

# 55 New Features In JDK 9

© Copyright Azul Systems 2015

**Simon Ritter**

Deputy CTO, Azul Systems

[azul.com](http://azul.com)



@speakjava

# Major Features



# 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
    - `--add-exports`
    - `--add-opens`
- Modular source code (JEP 201)
  - JDK source code re-organised to support modules

# 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

# Reversing Encapsulation

- "The Big Kill Switch" to turn off encapsulation
  - `--illegal-access`
    - `permit`: Warning for first use of an encapsulated API
    - `warn`: Warning for every use of an encapsulated API
    - `debug`: Warning and stack trace for every use
    - `deny`: No access to encapsulated APIs

# Reversing Encapsulation

- Allowing direct access to encapsulated APIs

- `--add-exports`

- `--add-exports java.management/com.sun.jmx.remote.internal=mytest`

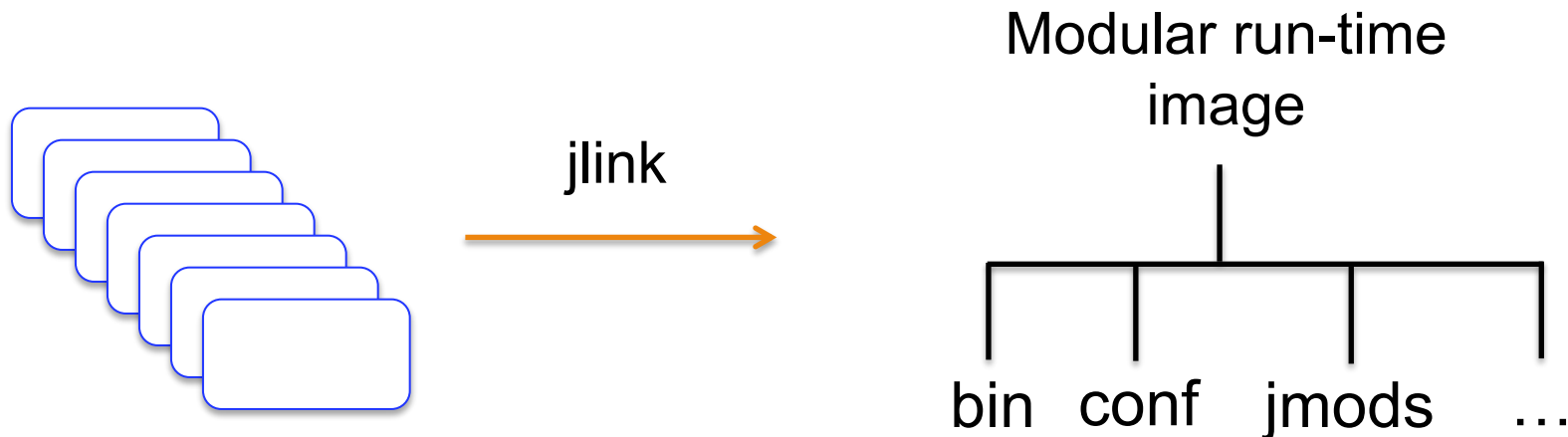
- `--add-exports java.management/sun.management=ALL-UNNAMED`

- Allowing reflective access to encapsulated APIs

- `--add-opens`

- `--add-opens java.base/java.util=ALL-UNNAMED`

# 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.app --output myimage
```

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

```
java.base@9
```

```
java.logging@9
```

```
java.sql@9
```

```
java.xml@9
```

```
com.azul.app@1.0
```

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

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



# Factory Methods For Collections (JEP 269)

- Static factory methods on List, Set and Map interfaces
  - Create compact, immutable instances
  - 0 to 10 element overloaded versions
  - Plus varargs version for arbitrary number of elements

```
Set<String> set = new HashSet<>();  
set.add("a");  
set.add("b");  
set.add("c");  
set = Collections.unmodifiableSet(set);  
  
