JDK 9, 10, 11 And Beyond
Delivering New Features in the JDK

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava

JDK 9: Big And Small Changes

Process API Updates
HTTP 2 Client
Improve Contended Locking
Unified JVM Logging
Compiler Control
Variable Handles
Segmented Code Cache
Smart Java Compilation, Phase T
The Modular JDK
Modular Source Code
Elide Deprecation Warnings on I tatement
Resolve Lint and Doclint Warnin
Milling Project Coin
Remove GC Combinations Depre JDK 8
Tiered Attribution for javac
Process Import Statements Corr
Annotations Pipeline 2.0
Datagram Transport Layer Secur S)
Modular Run-Time Imag
Simplified Doclet API
jshell: The Java Shell (Rea nt 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
ve the jhat Tool
JVM Compil ace
ion-Layer Negotia nsion
Valid Comman ag Ar
Levera nstructi GHA KSA
Compil er Platt
Make G efault G
OCSP S or TLS
Store I Strings rch
Multi on Imag
Use ale Data lt
aFX UI Con SS APIs fo rization
t Strings
Merge Selected Xerces Fixes into JAX
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
Platf eatures
DRB Secure mplementations
En Method Ha
Mo va Applicat aging
Dy king of Lar efined Object Models
Enh
Addit s Objects in G1
Improve Test Failure T hooting
Indify String Concaten
HotSpot C++ Unit-Te ework
jlink The Java Link
Enal
New H 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

JDK 9

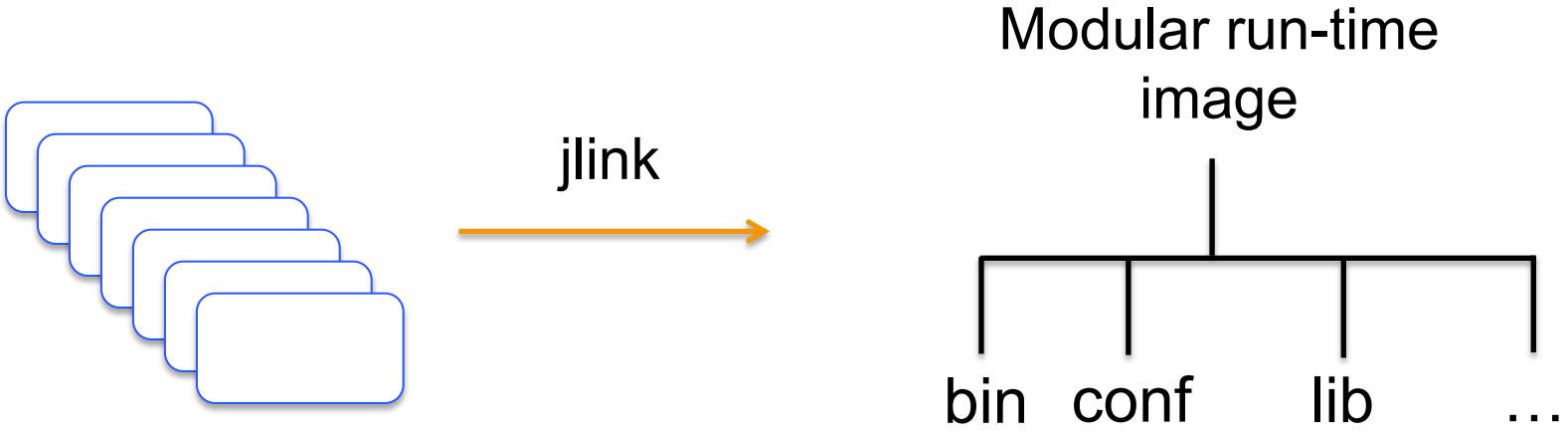
Java Platform Module System (JPMS)

- The core Java libraries are now a set of modules (JEP 220)
  - 97 modules: 28 Java SE, 8 JavaFX, 59 JDK, 2 Oracle
- Most internal APIs now encapsulated (JEP 260)
  - `sun.misc.Unsafe`
  - Some can be used with command line options

