What's New in Java 8

Presented by Jeanne Hiesel

21 April 2014

Factoids

- 56 (give or take) JEP's covered over seven milestones
- Relatively narrow scope in number of jdk segments affected [i.e. --/--, core/(6 sub), vm/(3 sub), web/jaxp]
- Released 3/18/2014
- Only showstopper bugs are in the release, non-showstoppers have been deferred
- Webcast of launch by Mark Reinhold on March 25th
- www.oracle.com/events/us/en/java8/index.html

Java SE 8

Java SE 8—Language and Library Features

Brian Goetz

A New Date and Time API—JSR-310

Stephen Colebourne

JDK Hacker on Core Libraries

Paul Sandoz

Introduction to Lambda Expressions

Stuart Marks

Enhanced Metadata—Annotations and Access to Parameter Names

Alex Buckley and Michael Ernst

Performance Improvements in JDK 8

Staffan Friberg

What's New for JavaFX in Java SE 8

Jim Weaver

New Features in Java SE 8: A Developer's Guide

Simon Ritter

NetBeans IDE 8: Toolbox for Java SE 8

Geertjan Wielenga

Nashorn: JavaScript on the JVM

Jim Laskey

Java 8 Security Highlights

Milton Smith

Introducing Java Mission Control 5.3.0

Marcus Hirt

Java SE Embedded 8

Java 8 on the Raspberry Pi

Stephen Chin

Getting Started with Java SE Embedded on Lego Mindstorms EV3

Hinkmond Wong

Migrating from CDC 1.1.x to Java SE Embedded 8

Hinkmond Wong

Developing Embedded Applications with Java SE 8 Compact Profiles

Bob Vandette

Getting Started with JavaFX Embedded on a Raspberry Pi

Lisa Selle

Reducing Dynamic Memory in Java SE Embedded Application

Darryl Mocek

Choosing a Compact Profile for your Deployment

Jim Connors

Real-time Analytics: SE Embedded 8 and OEP on Freescale i.MX6

Jeff Kudrick

Java ME 8

Be an Embedded Developer in Minutes Using Java ME Embedded 8

Angela Caicedo

JSR 360—CLDC 8: Benefits of an Optimized Implementation

Oleg Pliss

JSR 360—CLDC 8: Java Platform for IoT

Michael Lagally

JSR 360—CLDC 8: Generic Networking APIs

Roger Riggs

Unified Development Experience for Java ME 8 and Java SE Embedded 8

Shiva Govindarajapuram

Accessing HW Devices Using Java ME 8 Device I/O API

Thierry Violleau and Jens Pätzold

Java ME 8: Top 10 Features

Terrence Barr

Java ME 8: Tackling the Challenges of Embedded Software Design

Terrence Barr

MEEP—A New Java Profile for the Embedded World

Volker Bauche and Andrey Petushkov

Developing Modular, Service-enabled Applications: Java ME 8

Leonardo Lima

JSR

- JSR 335: Language-level support for lambda expressions (officially, lambda expressions; unofficially, closures) under Project Lambda.
- JSR 223: Project Nashorn, a Javascript runtime which allows developers to embed Javascript code within applications
- JSR 308: Annotations on Java Types
- JSR 310: Date and Time API

M1

- core/lang
  - JEP117
    - Remove the Annotation-Processing Tool (apt) its associated API, and the documentation from the JDK.

M3

- core/sec
  - JEP124
    - Improve/Enhance the certificate revocation-checking API to support best-effort checking, end-entity certificate checking, and mechanism-specific options and parameters.
  - JEP130
    - Implement the SHA-224 message-digest algorithm and related algorithms.
  - JEP131
    - Include the Sun PKCS#11 Crypto Provider in the JDK for 64-bit Windows.

M4

- core/libs
  - JEP112
    - Improve the maintainability and performance of the standard and extended charset implementations.
- core/sec
  - JEP129
    - Provide implementations of the cryptographic algorithms required by NSA Suite B.