Set<String> set = Set.of("a", "b", "c");
```

# Stream Enhancements



- `dropWhile()/takeWhile()`
  - Like `skip/limit` but uses `Predicate` rather than number
- Improved `iterate`
  - Enables a stream to be more like a for loop
- `Parallel Files.lines()`
  - Memory mapped, divided on line break boundary
- Stream from `Optional`
  - Stream of zero or one element

# Multi-Release Jar Files (JEP 238)

- Multiple Java release-specific class files in a single archive
- Enhance jar tool to create multi-release files
- Support multi-release jar files in the JRE
  - Classloaders
  - JarFile API
- Enhance other tools
  - javac, javap, jdeps, etc.
- Also, modular jar files



# REPL: jshell (JEP 222)

- Read-Eval-Print Loop
  - Simple prototyping

```
[MBP > ./jshell
| Welcome to JShell -- Version 9-ea
| Type /help for help

[-> ArrayList<String> l
| Added variable l of type ArrayList<String>

[-> l.add("hello")
| java.lang.NullPointerException thrown
|     at (#10:1)

[-> l = new ArrayList<>()
| Variable l has been assigned the value []

[-> l.add("hello")
| Expression value is: true
|     assigned to temporary variable $4 of type boolean

[-> l
| Variable l of type ArrayList<String> has value [hello]

