From JDK 9 To 13 And Beyond

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava

JDK 9: Big And Small Changes

Process API Updates
HTTP 2 Client
Improve Contended Locking
Unified JVM Logging
Compiler Control
Variable Handles
Segmented Code Cache
Smart Java Compilation, Phase T
The Modular JDK
Modular Source Code
Elide Deprecation Warnings on I tatement
Resolve Lint and Doclint Warnin
Milling Project Coin
Remove GC Combinations Depre JDK 8
Tiered Attribution for javac
Process Import Statements Corr
Annotations Pipeline 2.0
Datagram Transport Layer Secur S)
Modular Run-Time Imag
Simplified Doclet API
jshell: The Java Shell (Rea nt Loop)
New Version-String Scheme
HTML5 Javadoc
Javadoc Search
UTF-8 Property Files
Unicode 7.0
Add More Diagnostic Commands
Create PKCS12 Keystores by Default
Remove Launch-Time JRE Version Selection

Improve Secure Application Performance
Generate Run-Time Compiler Tests Automatically
Test Class-File Attributes Generated by javac
Parser API for Nashorn
Linux/AArch64 Port
Multi-Release JAR Files
Remove the JVM TI hprof Agent
ve the jhat Tool
JVM Compil ace
ion-Layer Negotia nsion
Valid Comman ag Ar
Levera nstructi GHA KSA
Compil er Platt
Make G efault G
OCSP S or TLS
Store I Strings rch
Multi on Imag
Use ale Data lt
aFX UI Con SS APIs fo rization
t Strings
Merge Selected Xerces Fixes into JAX
BeanInfo Annotations
Update JavaFX/Media to Newer Version of GStreamer
HarfBuzz Font-Layout Engine
Stack-Walking API
Encapsulate Most Internal APIs
Module System
TIFF Image I/O
HiDPI Graphics on Windows and Linux

Platform Logging API and Service
Marlin Graphics Renderer
More Concurrency Updates
Unicode 8.0
XML Catalogs
Convenience Factory Methods for Collections
Reserved Stack Areas for Critical Sections
Unified
Platf eatures
DRB Secure mplementations
En Method Ha
Mo va Applicat aging
Dy king of Lar efined Object Models
Enh
Addit s Objects in G1
Improve Test Failure T hooting
Indify String Concaten
HotSpot C++ Unit-Te ework
jlink The Java Link
Enal
New H System
Spin-Wait Hints
SHA-3 Hash Algorithms
Disable SHA-1 Certificates
Deprecate the Applet API
Filter Incoming Serialization Data
Implement Selected ECMAScript 6 Features in Nashorn
Linux/s390x Port

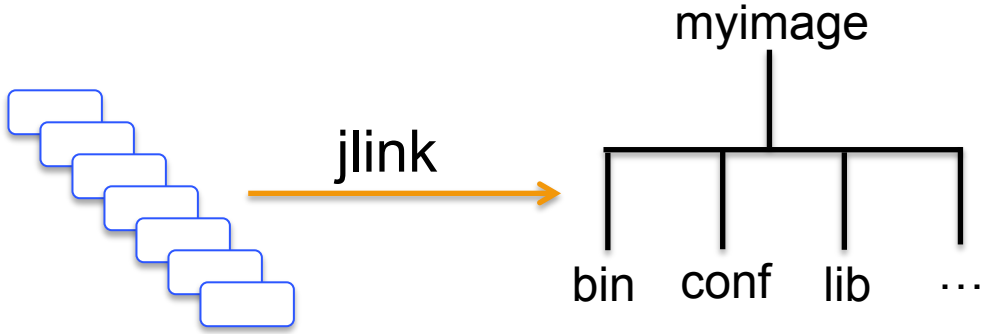
JDK 9

Java Platform Module System (JPMS)

- The core Java libraries are now a set of modules (JEP 220)
  - 75 OpenJDK modules:
    - 24 Java SE
    - 2 aggregator modules
    - 1 smartcard (???)
    - 48 JDK
  - Oracle JDK: 14 additional JDK, 8 JavaFX, 2 Oracle specific
- Most internal APIs now encapsulated (JEP 260)
  - `sun.misc.Unsafe`
  - Some can be used with command line options

jlink: The Java Linker (JEP 282)

