55 New Features In JDK 9

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava

Major Features

Java Platform Module System JSR 376 Public Review Reconsideration Ballot

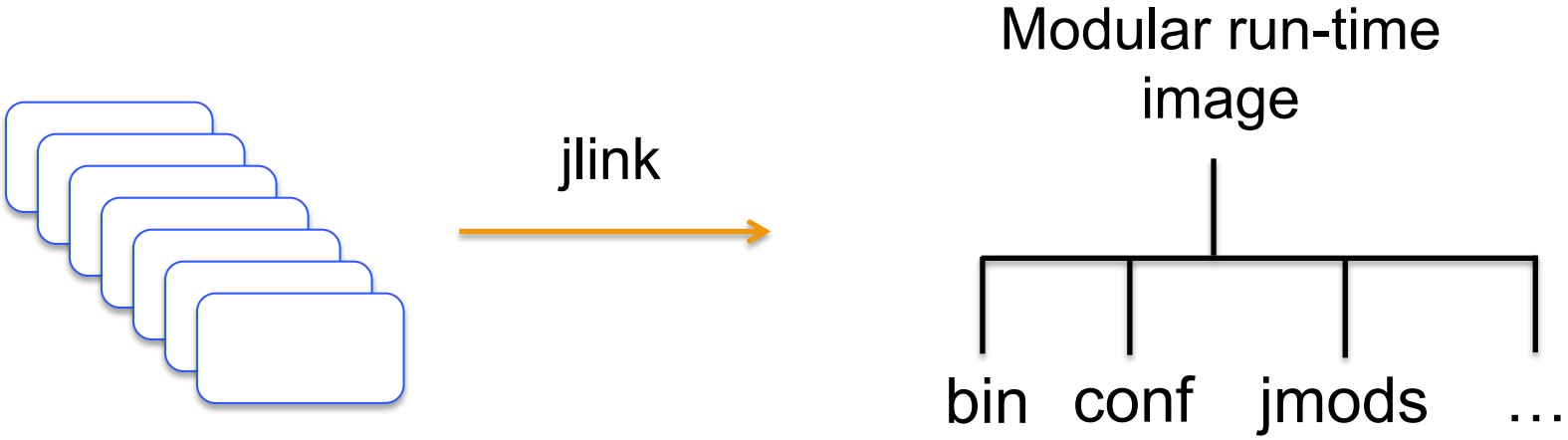
Modularity/Jigsaw

- Standard class libraries modularised (JEP 220)
  - JDK now 94 modules
- 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
  - Not the module system

Modularity/Jigsaw

- Compatability is an issue
  - Mostly for libraries and frameworks
  - Which most people use
  - Override encapsulation of private APIs
- Leave everything on classpath to start with
- “Big kill switch” now has options
  - Turns off encapsulation completely by default
  - `--illegal-access={permit,warn,debug,deny}`

jlink: The Java Linker (JEP 282)



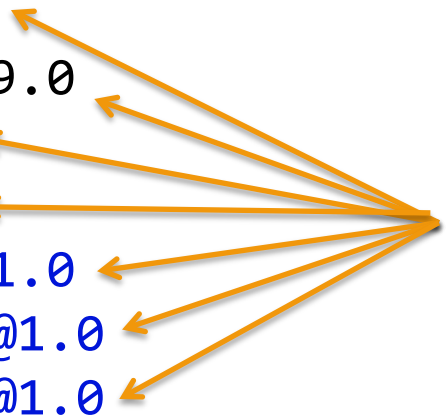

```
$ jlink --modulepath $JDKMODS \
  --addmods java.base –output myimage

$ myimage/bin/java –-list-modules
java.base@9.0
```

jlink: The Java Linker (JEP 282)