Migrating Applications to JPMS

- Initially, leave everything on the classpath
- Anything on the classpath is in the unnamed module
  - All packages are exported
  - The unnamed module depends on all modules
- Migrate to modules as required
  - Try automatic modules
  - Move existing jar files from classpath to modulepath

jlink: The Java Linker (JEP 282)




```
$ jlink --modulepath $JDKMODS \
  --addmods java.base –output myimage

$ myimage/bin/java –-list-modules
java.base@9
```

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
```

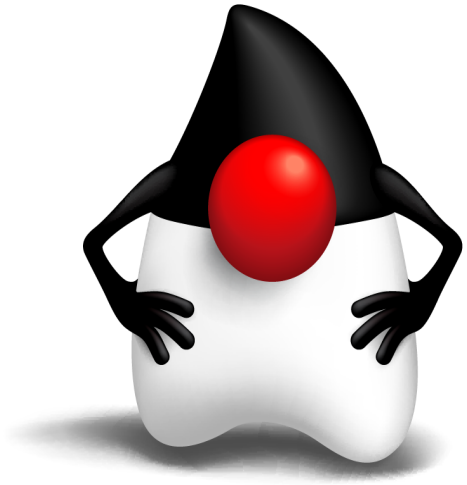
JDK 9 Onwards And Compatibility

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

"Missing" Modules

- Remember, "Clean applications that only use java.se..."
- The `java.se.ee` module is not included by default
  - Compilation and runtime
- Affected modules
  - `java.corba`
  - `java.transaction`
  - `java.activation`
  - `java.xml.bind`
  - `java.xml.ws`
  - `java.xml.ws.annotation`

Moving Java Forward Faster



OpenJDK: New Release Model

- A new version of the JDK will be released every six months
  - March and September
  - Started this year
- OpenJDK development will be more agile
  - Previous target was a release every two years
    - Three and a half years between JDK 8 and JDK 9
- Features will be included only when ready
  - Targeted for a release when feature complete

JDK Version Numbering

- New, new scheme: JEP 322
  - `${FEATURE}.${INTERIM}.${UPDATE}.${PATCH}`

  - `FEATURE`: Was `MAJOR`, i.e. 10, 11, etc.
  - `INTERIM`: Was `MINOR`. Always zero, reserved for future use
  - `UPDATE`: Was `SECURITY`, but with different incrementing rule
  - `PATCH`: Emergency update outside planned schedule