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

$ myimage/bin/java --list-modules
java.base@9
java.logging@9
java.sql@9
java.xml@9
com.azul.zapp@1.0
com.azul.zoop@1.0
com.azul.zeta@1.0
```



JDK 9 Onwards And Compatibility

"Clean applications that just depend on java.se *should* just work" - *Oracle*

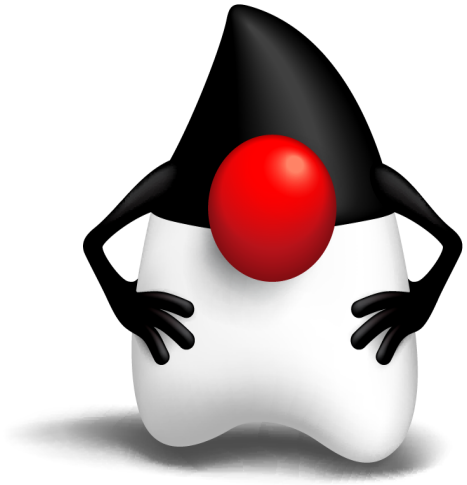
JDK 9: The Clean Up Starts

- JDK 9 was a significant change for Java
  - Deprecated APIs were removed for the first time
    - Six methods and one class
    - JDK 10 removed 1 package, 6 classes, 9 methods and 1 field
  - Redundant features eliminated
    - jhat tool, JVM TI hprof agent
    - Numerous deprecated GC options removed
- JDK 10, 11 and 12 have continued this work
- More features will be removed in the future
    - CMS GC, Nashorn and Pack200 all deprecated. Others?

Compatibility Not Guaranteed

- New versions of Java may include breaking changes
  - Anything for removal will be deprecated first
  - Minimum of one release warning
    - Could be only six months

JDK 10



Local Variable Type Inference (JEP 286)

- Java gets var

```
var userList = new ArrayList<String>(); // infers ArrayList<String>
var stream = list.stream();             // infers Stream<String>


for (var name : userList) {             // infers String
   ...
}

for (var i = 0; i < 10; i++) {              // infers int
   ...
}
```

var: Clearer try-with-resources

```
try (InputStream inputStream = socket.getInputStream();
     InputStreamReader inputStreamReader =
       new InputStreamReader(inputStream, UTF_8);
     BufferedReader bufferedReader =
       new BufferedReader(inputStreamReader)) {
     // Use bufferedReader
}
```

var: Clearer try-with-resources

```
try (var inputStream = socket.getInputStream();
     var inputStreamReader = new InputStreamReader(inputStream, UTF_8);
     var bufferedReader = new BufferedReader(inputStreamReader)) {
     // Use bufferedReader
}
```

var: Reserved Type (Not Keyword)

```
var var = new ValueAddedReseller();
```

```
public class var {
  public var(String x) {
    ...
  }
}
```

```
public class Var {
  public Var(String x) {
    ...
  }
}
```

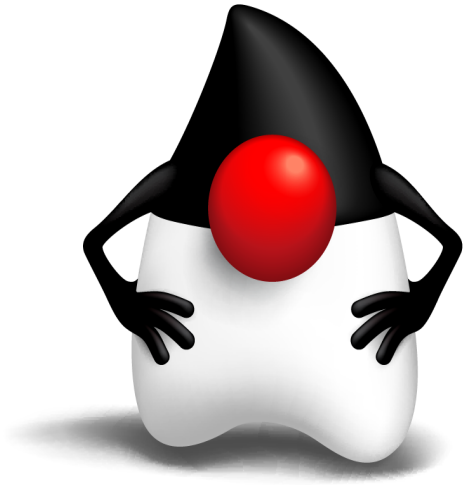
JDK 10: Selected JEPs

- JEP 307: Parallel Full GC for G1
- JEP 310: Application Class-Data Sharing
- JEP 317: Experimental Java-based JIT compiler (Graal)
- JEP 316: Heap allocation on alternative devices (Intel)

JDK 10: APIs

- 73 New APIs
  - `List`, `Set`, `Map.copyOf(Collection)`
  - `Collectors`
    - `toUnmodifiableList`
    - `toUnmodifiableMap`
    - `toUnmodifiableSet`
  - `Optional.orElseThrow()`

JDK 11



323: Extend Local-Variable Syntax

- Local-variable syntax for lambda parameters

