Java 9: Exploring New Features

Paul Webber
2017-04-18

Java 9 Feature Set

- JDK 9 JEPS
- JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)
- JEP 269: Convenience Factory Methods for Collections
- JEP 277: Enhanced Deprecation

JDK Enhancement Proposals (JEPs) for JDK 9

- Store Interned Strings in CDS Archives
- Improve Contended Locking
- Compact Strings
- Improve Secure Application Performance
- Leverage CPU Instructions for GHASH and RSA
- Tiered Attribution for javac
- Javadoc Search
- Marlin Graphics Renderer
- HiDPI Graphics on Windows and Linux
- Enable GTK 3 on Linux
- Update JavaFX/Media to Newer Version of GStreamer

Behind the scenes

- Module System
- The Modular JDK
- Modular Source Code
- Modular Run-Time Images
- Encapsulate Most Internal APIs
- jlink: The Java Linker
- Enhanced Deprecation
- Stack-Walking API
- Convenience Factory Methods for Collections
- Platform Logging API and Service
- jshell: The Java Shell (Read-Eval-Print Loop)
- Compile for Older Platform Versions
- Multi-Release JAR Files
- Platform-Specific Desktop Features
- TIFF Image I/O\
- Multi-Resolution Images

New functionality

- Process API Updates
- Variable Handles
- Spin-Wait Hints
- Dynamic Linking of Language-Defined Object Models
- Enhanced Method Handles
- More Concurrency Updates
- Compiler Control

Specialized

- HTTP 2 Client
- Unicode 8.0
- UTF-8 Property Files
- Datagram Transport Layer Security (DTLS)
- OCSP Stapling for TLS
- TLS Application-Layer Protocol Negotiation Extension
- SHA-3 Hash Algorithms
- DRBG-Based SecureRandom Implementations
- Create PKCS12 Keystores by Default
- Merge Selected Xerces 2.11.0 Updates into JAXP
- XML Catalogs
- HarfBuzz Font-Layout Engine
- HTML5 Javadoc

New standards

- Parser API for Nashorn
- Prepare JavaFX UI Controls & CSS APIs for Modularization
- Modular Java Application Packaging
- New Version-String Scheme
- Reserved Stack Areas for Critical Sections
- Segmented Code Cache
- Indify String Concatenation
- Unified JVM Logging
- Unified GC Logging
- Make G1 the Default Garbage Collector
- Use CLDR Locale Data by Default
- Validate JVM Command-Line Flag Arguments
- Java-Level JVM Compiler Interface
- Disable SHA-1 Certificates
- Simplified Doclet API
- Deprecate the Applet API
- Process Import Statements Correctly
- Annotations Pipeline 2.0
- Elide Deprecation Warnings on Import Statements
- Milling Project Coin
- Filter Incoming Serialization Data

Housekeeping

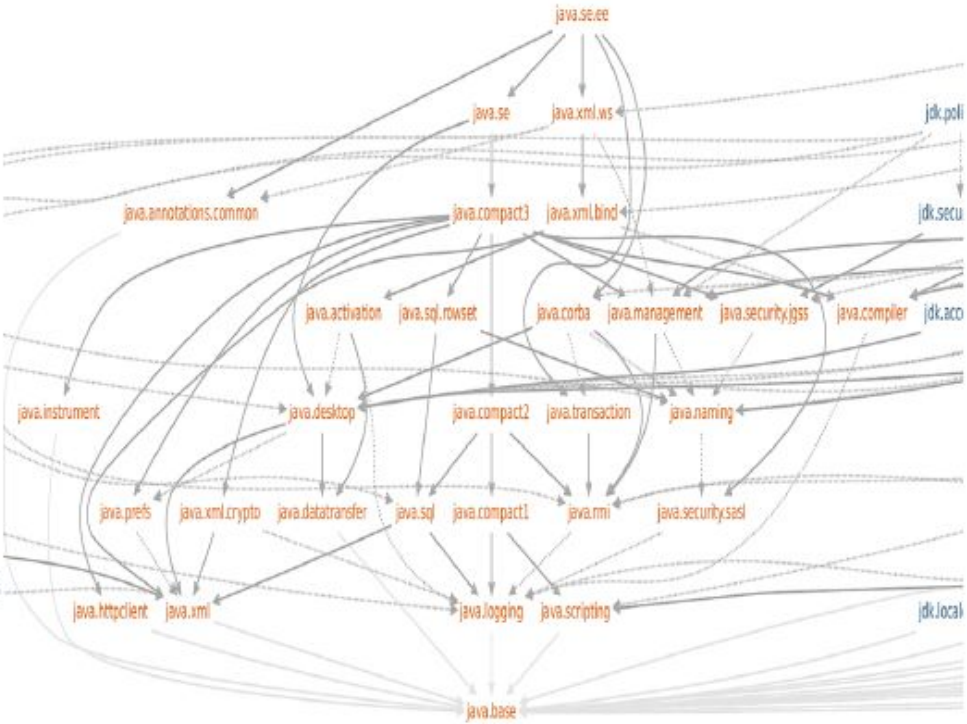
- Remove GC Combinations Deprecated in JDK 8
- Remove Launch-Time JRE Version Selection
- Remove the JVM TI hprof Agent
- Remove the jhat Tool