M5

- vm/gc
- vm/rt
- core/--
- core/lang
- core/libs
- core/i18n
- core/sec

M5

- vm/gc
  - JEP122
    - Remove the permanent generation from the Hotspot JVM and thus the need to tune the size of the permanent generation.
- vm/rt
  - JEP136
    - Provide additional contextual information about bytecode-verification errors to ease diagnosis of bytecode or stackmap deficiencies in the field.
- core/--
  - JEP153
    - Enhance the java command-line launcher to launch JavaFX applications.

M5

- core/lang
  - JEP105
    - Extend the Compiler Tree API to provide structured access to the content of javadoc comments.
  - JEP106
    - Extend the javax.tools API to provide access to javadoc.
- core/libs
  - JEP177
    - Optimize java.text.DecimalFormat.format by taking advantage of numerical properties of integer and floating-point arithmetic to accelerate cases with two or three digits after the decimal point.

M5

- core/i18n
  - JEP127
    - Create a tool to convert LDML (Locale Data Markup Language) files into a format usable directly by the runtime library, define a way to package the results into modules, and then use these to incorporate the de-facto standard locale data published by the Unicode Consortium's CLDR project into the JDK.
  - JEP128
    - Define APIs so that applications that use BCP 47 language tags (see RFC 5646) can match them to a user's language preferences in a way that conforms to RFC 4647.
  - JEP133
    - Extend existing platform APIs to support version 6.2 of the Unicode Standard.

M5

- core/sec
  - JEP113
    - Add the MS-SFU extensions to the JDK's Kerberos 5 implementation.
  - JEP114
    - Add support for the TLS Server Name Indication (SNI) Extension to allow more flexible secure virtual hosting and virtual-machine infrastructure based on SSL/TLS protocols.
  - JEP121
    - Provide stronger Password-Based-Encryption (PBE) algorithm implementations in the SunJCE provider.

M6

- --/--
- vm/--
- vm/gc
- vm/rt
- core/lang
- core/sec

M6

- --/--
  - JEP138
    - Introduce autoconf (./configure-style) build setup, refactor the Makefiles to remove recursion, and leverage JEP 139 (Enhance javac to Improve Build Speed).
  - JEP160
    - Improve the implementation of method handles by replacing assembly language paths with an optimizable intermediate representation and then refactoring the implementation so that more work is done in portable Java code than is hardwired into the JVM.
  - JEP164
    - Improve the out-of-box AES Crypto performance by using x86 AES instructions when available, and by avoiding unnecessary re-expansion of the AES key.

M6

- vm/--
  - JEP142
    - Define a way to specify that one or more fields in an object are likely to be highly contended across processor cores so that the VM can arrange for them not to share cache lines with other fields, or other objects, that are likely to be independently accessed.
- vm/gc
  - JEP173
    - Remove three rarely-used combinations of garbage collectors in order to reduce ongoing development, maintenance, and testing costs.
- vm/rt
  - JEP147
    - Reduce the HotSpot's class metadata memory footprint in order to improve performance on small devices.
  - JEP148
    - Support the creation of a smaller VM that is no larger than 3MB.

M6

- core/lang
  - JEP139
    - Reduce the time required to build the JDK and enable incremental builds by modifying the Java compiler to run on all available cores in a single persistent process, track package and class dependences between builds, automatically generate header files for native methods, and clean up class and header files that are no longer needed.
  - JEP172 DocLint
    - Provide a means to detect errors in Javadoc comments early in the development cycle and in a way that is easily linked back to the source code.

M6

- core/libs
  - JEP103
    - Add additional utility methods to java.util.Arrays that use the JSR 166 Fork/Join parallelism common pool to provide sorting of arrays in parallel.
  - JEP135
    - Define a standard API for Base64 encoding and decoding.
  - JEP149
    - Reduce the dynamic memory used by core-library classes without adversely impacting performance.
  - JEP150
    - Define a new date, time, and calendar API for the Java SE platform.
  - JEP170
    - Minor enhancements to JDBC to improve usability and portability, now V4.2

M6

- core/sec
  - JEP166
    - Facilitate migrating data from JKS and JCEKS keystores by adding equivalent support to the PKCS#12 keystore. Enhance the KeyStore API to support new features such as entry metadata and logical views spanning several keystores. Enable the strong crypto algorithms introduced in JEP-121 to be used to protect keystore entries.