```
list.stream()
   .map(s -> s.toLowerCase())
   .collect(Collectors.toList());

list.stream()
   .map((var s) -> s.toLowerCase())
   .collect(Collectors.toList());

list.stream()
   .map((@Notnull var s) -> s.toLowerCase())
   .collect(Collectors.toList());
```

330: Launch Single File Source Code

- JDK 10 has three modes for the Java launcher
  - Launch a class file
  - Launch the main class of a JAR file
  - Launch the main class of a module
- JDK 11 adds a forth
  - Launch a class declared in a source file

```
$ java Factorial.java 4
```

Single File Source Code Shebang

```
#!$JAVA_HOME/bin/java --source 11
public class Factorial {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    int r = (n == 0) ? 0 : 1;
    for (int i = 1; i <= n; i++)
      r *= i;
    System.out.println("n = " + n + ", n! = " + r);
  }
}

$ ./Factorial 4
n = 4, n! = 24
```

JDK 11 Selected JEPs

- 181: Nest-based Access Control
- 309: Dynamic Class-file constants
- 318: Epsilon garbage collector
- 321: HTTP client
- 332: Transport Layer Security (TLS) 1.3
- 333: ZGC: Experimental low-latency garbage collector

New APIs

- New I/O methods
  - `InputStream nullInputStream()`
  - `OutputStream nullOutputStream()`
  - `Reader nullReader()`
  - `Writer nullWriter()`
- `Optional`
  - `isEmpty()  // Opposite of isPresent`

New APIs

- New String methods
  - `isBlank()`
  - `Stream lines()`
  - `String repeat(int)`
  - `String strip()`
  - `String stripLeading()`
  - `String stripTrailing()`

New APIs

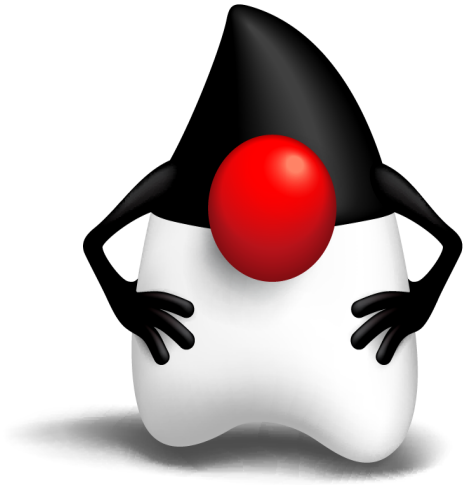
- `Predicate not(Predicate)`

```
lines.stream()
  .filter(s -> !s.isBlank())

lines.stream()
  .filter(Predicate.not(String::isBlank))

lines.stream()
  .filter(not(String::isBlank))
```

JDK 11: Modules Removed

- The `java.se.ee` aggregator-module has been removed
  - `java.corba`
  - `java.transaction`
  - `java.activation`
  - `java.xml.bind`
  - `java.xml.ws`
  - `java.xml.ws.annotation`

JDK 12



Switch Expressions

- First *preview* feature in the OpenJDK
  - Not included in the Java SE standard
- Switch construct was a statement
  - No concept of generating a result that could be assigned
- Rather clunky syntax
  - Every case statement needs to be separated
  - Must remember break (default is to fall through)
  - Scope of local variables is not intuitive

Old-Style Switch Statement

```
int numLetters;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException("Huh?: " + day); };
```

New-Style Switch Expression

```
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
    default -> throw new IllegalStateException("Huh?: " + day);
};
```

New Old-Style Switch Expression

