Java™

ORACLE

55 New Things in JDK 8

Dalibor Topic (@robilad)
Principal Product Manager
May 15th, 2013 - GeeCON

Big Disclaimer

The Java SE 8 Specification is not final
Some features are subject to change
Some features are not implemented yet

Java SE 8 (JSR 337)

Component JSRs

New functionality

- JSR 308: Annotations on types
- JSR 310: Date and Time API
- JSR 335: Lambda expressions

Updated functionality

- JSR 114: JDBC Rowsets
- JSR 160: JMX Remote API
- JSR 199: Java Compiler API
- JSR 173: Streaming API for XML
- JSR 206: Java API for XML Processing
- JSR 221: JDBC 4.0
- JSR 269: Pluggable Annotation-Processing API

JDK Enhancement Proposals (JEPs)

- Regularly updated list of proposals
  - Serve as the long-term roadmap for JDK release projects
  - Roadmap extends for at least three years
- Uniform format and a central archive for enhancement proposals
  - Interested parties can find, read, comment, and contribute
- Process is open to every OpenJDK Committer
- Enhancement is a non-trivial change to the JDK code base
  - Two or more weeks of engineering effort
  - significant change to JDK or development processes and infrastructure
  - High demand from developers or customers

Language

Lambda Expressions

Closures and Functional Programming

- Add lambda expressions (closures) and supporting features
  - Method references, enhanced type inference, virtual extension methods
- Simplify creation and consumption of more abstract, performant libraries
  - Open up possibilities for improved multicore support
- Support smoother library evolution with migration compatibility
  - Allow interfaces to be evolved in a source and binary compatible fashion
- Lambda expressions provide anonymous function types to Java
  - Replace use of single abstract method types

Extension Methods

Bringing Multiple Inheritance (of Functionality) to Java

Provide a mechanism to add new methods to existing interfaces

- Without breaking backwards compatibility
- Gives Java multiple inheritance of behavior, as well as types (but not state!)

```
public interface Set<T> extends Collection<T> {
   public int size();

   ...   // The rest of the existing Set methods

   public T reduce(Reducer<T> r)
     default Collections.<T>setReducer;
}
```

Generalized Target-Type Inference

Method type-parameter inference in method context & chained calls

```
class List<E> {
  static <Z> List<Z> nil() { ... };
  static <Z> List<Z> cons(Z head, List<Z> tail)
  { ... };
  E head() { ... }
}
```

```
List<String> ls = List.nil(); // Inferred correctly
```

error: expected List<Integer>, found List<Object>

```
List.cons(42, List.nil());
```

Annotations On Java Types

Annotations can currently only be used on type declarations

- Classes, methods, variable definitions

Extension for places where types are used

- e.g. parameters

Permits error detection by pluggable type checkers

- e.g. null pointer errors, race conditions, etc

```
public void process(@notnull List data) {…}
```

Access To Parameter Names At Runtime

- Mechanism to retrieve parameter names of methods and constructors
  - At runtime via core reflection
- Improved code readability
  - Eliminate redundant annotations
- Improve IDE capabilities
  - Auto-generate template code

Small Things

Repeating annotations

Multiple annotations with the same type applied to a single program element

No more `apt` tool and associated API

Complete the transition to the JSR 269 implementation

DocTree API

Provide access to the syntactic elements of a javadoc comment

DocLint tool

Use DocTree API to identify basic errors in javadoc comments

Javadoc support in `javax.tools`

Invoke javadoc tools from API as well as command line/exec

Core Libraries

Enhance Core Libraries With Lambdas

- No small task!
  - Java SE 7 has 4024 standard classes
- Modernize general library APIs
- Improve performance
  - Gains from use of invokedynamic to implement Lambdas
- Demonstrate best practices for extension methods

Bulk Data Operations For Collections

Filter, Map, Reduce for Java

Adding map/reduce functionality to collections

- LINQ style processing

Serial and parallel implementations

Parallel implementation builds on Fork-Join framework

Concurrency Updates

Scalable update variables

- `DoubleAccumulator`, `DoubleAdder`, etc
- Multiple variables avoid update contention
- Good for frequent updates, infrequent reads

