



55 New Features in Java SE 8

Stephen Chin (@steveonjava)
Java Technology Ambassador
JavaOne Content Chair

MAKE THE
FUTURE
JAVA

ORACLE®
甲骨文

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

- Lambda expressions provide anonymous function types to Java
 - Replace use of single abstract method types

```
public interface DoStuff {  
    boolean isGood(int x);  
}
```

```
void doSomething(DoStuff d) {  
    if (d.isGood(myVariable))  
        ...  
}
```

```
doSomething(answer -> answer ==  
42);
```

Extension Methods

Bringing Multiple Inheritance (of Functionality) to Java

- Provide a mechanism to add new methods to existing interfaces
 - Without breaking backwards compatability
 - Gives Java multiple inheritance of behaviour, 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;  
}
```

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(@nonnull List data) {...}
```


Generalized Target-Type Inference

Improved usability of generics

```
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
```

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

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



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 **javafx.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
- Modernise general library APIs
- Improve performance
 - Gains from use of invokedynamic to implement Lambdas
- Demonstrate best practices for extension methods

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
 - Thread that is blocked hands work to thread that is running

Bulk Data Operations For Collections

Filter, Map, Reduce for Java

- Adding .Net functionality
 - LINQ style processing
- Serial and parallel implementations
 - Generally expressed with Lambda statements
- Parallel implementation builds on Fork-Join framework

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

- Add 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
- Reduced core-library memory usage
 - Reduced object size, disable reflection compiler, internal table sizes, etc

Small Things

- Optimize `java.text.DecimalFormat.format`
 - Improve performance, multiply by 100.0 or 1000.0 (2 or 3 DP only)
- Statically Linked JNI Libraries
 - Needed for embedded applications
 - Currently only dynamically linked supported
- Handle frequent **HashMap** collisions with balanced trees
 - Hash bucket switches from linked list to balanced tree at certain threshold

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`

HTTP URL Permissions

- New type of network permission
 - Grant access in terms of URLs, rather than IP addresses
- Current way to specify network permissions
 - `java.net.SocketPermission`
 - Not restricted to just HTTP
 - Operates in terms of IP addresses only
- New, higher level capabilities
 - Support HTTP operations (POST, GET, etc)
 - Build on limited `doPrivilege` feature

Small Items

- Limited **doPrivilege**
 - Execute Lambda expression with privileges enabled
- 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

- 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

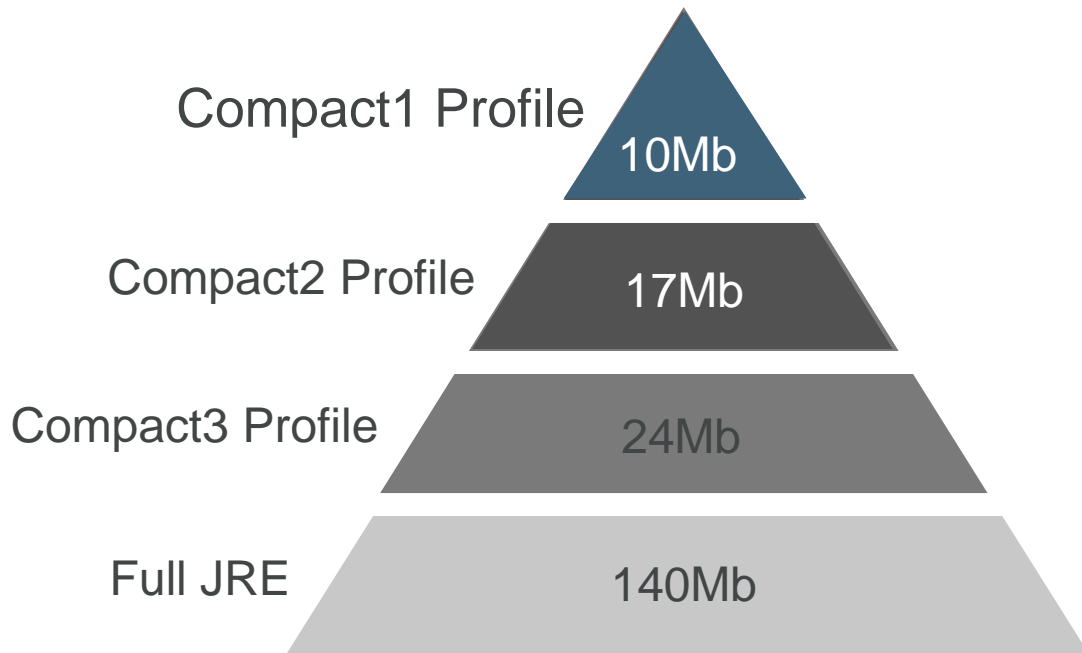


Launch JavaFX Applications

- Support the direct launching of JavaFX applications
- Enhancement to the java command line launcher

Compact Profiles

Approximate static footprint goals



Modularisation Preparation

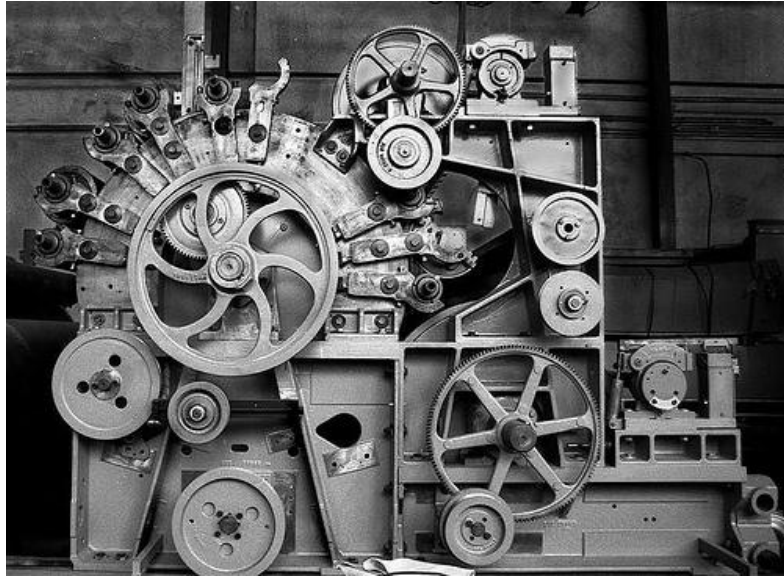
Getting Ready For Jigsaw

- Fix some assumptions about classloaders
- Use **ServiceLoader** rather than proprietary SPI code
- JDK tool to analyse application code dependencies
- Deprecate APIs that will impede modularisation
 - e.g. `java.util.logging.LogManager.addPropertyChangeListener`
- Review and possibly change **\$JAVA_HOME** normative references
 - Relative v. absolute pathnames

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

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

- 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, `jjc` to run JavaScript
- Internationalised 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 Intrinsic

- 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

Mechanical Checking of Caller-Sensitive Methods

- Improve security of JDK method-handle implementation
- New `@CallerSensitive` annotation
- `SecurityManager.checkMemberAccess` deprecated
 - In future may throw an unconditional exception
- `java.util.logging.Logger` revised
 - Remove stack walk in search of resource bundle
 - Related to modularisation preparation

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



Stephen Chin

tweet: @steveonjava

blog: <http://steveonjava.com>

Credit to Simon Ritter for Original Technical Content

NIGHTHACKING TOUR



REAL GEEKS
LIVE HACKING

NIGHTHACKING.COM

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.