```
int numLetters = switch (day) {
  case MONDAY:
  case FRIDAY:
  case SUNDAY:
    break 6;
  case TUESDAY
    break 7;
  case THURSDAY
  case SATURDAY
    break 8;
  case WEDNESDAY
    break 9;
  default:
    throw new IllegalStateException("Huh?: " + day);
};
```

Switch Expression: Code Blocks

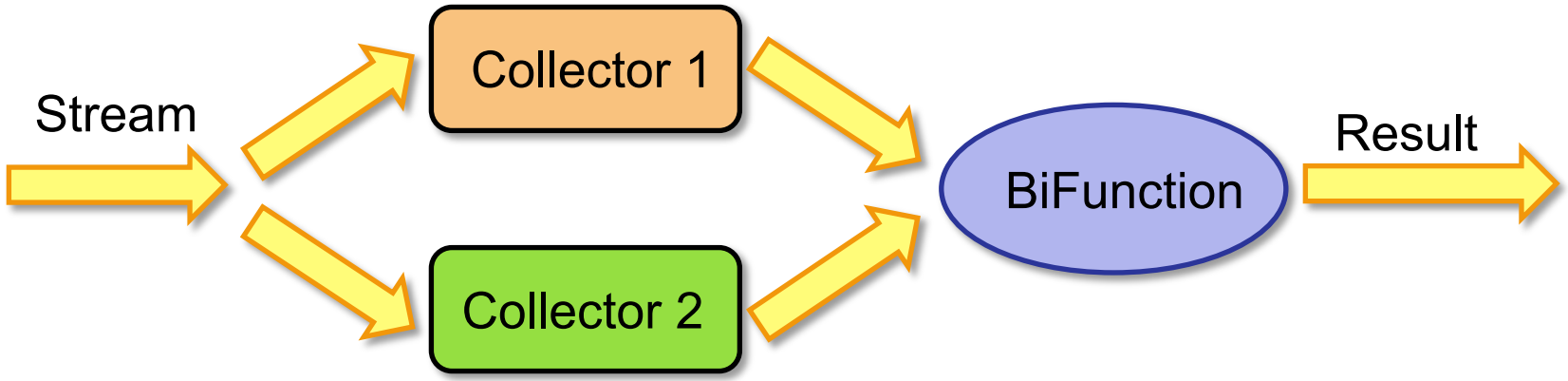
```
int levelResult = switch (level) {
  case 1 -> {
    var x = computeFrom(level);
    logger.info("Level 1 alert");
    break x;
  }
  case 2 -> {
    var x = negativeComputeFrom(level);
    logger.info("Level 2 alert");
    break x;
  }
  default -> throw new IllegalStateException("What level?: " + level);
};
```

JDK 12: Selected JEPs

- 189: Shenandoah GC (Experimental)
- G1 GC updates
  - 344: Abortable mixed collections
  - 346: Return unused committed memory
- 334: JVM constant API
- 341: Default CDS archive

Streams

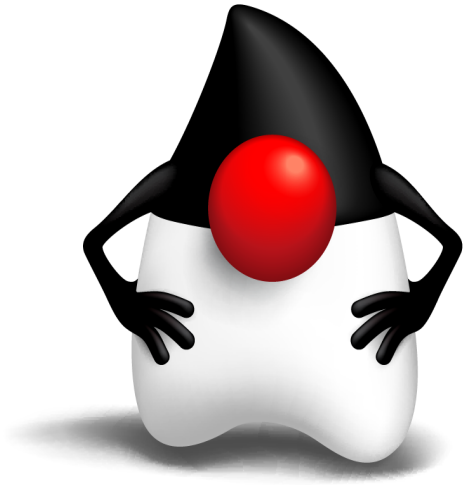
- New collector, teeing
  - `teeing(Collector, Collector, BiFunction)`
- Collect a stream using two collectors
- Use a BiFunction to merge the two collections



Streams

```
// Averaging
Double average = Stream.of(1, 4, 5, 2, 1, 7)
  .collect(teeing(summingDouble(i -> i), counting(),
                  (sum, n) -> sum / n));
```

JDK 13



Text Blocks

```
String webPage = """
                 <html>
                   <body>
                     <p>My web page</p>
                   </body>
                 </html>
                 """;
System.out.println(webPage);
```