`ConcurrentHashMap` updates

- Improved scanning support, key computation

`ForkJoinPool` improvements

- Completion based design for IO bound applications

Parallel Array Sorting

- Additional utility methods in `java.util.Arrays`
  - `parallelSort` (multiple signatures for different primitives)
- Anticipated minimum improvement of 30% over sequential sort
  - For dual core system with appropriate sized data set
- Built on top of the fork-join framework
  - Uses Doug Lea's `ParallelArray` implementation
  - Requires working space the same size as the array being sorted

Date And Time APIs

- A new date, time, and calendar API for the Java SE platform
- Supports standard time concepts
  - Partial, duration, period, intervals
  - date, time, instant, and time-zone
- Provides a limited set of calendar systems and be extensible to others
- Uses relevant standards, including ISO-8601, CLDR, and BCP47
- Based on an explicit time-scale with a connection to UTC

JDBC 4.2

Minor enhancements for usability and portability

Generic setter/update methods

- `ResultSet`, `PreparedStatement`, and `CallableStatement`
- Support new data types such as those being defined in JSR 310

REF_CURSOR support for `CallableStatement`

`DatabaseMetaData.getIndexInfo` extended

- new columns for CARDINALITY and PAGES which return a long value

New `DatabaseMetaData` method

- `getMaxLogicalLobSize`
- Return the logical maximum size for a LOB

Base64 Encoding and Decoding

Currently developers are forced to use non-public APIs

- **`sun.misc.BASE64Encoder`**
- **`sun.misc.BASE64Decoder`**

Java SE 8 now has a standard way

- **`java.util.Base64.Encoder`**
- **`java.util.Base64.Decoder`**
- **`encode`**, **`encodeToString`**, **`decode`**, **`wrap`** methods

Small Things

`javax.lang.model` implementation backed by core reflection

- Uniform annotation API to view compile-time and runtime reflective information

Charset implementation improvements

- Reduced size of charsets, improved performance of encoding/decoding

Handle Frequent HashMap Collisions with Balanced Trees

- Switch bucket to balanced tree after threshold to improve worst case perf.

Statically-Linked JNI Libraries

- Enable packing the runtime, application and native code in single binary

Document JDK API Support and Stability

- Specify support and stability contract for com.sun.* types with annotations

Reduced core-library memory usage

- Reduced object size, disable reflection compiler, internal table sizes, etc

Internationalisation (I18N)

Locale Data Packing

- Tool to generate locale data files
  - From LDML format
- Unicode Common Locale Data Repository (CLDR) support
- Locale elements supported from underlying platform

BCP 47 Locale Mapping

- Language tags to indicate the language used for an information object
  - RFC-5646 (Language range)
  - RFC-5456 (Language priority, preference)
- Language range **`Collection<String>`**
- Language priority **`List <String>`**
- Three operations added to **`Locale`** class
  - **`filterBasic`**
  - **`filterExtended`**
  - **`lookup`**

Unicode 6.2

Java SE 7 support Unicode 6.0

Changes in Unicode 6.1 (February, 2012)

- Add 11 new blocks to `java.lang.Character.UnicodeBlock`
- Add 7 new scripts to `java.lang.Character.UnicodeScript`
- Support over 700 new characters in `java.lang.Character`, `String`, and other classes

Changes in Unicode 6.2 (September, 2012)

- Support a new Turkish currency sign (U+20BA)

Security

Configurable Secure Random Number Generator

- Better implementation of `SecureRandom`
- Currently applications can hang on Linux
  - JVM uses `/dev/random`
  - This will block if the system entropy pool is not large enough
- Still a work in progress

Enhanced Certificate Revocation-Checking API

- Current **`java.security.cert`** API is all-or-nothing
  - Failure to contact server is a fatal error
- New classes
  - **`RevocationChecker`**
  - **`RevocationParameters`**

