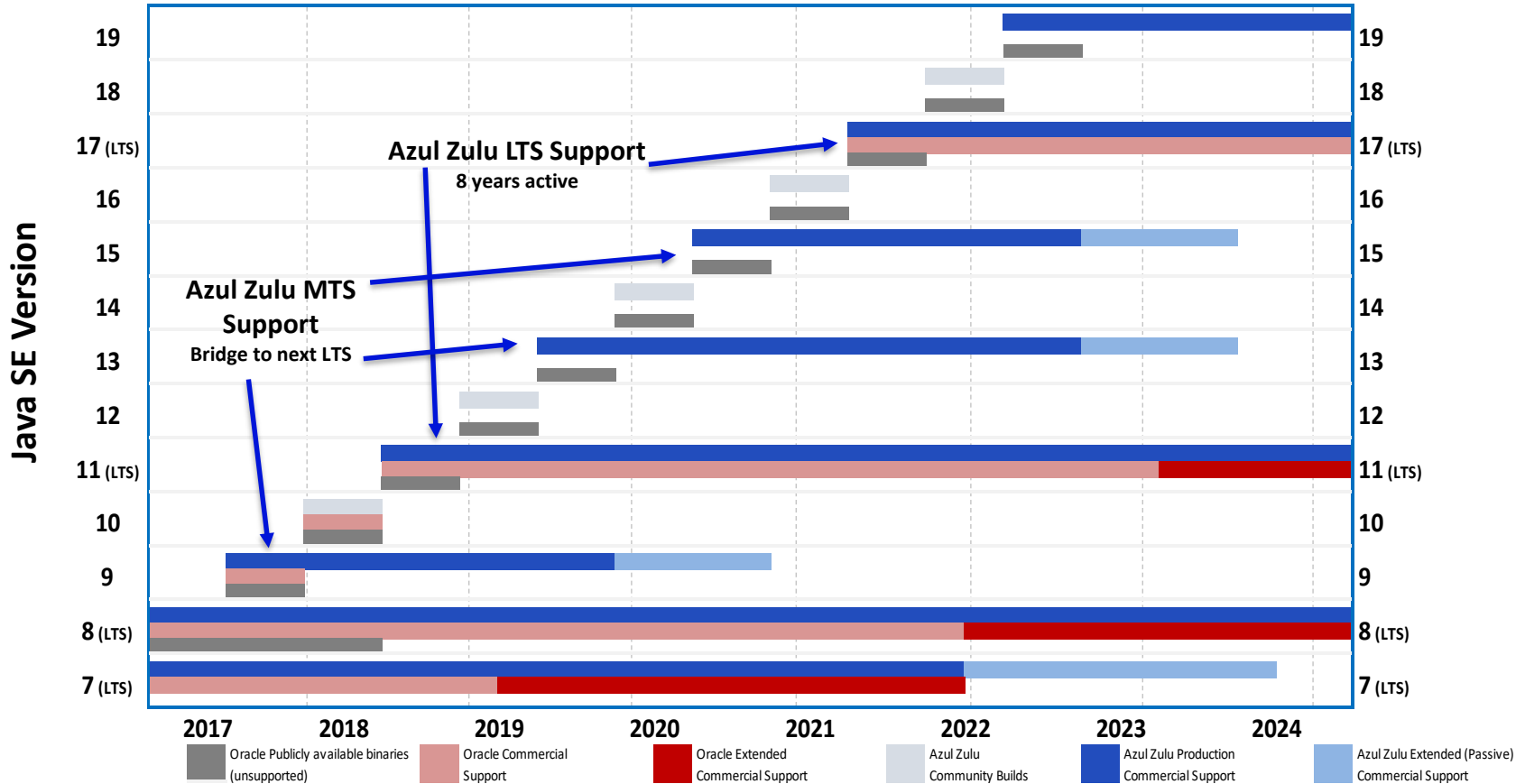
Zulu Extended Support

- Backporting of bug fixes and security patches from supported OpenJDK release
- Zulu 8 supported until March 2026
- LTS releases have 9 years active + 2 years passive support
- Medium Term Support releases
 - Two interim releases between LTS releases (9, 13, 15...)
 - Bridge to LTS releases
 - Supported until 18 months *after* next LTS release

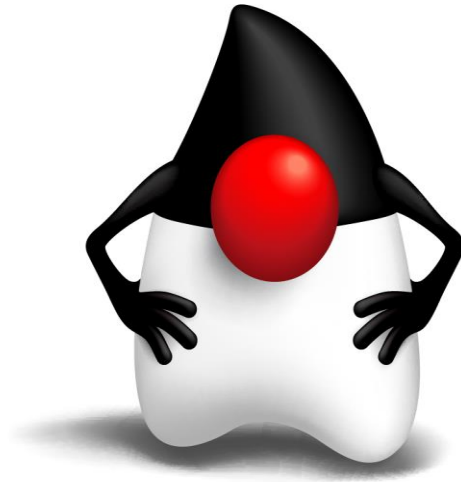
Zulu Complete Support

- 24x7x365 or 8x5 telephone and e-mail contact
 - Report JDK-related problems
- Follow-the-sun engineering team
 - Highly experienced engineers
 - Many ex-Sun and ex-Oracle Java team
- Root-cause and fix problems
 - Generate custom JDK binaries for fixes
 - Upstream fixes to OpenJDK where possible

Java SE Lifecycle: 5+ Years



Summary



Java Continues To Evolve

- Faster Java releases
 - Feature release every 6 months
 - Access to free updates is a consideration
- Lots of ideas to improve Java
 - Value types, fibres, syntax improvements
- Zulu Java has wide platform and JDK version support
 - Very reasonable cost for commercial support

Thank You!

© Copyright Azul Systems 2015

Simon Ritter

Deputy CTO, Azul Systems

azul.com

© Copyright Azul Systems 2019



@speakjava