M7

- --/--
- vm/rt
- core/lang
- core/libs
- core/net
- core/sec
- web/jaxp

M7

- --/--
  - JEP126
    - Add lambda expressions (closures) and supporting features, including method references, enhanced type inference, and virtual extension methods, to the Java programming language and platform.
  - JEP161
    - Define a few subset Profiles of the Java SE Platform Specification so that applications that do not require the entire Platform can be deployed and run on small devices.
  - JEP162
    - Undertake changes to smooth the eventual transition to modules in a future release, provide new tools to help developers prepare for the modular platform, and deprecate certain APIs that are a significant impediment to modularization.

M7

- --/--
  - JEP174
    - Design and implement a new lightweight, high-performance implementation of JavaScript, and integrate it into the JDK. The new engine available to Java applications via the existing javax.script API, and a new command-line tool.
  - JEP176
    - Improve the security of the JDK's method-handle implementation by replacing the existing hand-maintained list of caller-sensitive methods with a mechanism that accurately identifies such methods and allows their callers to be discovered reliably.
  - JEP179
    - There is a long-standing shortcoming in the JDK in terms of clearly specifying the support and stability usage contract for com.sun.* types and other types shipped with the JDK that are outside of the Java SE specification. These contracts and potential evolution policies should be clearly captured both in the source code of the types and in the resulting class files. This information can be modeled with JDK-specific annotation types.

M7

- vm/rt
  - JEP171
    - Add three memory-ordering intrinsics to the sun.misc.Unsafe class.
- core/lang
  - JEP101
    - Smoothly expand the scope of method type-inference to support (i) inference in method context and (ii) inference in chained calls.
  - JEP104
    - Extend the set of annotatable locations in the syntax of the Java programming language to include names which indicate the use of a type as well as (per Java SE 5.0) the declaration of a type.
  - JEP118
    - Provide a mechanism to easily and reliably retrieve the parameter names of methods and constructors at runtime via core reflection.
  - JEP120
    - Change the Java programming language to allow multiple application of annotations with the same type to a single program element.

M7

- core/libs
  - JEP107
    - Add functionality to the Java Collections Framework for bulk operations upon data. This is commonly referenced as "filter/map/reduce for Java." The bulk data operations include both serial (on the calling thread) and parallel (using many threads) versions of the operations. Operations upon data are generally expressed as lambda functions.
  - JEP109
    - Enhance the Java core library APIs using the new lambda language feature to improve the usability and convenience of the library.
  - JEP119
    - Provide an implementation of the javax.lang.model.* API backed by core reflection rather than by javac. In other words, provide an alternate API to access and process the reflective information about loaded classes provided by core reflection.

M7

- core/libs
  - JEP155
    - Scalable updatable variables, cache-oriented enhancements to the ConcurrentHashMap API, ForkJoinPool improvements, and additional Lock and Future classes.
  - JEP178
    - Enhance the JNI specification to support statically linked native libraries.
  - JEP180
    - Improve the performance of java.util.HashMap under high hash-collision conditions by using balanced trees rather than linked lists to store map entries. Implement the same improvement in the LinkedHashMap class.

M7

- core/net
  - JEP184
    - Define a new type of network permission which grants access in terms of URLs rather than low-level IP addresses.
- core/sec
  - JEP115
    - Support the AEAD/GCM cipher suites defined by SP-800-380D, RFC 5116, RFC 5246, RFC 5288, RFC 5289 and RFC 5430.
  - JEP123
    - Enhance the API for secure random-number generation so that it can be configured to operate within specified quality and responsiveness constraints.
  - JEP140
    - Enable code to assert a subset of its privileges without otherwise preventing the full access-control stack walk to check for other permissions.
- web/jaxp
  - JEP185
    - Upgrade JAXP to version 1.5, which adds the ability to restrict the set of network protocols that may be used to fetch external resources.