Gone

Project Jigsaw

Modularize the Java Platform

- JEP 261: Module System
- JEP 200: The Modular JDK
- JEP 201: Modular Source Code
- JEP 220: Modular Run-Time Images
- Plus
  - JEP 260: Encapsulate Most Internal APIs
  - JEP 282: jlink: The Java Linker



Going Away

- 214: Remove Deprecated GC Combinations
- 231: Remove Launch-Time JRE Version Selection
- 240: Remove the JVM TI hprof Agent
- 241: Remove the jhat Tool
- 260: Encapsulate Most Internal APIs
- 289: Deprecate the Applet API
- 298: Remove Demos and Samples

Possible Unexpected Behaviour

- 158: Unified JVM Logging
- 223: New Version-String Scheme
- 245: Validate JVM Command-Line Flag Arguments
- 248: Make G1 the Default Garbage Collector
- 271: Unified GC Logging

What is JShell - JEP 222

JEP 222: jshell: The Java Shell (Read-Eval-Print Loop)

Owner: Robert Field

Goals:

The JShell API and tool will provide a way to interactively evaluate declarations, statements, and expressions of the Java programming language within the JShell state. The JShell state includes an evolving code and execution state. To facilitate rapid investigation and coding, statements and expressions need not occur within a method, expressions need not have side-effects, variables need not occur within a class, and methods need not occur within a class or interface.

Most Recent Tutorial: http://cr.openjdk.java.net/~rfield/tutorial/JShellTutorial.html

What is JShell

- Tool providing a dynamic interaction with the Java™ programming language
- Read-Evaluate-Print Loop (REPL) for the Java™ platform
  - Type in a snippet of Java code, see the results
- Deeply integrated with JDK tool-set
  - Stays current and compatible
- Also, an API for use within other applications

JShell Command Line Parameters

Parameter	Description
--class-path <path>	Specify where to find user class files
--module-path <path>	Specify where to find application modules
--add-modules <module>(,<module>)*	Specify modules to resolve, or all modules on the module path if <module> is ALL-MODULE-PATHs
--startup <file>	One run replacement for the start-up definitions
--no-startup	Do not run the start-up definitions
--feedback <mode>	Specify the initial feedback mode. The mode may be predefined (silent, concise, normal, or verbose) or previously user-defined
-q	Quiet feedback. Same as: --feedback concise
-s	Really quiet feedback. Same as: --feedback silent
-v	Verbose feedback. Same as: --feedback verbose
-J<flag>	Pass <flag> directly to the runtime system. Use one -J for each runtime flag or flag argument
-R<flag>	Pass <flag> to the remote runtime system. Use one -R for each remote flag or flag argument
-C<flag>	Pass <flag> to the compiler. Use one -C for each compiler flag or flag argument
--version	Print version information and exit
--show-version	Print version information and continue
--help-extra, -X	Print help on non-standard options and exit

JShell Commands - Operations

Command	Description
/help [<command>|<subject>] ● intro ● shortcuts ● context	get information about jshell List commands
/? [<command>|<subject>]	same as /help
/exit	exit jshell
/save [-all|-history|-start] <file>	Save snippet source to a file
/open <file>	open a file as source input

JShell Commands - Lists

Command	Description
/history	history of what you have typed
/list [<name or id>|-all|-start]	list the source you have typed
/imports	list the imported items
/vars [<name or id>|-all|-start]	list the declared variables and their values
/methods [<name or id>|-all|-start]	list the declared methods and their signatures
/types [<name or id>|-all|-start]	list the declared types
/drop <name or id>	delete a source entry referenced by name or id
/edit <name or id>	edit a source entry referenced by name or id