```
$ java WebPage
<html>
  <body>
    <p>My web page</p>
  </body>
</html>

$
```

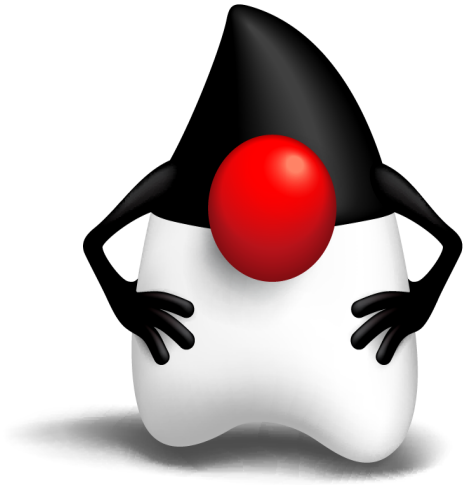
Switch Expression

```
int numLetters = switch (day) {
  case MONDAY:
  case FRIDAY:
  case SUNDAY:
    break 6;
  case TUESDAY
    break 7;
  case THURSDAY
  case SATURDAY
    break 8;
  case WEDNESDAY
    break 9;
  default:
    throw new IllegalStateException("Huh?: " + day);
};
```

Switch Expression

```
int numLetters = switch (day) {
  case MONDAY:
  case FRIDAY:
  case SUNDAY:
    yield 6;
  case TUESDAY
    yield 7;
  case THURSDAY
  case SATURDAY
    yield 8;
  case WEDNESDAY
    yield 9;
  default:
    throw new IllegalStateException("Huh?: " + day);
};
```

Longer Term JDK Futures



Project Valhalla

- Java has:
  - Primitives: for performance
  - Objects: for encapsulation, polymorphism, inheritance, OO
- Problem is where we want to use primitives but can't
  - `ArrayList<int>` won't work
  - `ArrayList<Integer>` requires boxing and unboxing, object creation, heap overhead, indirection reference

Project Valhalla

- Value types
- "Codes like a class, works like a primitive"
  - Can have methods and fields
  - Can implement interfaces
  - Can use encapsulation
  - Can be generic
  - Can't be mutated
  - Can't be sub-classed

Project Loom

- Further work on making concurrent programming simpler
  - Threads are too heavyweight
- Loom will introduce fibres
  - JVM level threads (remember green threads?)
  - Add continuations to the JVM
  - Use the `ForkJoinPool` scheduler
  - Much lighter weight than threads
    - Less memory
    - Close to zero overhead for task switching

Azul's Zulu Java

Zulu Community

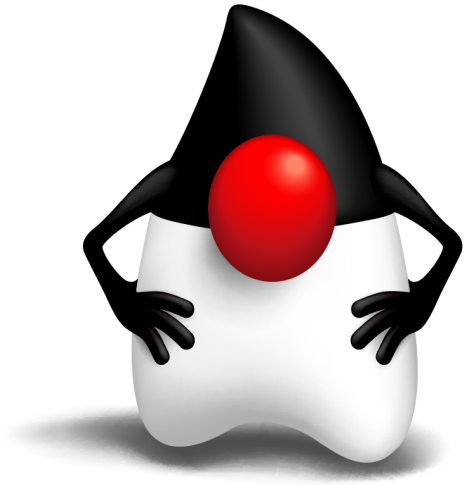
- Azul's ***FREE*** binary distribution of OpenJDK
  - Passes all TCK tests
- JDK 7, 8, 9, 10, 11 and 12 available
- Wide platform support:
  - Intel 64-bit Windows, Mac, Linux
  - Intel 32-bit Windows and Linux
  - ARM 32 and 64-bit

www.azul.com/downloads/zulu

Zulu Enterprise

- Backporting of bug fixes and security patches from supported OpenJDK release
- Zulu 8 supported until March 2026
- Zulu 6 supported until end of 2019
- LTS releases have 9 years active + 2 years passive support
- Medium Term Support releases
  - Two interim releases between LTS releases (13, 15...)
  - Bridge to LTS releases
  - Supported until 18 months *after* next LTS release

Summary



Java Continues To Evolve

- Faster Java releases
  - Feature release every 6 months
  - Access to updates is a consideration
- Lots of ideas to improve Java
  - Value types, fibres, syntax improvements
- Zulu Java has wide platform and JDK version support
  - Very reasonable cost for commercial support

Thank You

Simon Ritter

Deputy CTO, Azul Systems

azul.com

@speakjava