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

$ myimage/bin/java –-list-modules
java.base@9.0
java.logging@9.0
java.sql@9.0
java.xml@9.0
com.azul.app@1.0
com.azul.zoop@1.0
com.azul.zeta@1.0
```



Version numbering for information purposes only

"It is not a goal of the module system to solve the version-selection problem"

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 arbitary 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

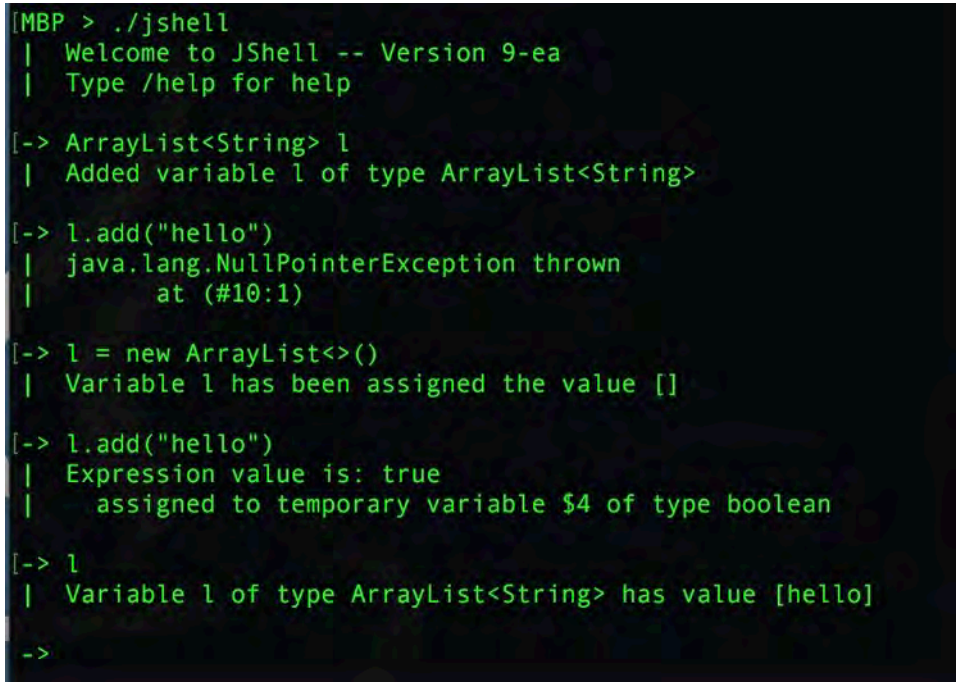
- `dropWhile()/takeWhile()`
  - Like `skip/limit` but uses `Predicate` rather than number
- Improved `iterate`
  - Enables a stream to be more like a `for` loop
- Parallel `Files.lines()`
  - Memory mapped, divided on line break boundary
- Stream from `Optional`
  - Stream of zero or one element

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



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

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)

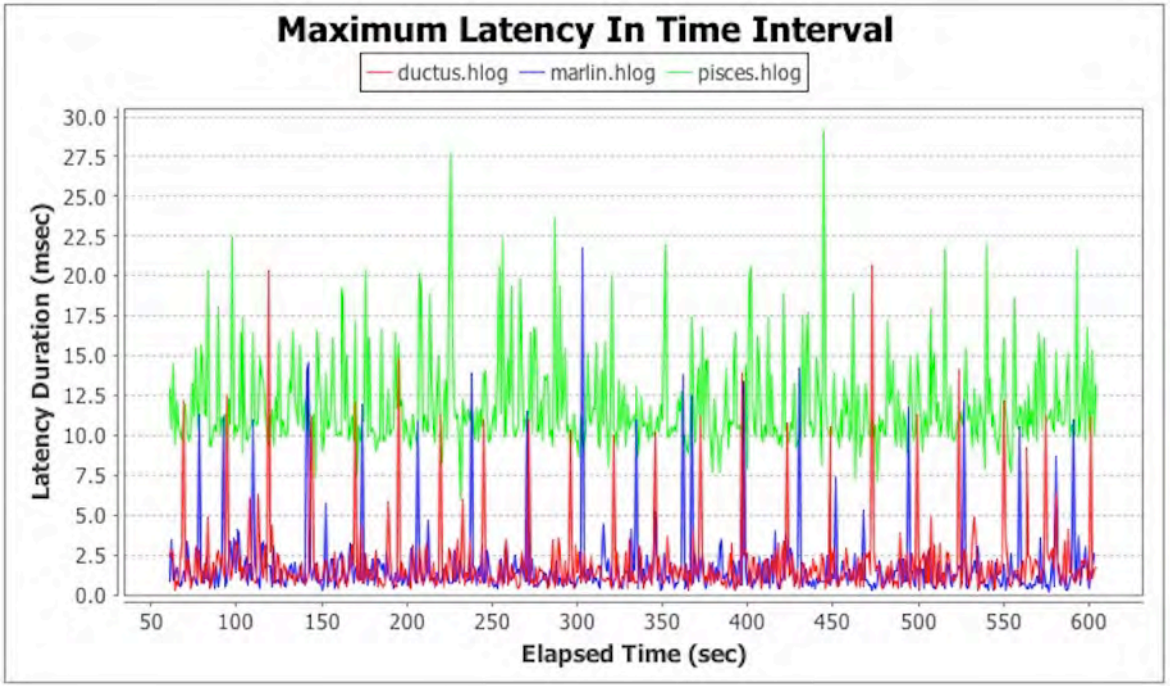
- G1 now mature in development
- Designed as low-pause collector
- Concurrent class unloading (JEP 156) JDK8u40
  - Useful enhancement to improve G1
- Still falls back to full compacting collection
  - Pause times proportional to heap size
  - Use Zing from Azul for truly pauseless

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)
  - 34-150x better performance (for certain 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()`

Enhanced Method Handles (JEP 274)

- Support for
  - loops
  - `try/finally` blocks
- Better argument handling
  - Spreading
  - Collection
  - Folding
- More lookup functions