JShell Commands - Environment

Command	Description
/env [-class-path <path>] [-module-path <path>] [-add-modules <modules>] …	view or change the evaluation context
/reset [-class-path <path>] [-module-path <path>] [-add-modules <modules>]...	reset jshell
/reload [-restore] [-quiet] [-class-path <path>] [-module-path <path>]...	reset and replay relevant history -- current or previous (-restore)
/set editor|start|feedback|mode|prompt|truncation|format …	set jshell configuration information
/!	re-run last snippet
/<id>	re-run snippet by id
/-<n>	re-run n-th previous snippet

JShell - Key Commands

Key	Action
Ctrl-r	Search backward through history
Ctrl-s	Search forward through history
Ctrl-x (	Start entering macro
Ctrl-x)	Finish macro
Ctrl-x e	Use macro
Ctrl-k	Kill (delete) the text from the cursor to the end of the line
Meta-d	Kill from the cursor to the end of the word
Ctrl-w	Kill from the cursor to the previous whitespace
Ctrl-y	Yank (paste) the most recently killed text into the line
Meta-y	After Ctrl-y, press to cycle through previously killed text

JShell - More Key Commands

Key	Action
Return	Enter the current line
Left-arrow	Move backward one character
Right-arrow	Move forward one character
Up-arrow	Move up one line (backward through history)
Down-arrow	Move down one line (forward through history)
Ctrl-a	Move to the beginning of the line
Ctrl-e	Move to the end of the line
Meta-b	Move backward a word
Meta-f	Move forward a word
Meta-y	After Ctrl-y, press to cycle through previously killed text

JShell Scripts

Script	Contents
DEFAULT	The default startup if one is not set, includes commonly needed import declarations
PRINTING	Defines JShell methods that redirect to the **print**, **println**, and **printf** methods in **PrintStream**
JAVASE	Imports all Java SE packages, this is big and will cause a noticeable startup delay

```
jshell> /open PRINTING
```

OR

```
Cmd: jshell --startup PRINTING
```

Testing Java Nuances

- Did you know that comparing autoboxed integers references which values are from range -128 to 127 (inclusive) returns true (they are cached)?

Integer i1 = 127

Integer i2 = 127

i1 == i2

Integer i1 = 128

Integer i2 = 128

i1 == i2

Testing Java Nuances

- Don’t forget basic type limitations!

  x = 123456

  y = 123456

  What Type does x*y return?

JEP 269 – Convenience Factory Methods for Collections

Owner: Stuart Marks

Goals:

- Provide static factory methods on the collection interfaces that will create compact, unmodifiable collection instances. The API is deliberately kept minimal.

JEP 269 - Examples

```
// Java 8

List<String> stringList = Arrays.asList("a", "b", "c");
Set<String> stringSet = new HashSet<>(Arrays.asList("a", "b", "c"));
Map<String,Integer> stringMap = new HashMap<>();
stringMap.put("a",1);
stringMap.put("b",2);
stringMap.put("c",3);

// Java 9

List<String> stringList = List.of("a", "b", "c");
Set<String> stringSet = Set.of("a", "b", "c");
Map<String,Integer> stringMap = Map.of("a", 1, "b", 2, "c", 3);
```

JEP 269 - Map With Arbitrary Number of Pairs

```
Map<String, String> tokens = Map.ofEntries(
        Map.entry("@", "AT"),
        Map.entry("|","VERTICAL_BAR"),
        Map.entry("#","HASH"),
        Map.entry("%","PERCENT"),
        Map.entry(":","COLON"),
        Map.entry("^","CARET"),
        Map.entry("&","AMPERSAND"),
        Map.entry("!","EXCLAM"),
        Map.entry("?","QUESTION"),
        Map.entry("$","DOLLAR"),
        Map.entry("::","PAAMAYIM_NEKUDOTAYIM"),
        Map.entry("=","EQUALS"),
        Map.entry(";","SEMICOLON")
);
```

JEP 277: Enhanced Deprecation

Owner: Stuart Marks

Goals:

- Provide better information about the status and intended disposition of APIs in the specification.
- Provide a tool to analyze an application's static usage of deprecated APIs.