->
```

# Concurrency Updates (JEP 266)

- Reactive streams publish-subscribe framework
- Asynchronous, non-blocking
- Flow
  - Publisher, Subscriber, Processor, Subscription
- SubmissionPublisher utility class
  - Asynchronously issues items to current subscribers
  - Implements Flow.Processor



# Concurrency Updates (JEP 266)

- `CompletableFuture` additions
  - Delays and timeouts
  - Better support for sub-classing
  - New static utility methods
    - `minimalCompletionStage`
    - `failedStage`
    - `newIncompleteFuture`
    - `failedFuture`



# Enhanced Deprecation (JEP 277)

- We have `@deprecated` and `@Deprecated`
  - Used to cover too many situations
- New methods in `Deprecated` annotation
  - `boolean forRemoval()`
    - Will this ever be removed?
  - `String since()`
    - JDK Version when this was deprecated
- Several `@Deprecated` tags added
  - `java.awt.Component.{show(),hide()}` removed
- `jdeprscan` command to produce report



# Milling Project Coin (JEP 213)

- Single underscore no longer valid as identifier
  - Ready for use in Lambdas
- Private methods in interfaces
  - Multiple inheritance of behaviour makes this logical
- Effectively final variables in try-with-resources
  - Variables from outside try block
- Allow @SafeVarargs on private instance methods
  - In addition to constructors, final and static methods
- Diamond operator with anonymous classes
  - Extending type inference further





# Standards



# Updating To Relevant Standards

- Unicode 7.0/8.0 (JEP 227/267)
  - 7.0: 2,834 new characters
  - 8.0: 7,716 new characters } seriously!
- PKCS12 key stores by default (JEP 229)
  - Move from JKS to PKCS12
- HTML5 javadocs (JEP 224)
  - Flashier documentation
- SHA 3 hash algorithms (JEP 287)
  - Keeping ahead of the hackers



Unicode

HTML



# Smaller Features

- UTF-8 property files (JEP 226)
  - ResourceBundle API updated to load these files
- DRBG-Based SecureRandom implementations (JEP 273)
  - Deterministic Random Bit Generator
- XML Catalog API (JEP 268)
  - Supports OASIS XML catalog API v1.1
  - For use with JAXP

# Inside The JVM



# Default Collector: G1 (JEP 248)

- G1 now mature in development
- Designed as low-pause collector
- Concurrent class unloading (JEP 156) JDK8u40
  - Useful enhancement to improve G1
- Still falls back to full compacting collection
  - Pause times proportional to heap size
  - Use Zing from Azul for truly pauseless



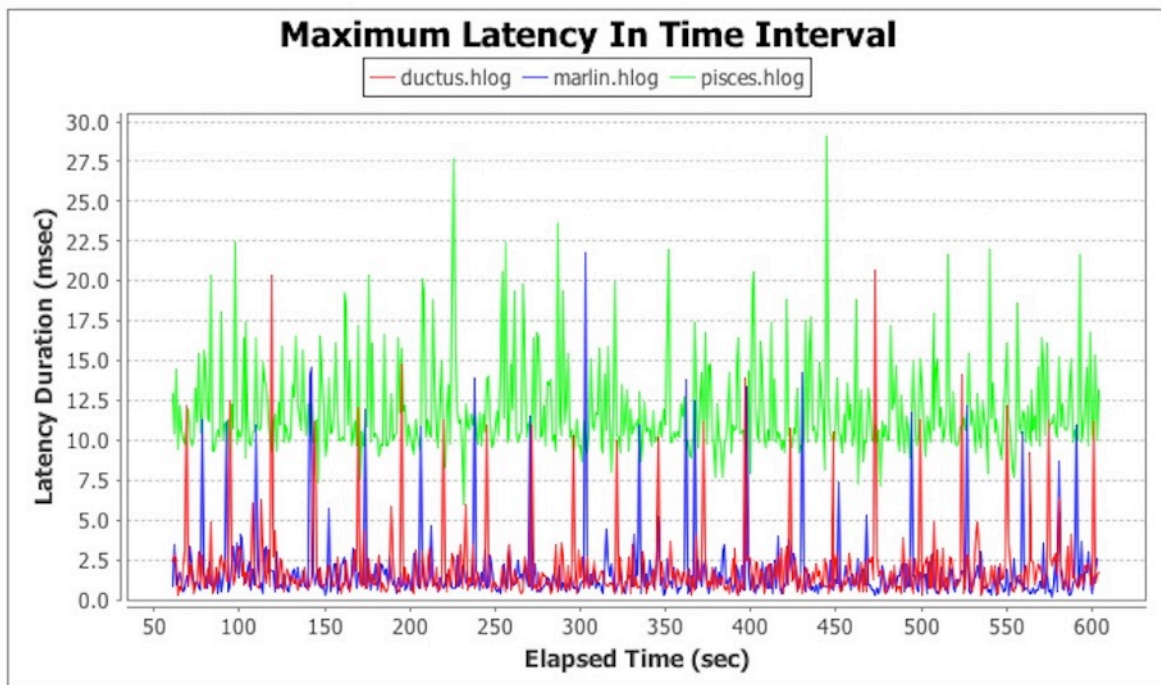
# Better String Performance



- Compact strings (JEP 254)
  - Improve the space efficiency of the `String` class
  - Not using alternative encodings
- Store interned strings in CDS archive (JEP 250)
  - Share `String` and `char[]` objects between JVMs
- Indify `String` concatenation (JEP 280)
  - Change from static `String`-concatenation bytecode sequence to `invokedynamic`