Small Items

Limited `doPrivilege`

Execute Lambda expression with privileges enabled

Mechanical Checking of Caller-Sensitive Methods

Introduce @CallerSensitive annotation to replace hand-maintained lists

NSA Suite B cryptographic algorithms

Conform to standards to meet U.S. government, banking requirements

AEAD CipherSuite support

Conform to standards to meet U.S. government, banking requirements

SHA-224 message digests

Required due to known flaw in SHA-1

Leverage CPU instructions for AES cryptography

Improve encryption/decryption performance

Small Changes

HTTP URL Permissions

- A new type of permission granting access in terms of URL rather then IPs

Microsoft Services For UNIX (MS-SFU) Kerberos 5 extensions

- Enhanced Microsoft interoperability

TLS Server Name Indication (SNI) extension

- More flexible secure virtual hosting, virtual-machine infrastructure

PKCS#11 crypto provider for 64-bit Windows

- Allow use of widely available native libraries

Stronger algorithms for password-based encryption

- Researchers and hackers move on

Overhaul JKS-JCEKS-PKCS12 keystores

- Simplify interacting with Java SE keystores for cryptographic applications

The Platform

Wednesday, May 15, 13

Launch JavaFX Applications

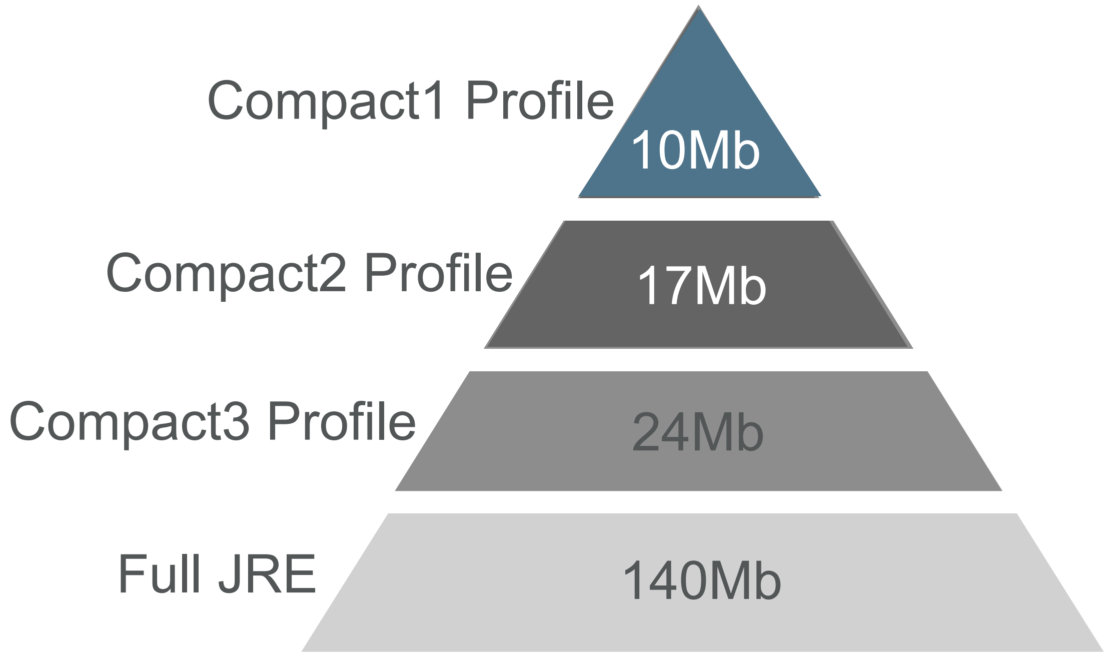
Support the direct launching of JavaFX applications