Availability Of JDK Updates

- Oracle is switching to a long term support (LTS) model
  - LTS release every three years
  - JDK 8 is an LTS release
  - JDK 11 will be the next LTS release

A Tale of Two Binaries

- 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
  - No overlap of updates with previous version

Oracle Binaries

- Reduced platform support
  - No more 32-bit binaries
  - No more Arm binaries
  - Windows, Linux, Mac and Solaris SPARC only
    - All 64-bit

OpenJDK: More Open Source

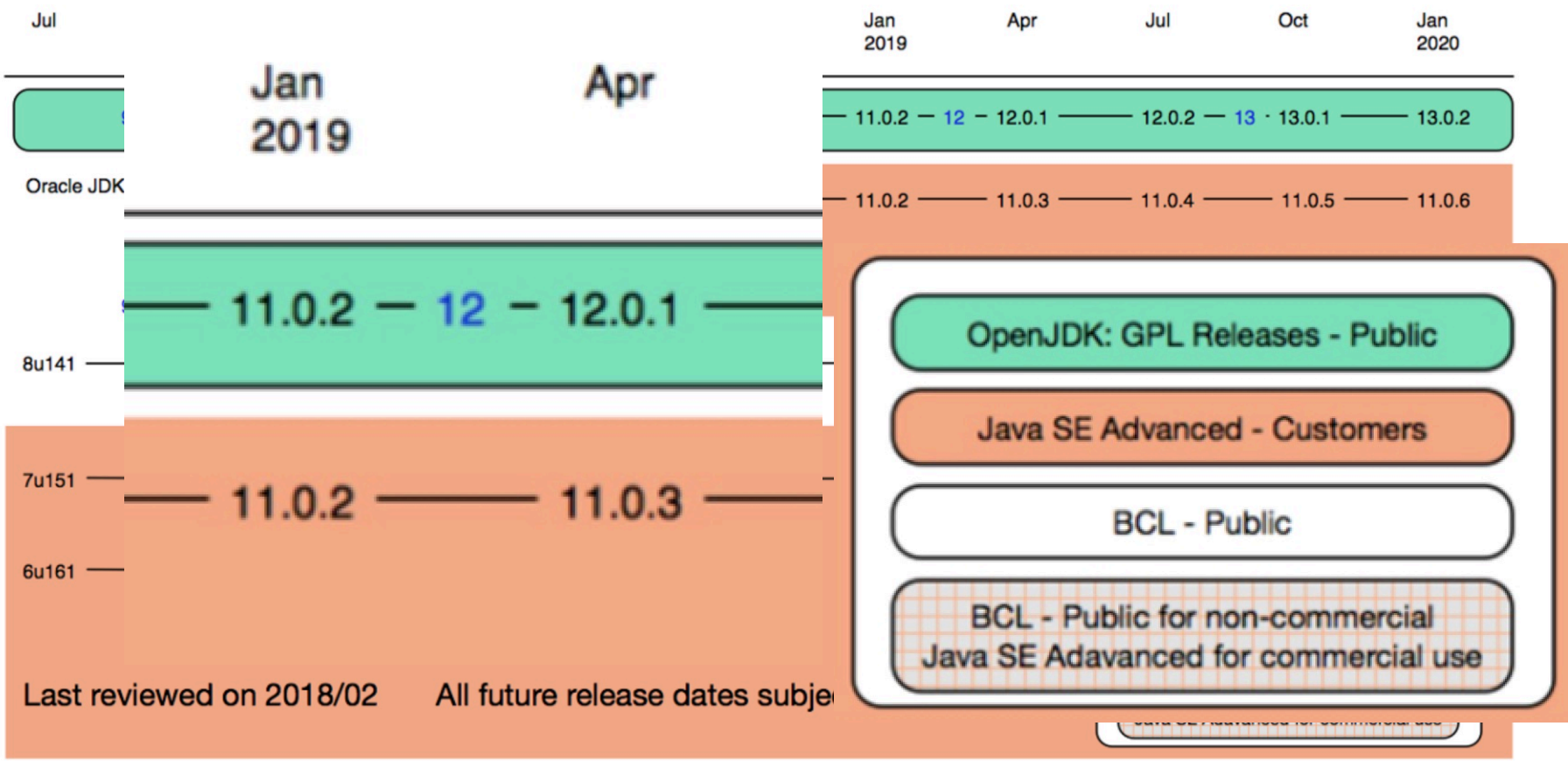
- Oracle will open-source closed-source parts of the JDK
  - Flight recorder
  - Mission control
  - Others
- The goal is for there to be no functional difference between an Oracle binary and a binary built from OpenJDK source
  - Targeted for completion late 2018

Eliminating Confusion

- An LTS release every three years
  - This does NOT mean 3 years of free updates
- LTS releases (from JDK 11) will ONLY be available for commercial support customers of Oracle
  - The only free JDK 11 and later will be OpenJDK binaries
- To continue to receive free updates to the JDK you MUST update your JDK **EVERY SIX MONTHS**
  - No more overlap of updates
  - Take some time to think about that

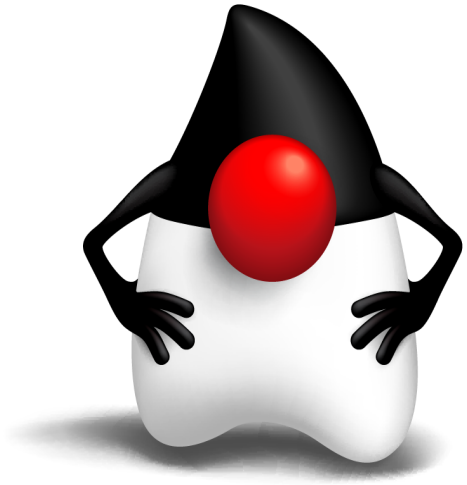
Desktop Java Users

New JDK Update Schedule



AZUL SYSTEMS

JDK 10



JDK 10 (JSR 383)

- Lead by Brian Goetz
- Expert Group: Oracle, IBM, Red Hat, SAP and Azul (Me)
- Released 20/3/18
  - Six month cadence does work
- 109 new features and APIs

Local Variable Type Inference

- JEP 286
  - Simple use cases

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

  - Style guidelines available
  - Good (appropriate use of variable name)