  - Allow future performance improvements

# Marlin Graphics Renderer (JEP 265)

- Replaces Pisces open-source renderer
  - Comparable performance to closed-source Ductus



# Smaller Features

- Improve contended locking (JEP 143)
  - Field reordering/cache line alignment
- Leverage CPU instructions for GHASH and RSA (JEP 246)
  - Up to 150x better performance (for specific tests)
- Update JavaFX to newer version of GStreamer (JEP 257)
  - Media class
  - Better security, stability and performance



# Smaller Features

- Segmented Code Cache (JEP 197)
  - Separate non-method, profiled and non-profiled code
- Unified JVM logging (JEP 158)
  - Common logging system for all components of JVM
- Unified GC logging (JEP 271)
  - Re-implement GC logging using unified JVM logging
  - Many command line options changed



# Specialised



# Spin-Wait Hints (JEP 285)

- Proposed by Azul
  - We rock!
- A new method for Thread class
  - `onSpinWait()`
- Enables the x86 PAUSE instruction to be used from Java code
  - If available
  - Ignored otherwise
  - Improved performance for things like Disruptor



# Variable Handles (JEP 193)

- Replacement for parts of `sun.misc.Unsafe`
- Fence operations
  - Fine grained memory control
  - Atomic operations on object fields and array elements
- `VarHandle`
  - `compareAndExchange()`, `compareAndSet()`
  - `getAndAdd()`, `getAndSet()`
  - `acquireFence()`, `releaseFence()`



# AOT Compilation (JEP 295)

- Experimental feature in JDK 9
  - `-XX:+UnlockExperimentalVMOptions`
  - Only the `java.base` module has AOT version
- `jaotc` command
  - `jaotc --output libMyStuff.so MyStuff.jar`
- JVM uses AOT code to replace interpreted
  - Can recompile with C1 to collect further profiling data
  - Recompile with C2 for optimum performance

# Smaller Features

- Compiler control (JEP 165)
  - Control of C1/C2 JIT, not javac
  - Directive file
  - Runtime changes via jcmd
- Process API updates (JEP 102)
  - Native process (Process/ProcessHandle)
  - More information: pid, arguments, start time, CPU usage, name
  - Control subject to security manager permissions



# Housekeeping



# New Version String Format (JEP 223)

- Old

- Limited update release/Critical patch update (CPU)
- Download: Java SE 8u131, `java -version: jdk1.8.0_131`
- Which has more patches, JDK 7u55 or JDK 7u60?

- New

- ~~– JDK \$MAJOR.\$MINOR.\$SECURITY \$PATCH~~
- ~~– Easy to understand by humans and apps~~
- ~~– Semantic versioning~~





# New Version String Format (JEP 223)

- Old
  - Limited update release/Critical patch update (CPU)
  - Download: Java SE 8u131, `java -version: jdk1.8.0_131`
  - Which has more patches, JDK 7u55 or JDK 7u60?
- New, new
  - ~~– JDK \$YEAR.\$MONTH~~
  - ~~– New 6 month release cadence, Mar/Sep~~
  - ~~– Next version would be JDK 18.3, then 18.9~~



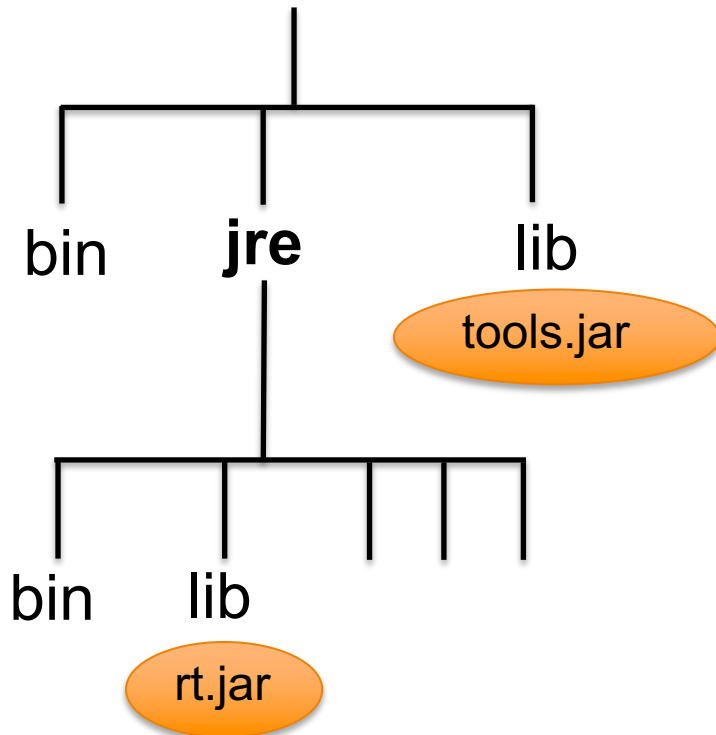