Books Already Available

- **Java 8 In Action (Urma, Fusco, Mycroft - Manning MEAP)**
- **Java SE 8 for the Really Impatient (Cay S. Horstmann**, Addison-Wesley Professional, Publication Date: **January 24, 2014** | ISBN-10: **0321927761** | ISBN-13: **978-0321927767 on Amazon)**
- **Java 8 Lambdas: Pragmatic Functional Programming (O'Reilly)**
- **What's New in Java 8: An unofficial guide (Adam L. Davis, LeanPub)**
- **plus others…..**

Pause for Q & A

The Main Features

- Lambdas (to be covered in detail next)
- Streams (my favorite) (or the Stream API)
- Default methods/ functional interfaces (functional programming capability)
- java.time package (alias new Date and Time API)
- Concurrency enhancements
- Nashorn Javascript engine (potential presentation here… No, not for me….one of you!)

Lambdas

- Implemented support for lambda expressions and virtual extension methods
- Improved multicore support using enabling internal iterations
- Also known as ‘closures’ which feature prominently in JVM languages like Groovy, Scala, and Clojure
- To allow code to be streamlined
- Translated into functional interface at compile time
- Can be passed to utility functions

Streams

- 2 modes – sequential and parallel
- Makes collections more flexible and efficient
- Meant to be used with closures/lambdas using a clearer, terser syntax
- Not meant to replace ArrayLists
- Meant make manipulating data more efficient and compete for a share of the 'Big Data' market
- java.util.stream allows use of filter/map/reduce type operations

Streams

- Can use fork/join parallelism
- Can be infinite but are lazy
- java.util.stream allows use of filter/map/reduce type operations
- java.util.stream.Stream serves as gateway to bulk data operations
- Source data is not mutated during operations
- Can use collections and generators as sources
- More than 40 operations to choose from

Streams

List entries = …

- Sequential

```
List<Entry>entry = list.getStream.collect(Collectors.toList()));
Stream stream = entries.stream();
```

- Parallel

```
List<Entry>entry = list.getStream.parallel().collect(Collectors.toList()));
Stream parallelStream = entries.parallelStream();
```

```
List blogNotes = entries.stream()
.parallel()
.filter(e -> e.getEntryDate > 12/31/2014) // concurrent
.sequential()
.map(Entry::new)
.collect(Collectors.toCollection(ArrayList::new));
```

Functional Interfaces and Default Methods

- Default methods added to interface
- Don't have to be overridden in the interface
- Can be run directly from interface
- Done for backward compatibility
- Also to allow addition of Stream without having to change all the classes to implement new method
- Code examples should be part of the Lambda presentation next month

java.time

- JSR310
- Derived from popular Joda Time library
- Meant to eliminate some of the date and time #%@# we currently have to do

Concurrency enhancements

- To take advantage of lambda expressions
- Provide performance improvements for shared counters and hash tables
- Some examples provided as part of Stream
- Some more examples should be part of the Lambdas presentation

Concurrency enhancements

```
List entries =
 asList(new Entry("one"), new Entry("two"), new Entry("three"));
```

- Before Java 8

```
For (Entry entry : entries) {  takeSomeAction(entry); }
```

- Java 8

```
Entries.forEach(this::takeSomeAction);
```

Javascript engine

- Nashorn Javascript engine

  (potential presentation here… No, not for me….one of you!)
- JavaFX designed to replace the Swing GUIs

Q & A

Pages Referenced

- http://openjdk.java.net/projects/jdk8/
- http://openjdk.java.net/projects/lambda/
- https://jdk8.java.net/
- http://bugs.java.com/
- http://www.javacodegeeks.com/
- http://www.i-programmer.info/news/80-java/7048-java-se-8-imminent.html
- http://www.oracle.com/technetwork/java/javase/community/index.html (Bug Database)
- http://java.dzone.com/articles/whats-new-java-8-part-i-javafx
- https://blogs.oracle.com/thejavatutorials/entry/jdk_8_documentation_developer_preview
- https://blogs.oracle.com/java/entry/java_se_8_schedule
- http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
- http://www.infoq.com/articles/javaone2013-roundup
- http://www.infoq.com/articles/Java-7-Features-Which-Enable-Java-8