  - Non-abstract methods in interfaces, classes

Smaller Features

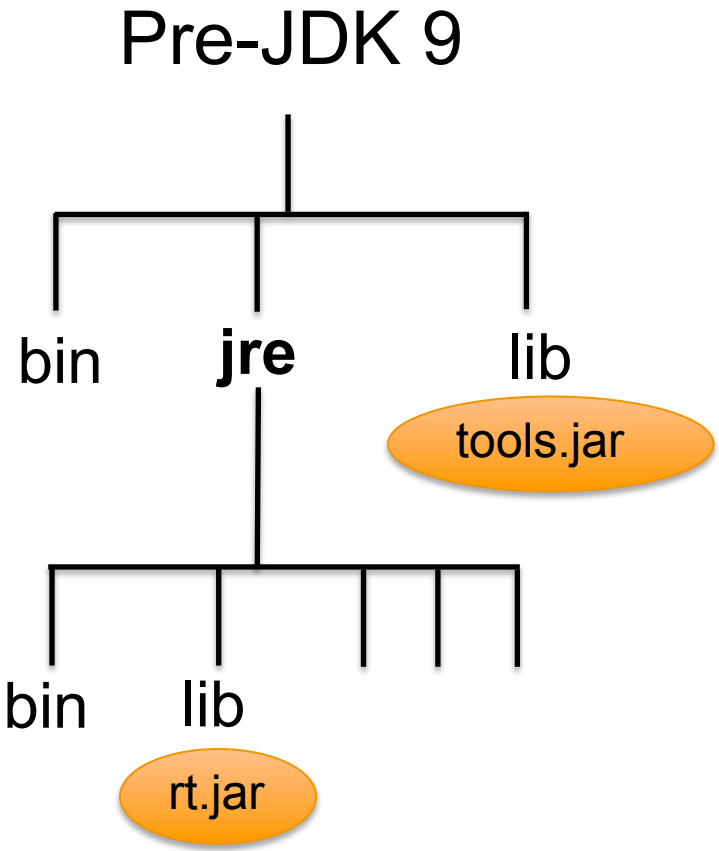
- Compiler control (JEP 165)
  - Control of C1/C2 JIT, not javac
  - Directive file
  - Runtime changes via `jcmd`
- Process API updates (JEP 102)
  - Native process (`Process`/`ProcessHandle`)
  - More information: pid, arguments, start time, CPU usage, name
  - Control subject to security manager permissions

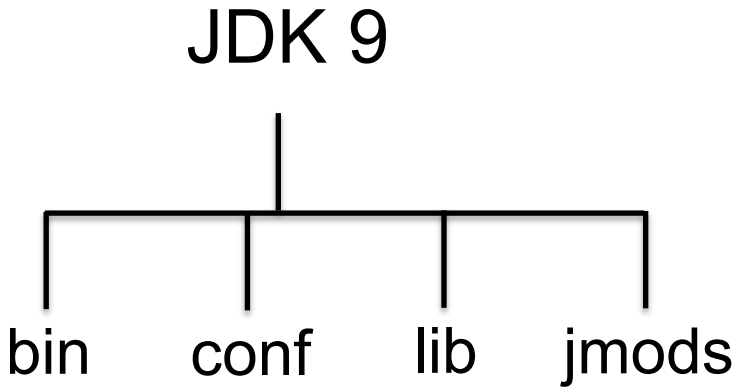
Housekeeping

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

JDK/JRE File Structure (JEP 220)





~~jre directory~~
~~tools.jar~~
~~rt.jar~~

Smaller Features

- Searchable API documentation (JEP 225)
  - Finally! Java API docs enter the 21st 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)
  - Out-of-date, unmaintained

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
```

Incubator Modules (JEP 11)

- Develop APIs without making them part of the standard
  - At least not straight away
- Allow developers to “kick the tyres”
  - Not always possible to get a new API right first time
- Move from incubator to full module
  - Becomes part of the standard
- JDK 9 only has one incubator: HTTP/2 (JEP 110)
- Some concerns about fragmentation

```
--do-not-resolve-by-default
```

Summary

JDK 9

- Big new feature is modularity
  - Covers numerous different areas
  - Modules, `jink`, 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 (Early Access)

www.zulu.org/download

Thank You

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava

Thank You

Survey:
bit.ly/AZULJDK9

Simon Ritter
Deputy CTO, Azul Systems
azul.com

@speakjava