```
var stringList = List.of("a", "b", "c");
```

  - Poor

```
var itemQueue = new PriorityQueue<>();
```

    - Infered type is `PriorityQueue<Object>`

JDK 10: JEPs

- JEP 307: Parallel Full GC for G1
  - Still a full GC with potentially big pauses
- JEP 310: Application Class-Data Sharing
  - Previous commercial Oracle feature
- JEP 317: Experimental Java-based JIT compiler (Graal)
- JEP 319: Root Certificates
  - default set of root CA certificates
- JEP 296: Consolidate JDK forests into single repo
  - Eight repos becomes one

JDK 10: JEPs

- JEP 316: Heap allocation on alternative devices
  - NV-RAM with same semantics as DRAM
- JEP 313: Remove `javah` tool
  - Same functionality through `javac`
- JEP 304: Garbage Collector Interface (Internal JVM)
  - Easier to add new algorithms
- JEP 312: Thread-Local Handshakes
  - Execute callbacks on threads without performing a global VM safepoint

JDK 10: APIs

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

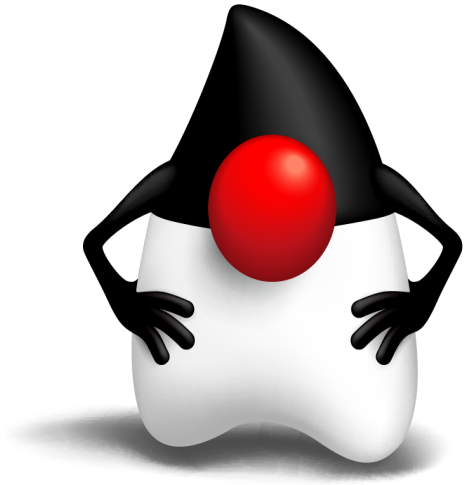
JDK 10: Miscellaneous

- `XMLInputFactory.newFactory()` de-deprecated (precated?)
- `com.sun.security.auth` package
  - Six deprecated classes removed
- `java.lang.SecurityManager`
  - One deprecated field and seven methods removed
- JVM now more Docker container aware
  - Uses container CPU count and memory size

Command Line Flags

- JDK 9 removed 50 `-XX` options
- JDK 10 removes 366 `-XX` options
- Also `-d32`, `-d64` and 5 `-X` options
  - `-Xoss`
  - `-Xsqnopause`
  - `-Xoptimize`
  - `-Xboundthreads`
  - `-Xusealtsigs`

JDK 11



JDK 11 (JSR 384)

- Spec lead and EG unchanged
  - Working on more of a rolling JSR
- Early draft review and builds available
- Scheduled GA: 25th September
- Designated an Oracle LTS release
  - Only for commercial customers
  - OpenJDK GPLv2 with CPE binaries will only have updates for six months

JDK 11: [Current] JEPs

- JEP 309: Dynamic Class-file constants
  - Like invokedynamic but for class-file constants
  - Bootstrap method used at runtime, not compile time
  - Useful for compiler and language developers
- JEP 318: Epsilon garbage collector
  - Which doesn't collect garbage!

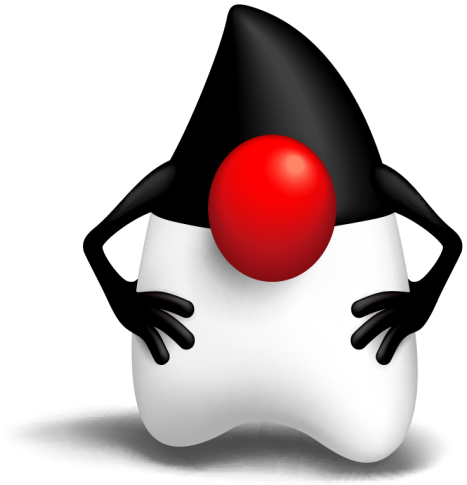
JDK 11: [Current] JEPs

- JEP 320: Remove CORBA and Java EE modules
  - The power of a module system
- JEP 321: HTTP client
  - HTTP/2 support
  - Incubating module from JDK 9 standardised
- JEP 323: Local variable syntax for Lambda parameters
  - Extending JEP 286 in JDK 10
  - `(var x, var y) -> x.process(y)`

JDK 11: New APIs

- Currently 36 new APIs
  - `InputStream nullInputStream()`
  - `Writer nullWriter()`
  - `Path of(String, String ...)`

Longer Term JDK Futures



OpenJDK Projects

- Amber
  - Simplifying syntax
- Valhalla
  - Value types and specialised generics
- Loom
  - Continuations and fibres
- Metropolis
  - The JVM re-written in Java
- Panama
  - FFI replacement for JNI

Project Amber

- JEP 286: Local variable type inference
  - Delivered in JDK 10
- JEP 323: Local variable syntax for Lambdas
  - Targeted for JDK 11

Project Amber

- JEP 301: Enhanced enums
  - Generic enums with type parameters
- JEP 302: Lambda leftovers
  - Single underscore for unused parameters
- JEP 326: Raw string literals
  - Use single backquote
  - `` `c:\Users\simon` ``

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();
}
```