Enhancement to the java command line launcher

Wednesday, May 15, 13

Compact Profiles

Approximate static footprint goals



Stripped Implementations

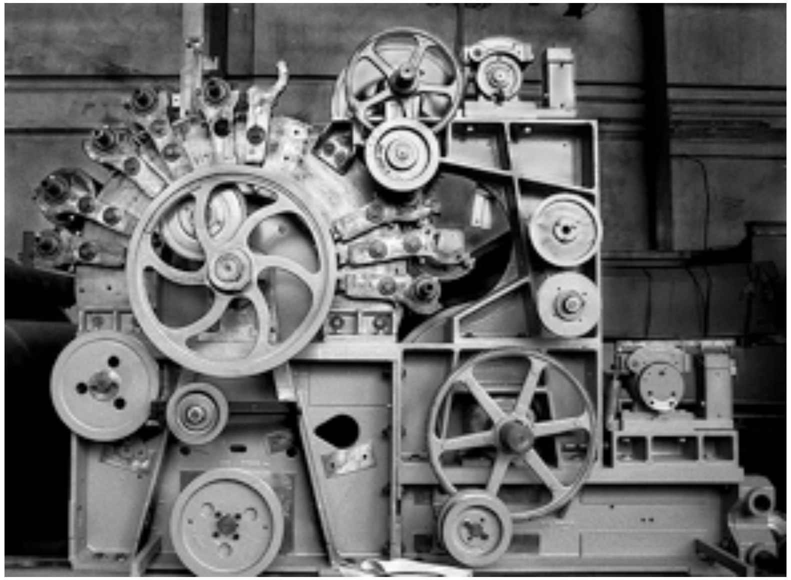
- Applications that ship bundled with a JRE don't need to include all the class libraries
- This does not break 'Write once, run anywhere'
- Only applicable for bundled JRE
  - JRE cannot be used by other applications

Modularization Preparation

Getting Ready For Jigsaw

- Fix some assumptions about ClassLoaders
- Use **`ServiceLoader`** rather than proprietary SPI code
- Tool to analyze application code dependencies
- Deprecate 4 APIs that will impede modularization
  - e.g. **`java.util.logging.LogManager.addPropertyChangeListener`**
- Review and possibly change **`$JAVA_HOME`** normative references
  - Relative v. absolute pathnames

Virtual Machine



Lambda-Form Representation For Method Handles

Assembly language code re-written in Java

Improve performance, quality, and portability of method handles and invokedynamic
Reduce the amount of assembly code in the JVM
Reduce native calls during method handle processing
Better reference implementation of JSR 292 (invokedynamic)

Nashorn JavaScript Engine

Talk by Marcus Lagergren @ 4PM Today, this room

Lightweight, high-performance JavaScript engine

- Integrated into JRE

Use existing `javax.script` API

ECMAScript-262 Edition 5.1 language specification compliance

New command-line tool, `jjs` to run JavaScript

Internationalized error messages and documentation

Retire Rarely-Used GC Combinations

- Rarely used
  - DefNew + CMS
  - ParNew + SerialOld
  - Incremental CMS
- Large testing effort for little return
- Will generate deprecated option messages
  - Won't disappear just yet

Remove The Permanent Generation

Permanently

No more need to tune the size of it

Current objects moved to Java heap or native memory

- Interned strings
- Class metadata
- Class static variables

Part of the HotSpot, JRockit convergence

Fence Intrinsics

- Three new methods in **`sun.misc.Unsafe`** class
  - **`loadFence`**
  - **`storeFence`**
  - **`ringFence`**
- Required by library code
  - Ensure memory access operations do not get reordered
- Not intended to be used by application developers
  - May be exposed as public API later

Small Things

- Enhanced verification errors
  - Additional contextual information on bytecode verification errors
- Reduce cache contention on specified fields
  - Pad variables to avoid sharing cache lines
- Reduce class metadata footprint
  - Use techniques from CVM of Java ME CDC
- Small VM
  - `libjvm.so` <3MB by compiling for size over speed

The JDK

Increased Build Speed, Simplified Setup

Autoconf based build system

- `./configure` style build setup

Enhance javac to improve build speed

- Run on all available cores
- Track package and class dependences between builds
- Automatically generate header files for native methods
- Clean up class and header files that are no longer needed

Conclusions

Java SE 8 will add plenty of new features (and remove a few)

- Language
- Libraries
- JVM

Java continues to evolve!

- openjdk.java.net/projects/jdk8
- jdk8.java.net
- www.jcp.org
- openjdk.java.net/jeps

MAKE THE
FUTURE
JAVA

Java

ORACLE

Java™