Deprecation Key Concepts

- Deprecation is
  - Notification to developers that they should migrate their code away from the deprecated API
- Possible reasons
  - The deprecated API has something wrong with it
  - There’s a newer, better API that can be used instead
  - The deprecated API is going to be removed

History (1)

- Deprecation introduced in JDK 1.1 as a javadoc tag **@deprecated**
- Early on, massive wave of deprecations for widely varying reasons
  - dangerous APIs (Thread.destroy)
  - simple renames (AWT Component.show/hide => setVisible)
  - caused disruption, compiler warnings, fear, confusion
  - result: deprecations slowed drastically, no APIs removed
- Annotation **@Deprecated** introduced in Java 5
  - note capitalization
  - no change/clarification of semantics

History (2)

- Has anything ever been un-deprecated?
  - Yes! The System.getenv() API was in JDK 1.0
  - first sentence of its javadoc: “Obsolete.”
  - implementation: threw an exception unconditionally
  - officially deprecated in 1.1 (when deprecation was created)
  - fully implemented and un-deprecated in Java SE 5
- Has anything been newly introduced as deprecated?
  - Yes! Three overloads of javax.management.MBeanServer.deserialize()
  - introduced in Java SE 5, deprecated in Java SE 5!
  - JMX was a standalone JSR, with deprecated stuff, before integration into SE 5

History (3)

- Confusion continued through Java 8
  - “Don’t use deprecated APIs, since they might be removed”
  - “Sun/Oracle have never removed anything, and they never will”
- Wrong! Oracle will actually remove stuff in Java 9
  - java.util.logging.LogManager.add/removePropertyChangeListener
  - java.util.jar.Pack200.Packer/Unpacker.add/removePropertyChangeListener
  - main driver was modularity, to break dependencies on Java Beans, part of the java.desktop module
  - also: java.awt.Component.getPeer() removed
    - flawed API, referred to types outside of Java SE, e.g., ComponentPeer

Java 9 Deprecations So Far: forRemoval=false

- Boxed primitive constructors: new Integer(), new Boolean(), etc.
  - use Integer.valueOf(), Boolean.valueOf(), etc.
- Java.applet
  - applets and browser plugins are (slowly) on their way out
- java.util.Observer/Observable
  - anybody ever use these?

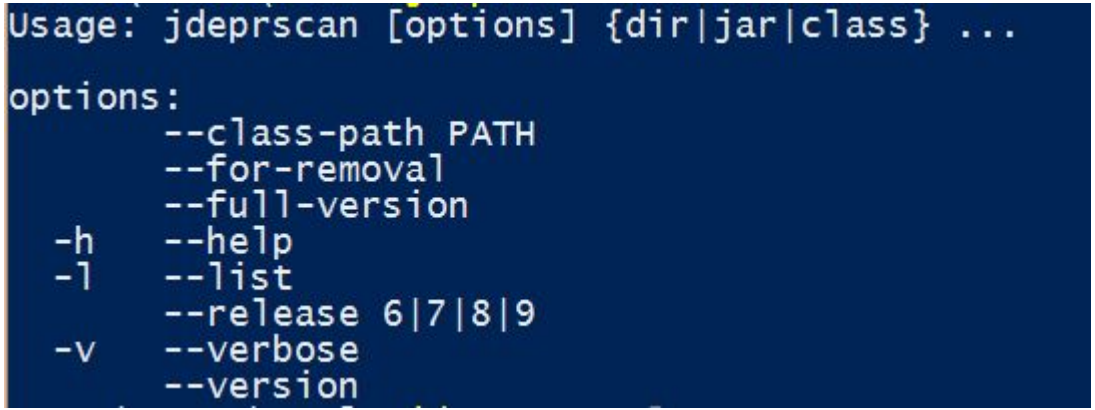
Java 9 Deprecations So Far: forRemoval=true

- Thread.destroy()
- Thread.stop(Throwable)
  - no-arg Thread.stop() still deprecated, but not for removal
- System.runFinalizersOnExit(boolean)
  - JDK-4240589: if called with true, can cause unavoidable VM crash
  - Filed in 1999! The only fix is to remove it.
- Obsolete SecurityManager calls
  - inCheck, getInCheck(), currentClassLoader(), currentLoadedClass(), classDepth(), classLoaderDepth(), inClass(), inClassLoader(), checkTopLevelWindow(), ...
  - all vestiges of the old security model, superseded in Java 1.2

Static Analysis (jdeprscan)

- New tool introduced in Java 9
- Statically analyzes class files and jar files against Java SE APIs
- Looks for and reports usage of deprecated Java SE APIs

*API information for past releases is built into Java 9. No need to keep old versions of the JDK around.



Demo (Windows 7 compilation): javac -version javac 1.6.0_45

Any More Questions?

Thank you!