# New Version String Format (JEP 322)

- Old
  - Limited update release/Critical patch update (CPU)
  - Download: Java SE 8u131, `java -version: jdk1.8.0_131`
  - Which has more patches, JDK 7u55 or JDK 7u60?
- New, new, new
  - `JDK $FEATURE.$INTERIM.$UPDATE.$PATCH`
  - Back to JDK 10 and JDK 11
    - Or JDK 10.0.1.1

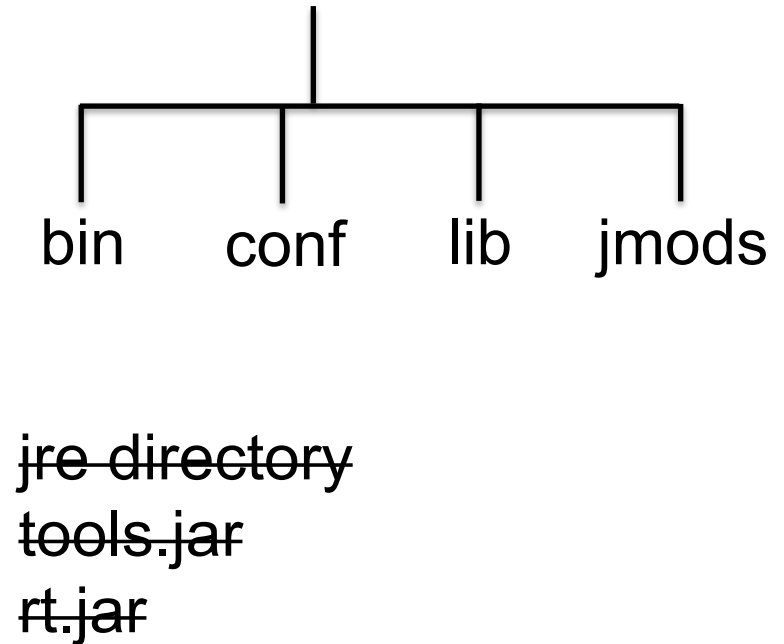


# JDK/JRE File Structure (JEP 220)

## Pre-JDK 9



## JDK 9



# Smaller Features

- Searchable API documentation (JEP 225)
  - Finally! Java API docs enter the 21<sup>st</sup> century
- Annotations pipeline 2.0 (JEP 217)
  - Repeating, type and Lambda annotations in JDK 8
  - Redesign of javac annotation pipeline
- Parser API for Nashorn (JEP 236)
  - API for Nashorn abstract tree syntax
  - Nashorn implements ECMAScript 5.1 spec.



# General Clean Up

- Disable SHA-1 certificates (JEP 288)
  - Mostly
    - In some situations SHA-1 certs. will still be accepted
- Deprecate the Applet API (JEP 289)
  - Not many people still use this



# Removed From JDK 9

- Six deprecated APIs (JEP 162)
  - {Add, Remove}ActionListener
  - Pack200, Unpack200 and LogManager
- `com.sun.security.auth.callback.DialogCallbackHandler`
  - Part of JAAS
- JRE version selection command line option (JEP 231)
  - `-version` ***release*** no longer accepted
  - `-version` still works
- Demos and samples (JEP 298)

# Removed From JDK 9

- JVM TI hprof agent (JEP 240)
  - Only ever intended as a demo of JVM TI
  - Useful features now in other tools (like jmap)
- Remove the jhat tool (JEP 241)
  - Experimental tool added in JDK 6
  - Unsupported
  - Better heap visualisation tools available

# Removed GC Options (JEP 214)

- Deprecated in JDK 8 (JEP 173)

DefNew + CMS	:	-XX:-UseParNewGC -XX:+UseConcMarkSweepGC
ParNew + SerialOld	:	-XX:+UseParNewGC
ParNew + iCMS	:	-Xincgc
ParNew + iCMS	:	-XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC
DefNew + iCMS	:	-XX:+CMSIncrementalMode -XX:+UseConcMarkSweepGC -XX:-UseParNewGC
CMS foreground	:	-XX:+UseCMSCompactAtFullCollection
CMS foreground	:	-XX:+CMSFullGCsBeforeCompaction
CMS foreground	:	-XX:+UseCMSCollectionPassing



# Summary



# JDK 9

- Big new feature is modularity
  - Covers numerous different areas
  - Modules, jlink, etc.
- Smaller developer features
  - New APIs for streams
  - Reactive API
  - REPL/jshell
- Many smaller performance/standards features
- Time to start testing, if you're not already

# Zulu Java

- Azul's binary distribution of OpenJDK
  - Passes all TCK tests
  - Multi-platform (Windows, Linux, Mac)
  - FREE!
    - Happy to sell you support, including older versions
- JDK 6, 7, 8 and 9

[www.zulu.org/download](http://www.zulu.org/download)

# Thank You

© Copyright Azul Systems 2015

**Simon Ritter**

Deputy CTO, Azul Systems

[azul.com](http://azul.com)

© Copyright Azul Systems 2017



@speakjava