JEP 325: Switch Expressions

```
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);
};
```

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
  - Too much interaction with operating system
  - Need a lighter weight approach

Project Loom

- 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

Project Metropolis

- Run Java on Java
  - Rewrite most of the JVM in Java
- Use the Graal compiler project as significant input
- Easier ports to new platforms
  - Less native code to modify and compile
- Faster new features on front-end
  - Easier to write Java than C++
- Performance is an issue to be explored and resolved
  - AOT compiler in JDK 9 is the start of this

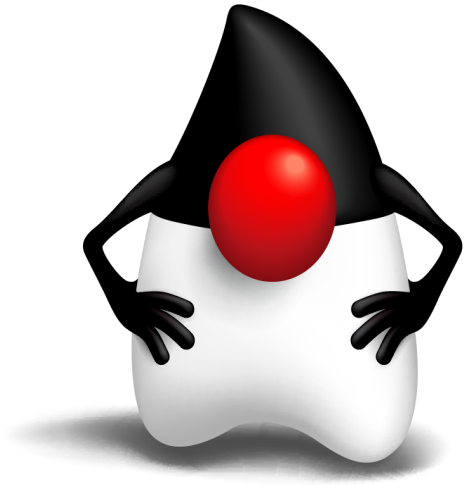
Azul's Zulu Java

Zulu Java

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

www.azul.com/downloads/zulu

Summary



Java Continues Moving Forward

- Faster Java releases
  - Feature release every 6 months
  - Quicker access to new features
  - Access to free updates is a consideration
- Lots of ideas to improve Java
  - Value types, fibres, syntax improvements
  - Many smaller things
- Helping Java to remain the most popular development platform on the planet!

Thank you!

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava