

ORACLE®



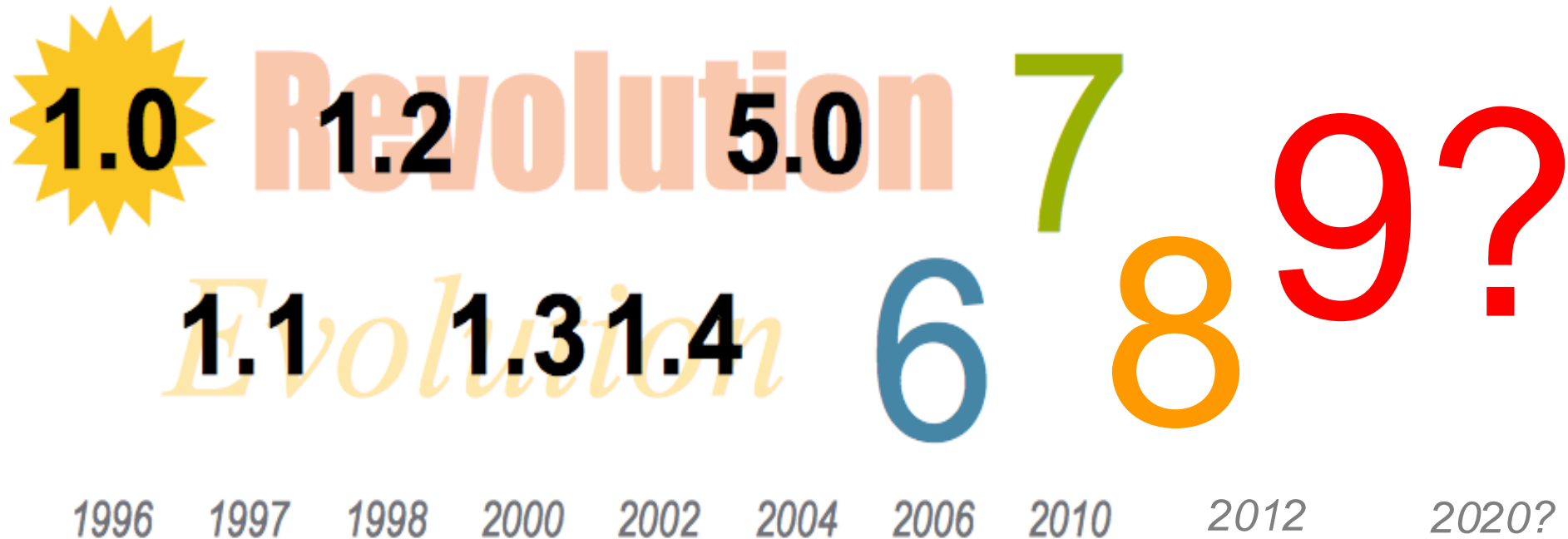
To Java SE 8, and Beyond!

Simon Ritter

Technology Evangelist, Oracle



ORACLE®



Priorities for the Java Platforms



Grow Developer Base



Grow Adoption



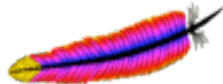
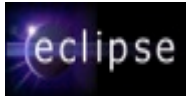
Increase Competitiveness



Adapt to change



Java Communities



Evolving the Language

From “Evolving the Java Language” - JavaOne 2005

- Java language principles
 - Reading is more important than writing
 - Code should be a joy to read
 - The language should not hide what is happening
 - Code should do what it seems to do
 - Simplicity matters
 - Every “good” feature adds more “bad” weight
 - Sometimes it is best to leave things out
- One language: with the same meaning everywhere
 - No dialects
- We will evolve the Java language
 - But cautiously, with a long term view
 - “first do no harm”

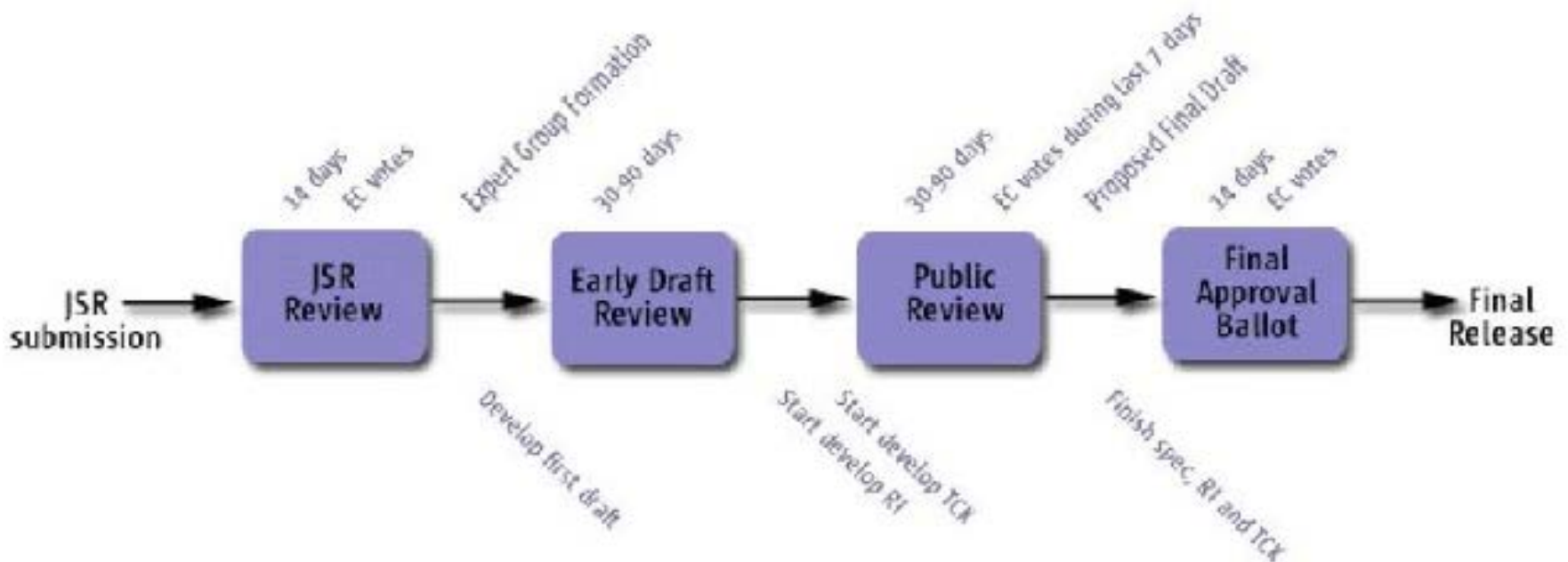
*also “Growing a Language” - Guy Steele 1999
“The Feel of Java” - James Gosling 1997*

How Java Evolves and Adapts

Of the community, by the community, for the community



Java
Community
Process



JSR-348: JCP.next

JCP Reforms



- Developers' voice in the Executive Committee
 - SOUJava
 - Goldman Sachs
 - London JavaCommunity
- JCP starting a program of reform
 - JSR 348: Towards a new version of the JCP



OpenJDK

- Oracle remains committed to OpenJDK
 - The best open source Java implementation
 - Oracle JDK is built on OpenJDK source code
 - Oracle will continue to improve OpenJDK
 - No plans to change licensing of OpenJDK
- Oracle continues to welcome external contributors
 - Companies, researchers, individuals, and more!
- JDK 7 & JDK 8 based on OpenJDK
- Oracle plans to continue to provide OpenJDK 6
 - In collaboration with the OpenJDK Community

OpenJDK



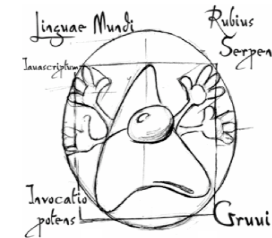
**Oracle and IBM collaborate to
accelerate Java innovation through
OpenJDK**



**Oracle and Apple
announce OpenJDK
Project for Mac OS X**

Java SE 7 Release Contents

- Java Language
 - Project Coin (JSR-334)
- Class Libraries
 - NIO2 (JSR-203)
 - Fork-Join framework, ParallelArray (JSR-166y)
- Java Virtual Machine
 - The DaVinci Machine project (JSR-292)
 - InvokeDynamic bytecode
- Miscellaneous things
- JSR-336: Java SE 7 Release Contents



JVM Convergence

Tim Lindholm • Frank Yellin

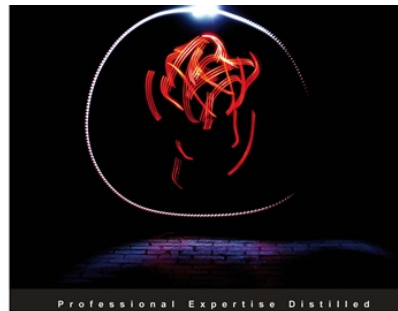
The Java™ Virtual Machine Specification Second Edition

The Java Series

Java™ 2 Platform



... from the Source™



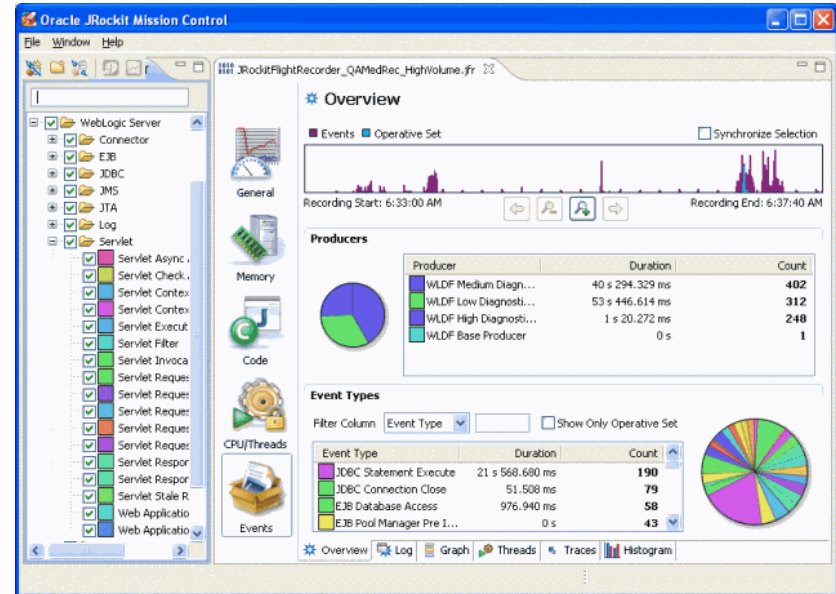
Oracle JRockit

The Definitive Guide

Develop and manage robust Java applications with Oracle's high-performance Java Virtual Machine

Foreword by Adam Messinger,
Vice President of Development in the Oracle Fusion Middleware group

Marcus Hirt Marcus Lagergren [PACKT] enterprise 88
PUBLISHING

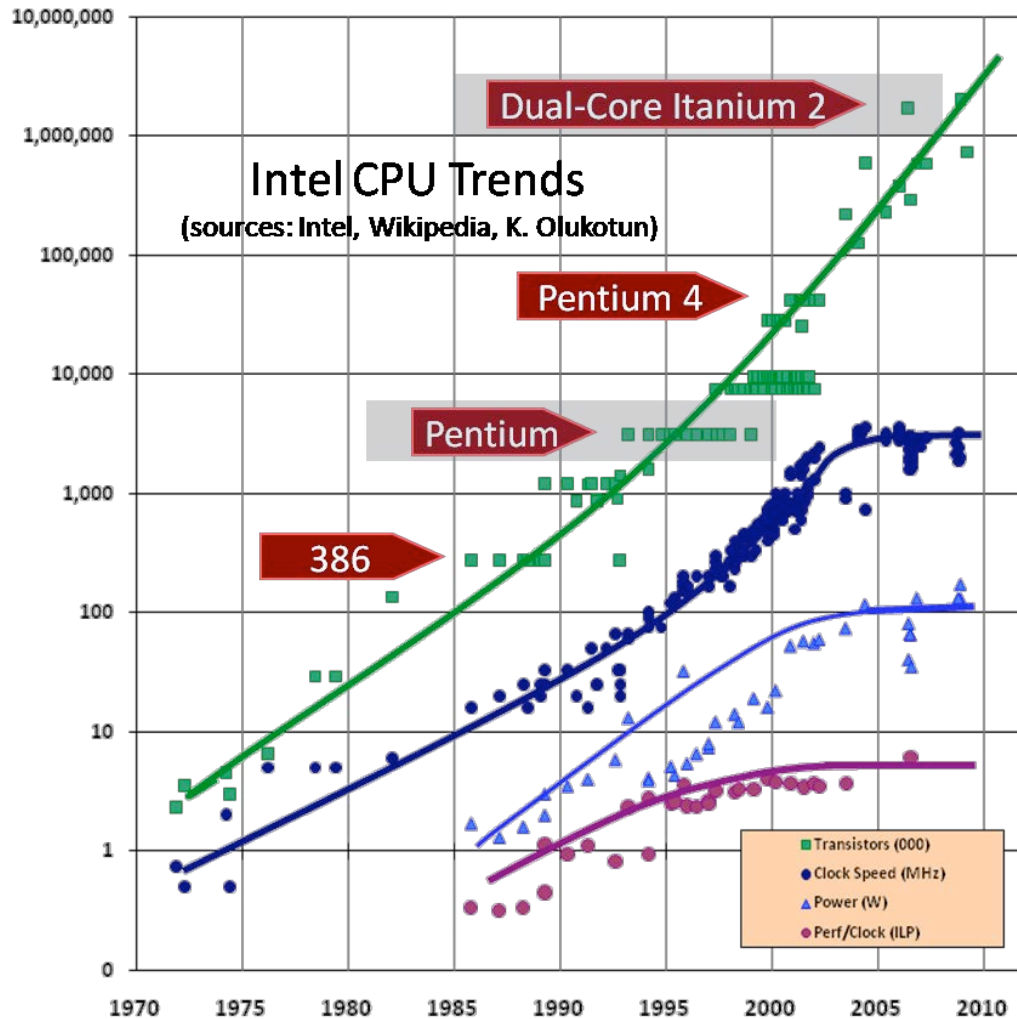


JVM Roadmap

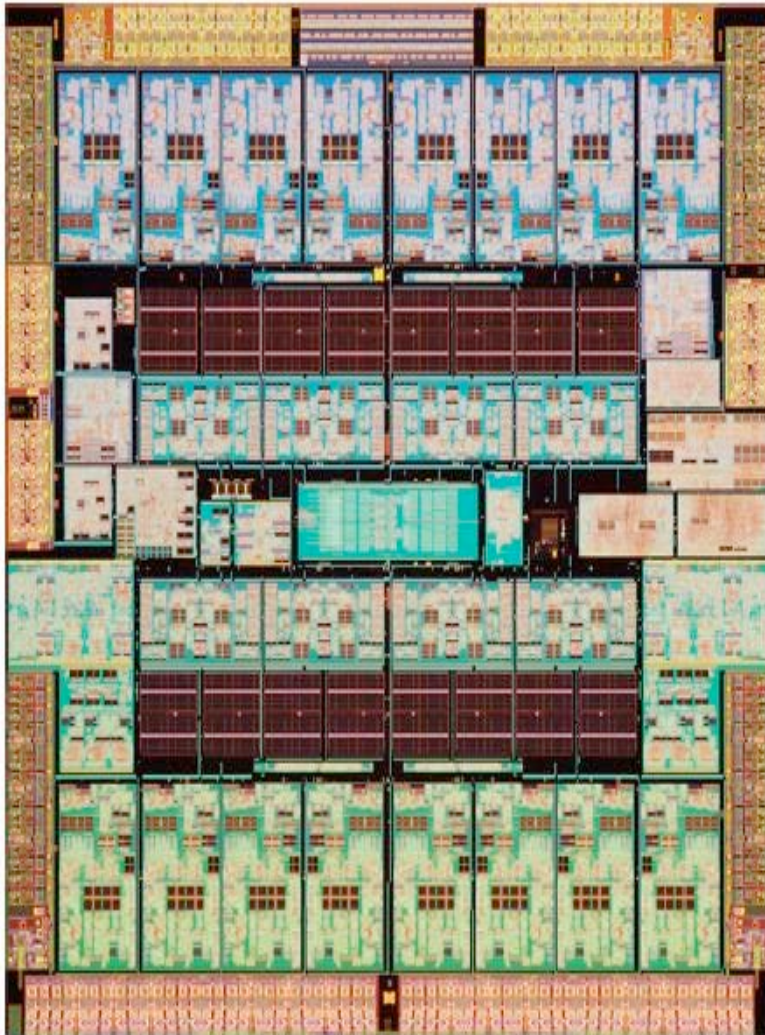
- JRocket Convergence
 - Flight Recorder, verbose logging, performance, deterministic GC-analogue
 - Incrementally ported to JDK 7 updates (2012)
 - Expected to be complete in JDK 8 time frame (2013)
- CVM Convergence
 - Memory footprint, startup time optimizations
 - Initial port in JDK 8 time frame (2013)
 - Incrementally ported in JDK 8 updates (2014+)

- Support for **Java SE 7**
 - Java editor support for project Coin
 - Bulk refactoring of projects and packages to Java SE 7
- Support for **Java EE 6**
 - Concurrent support for the latest Glassfish releases
- Support for JavaFX 2.0
 - Full edit/compile/debug cycle support
 - Visual debugging of JavaFX applications

The (Performance) Free Lunch Is Over



Reality of Modern Chip Architectures



SPARC T1

(2005)

$8 \times 4 = 32$

SPARC T2 (2007)

$8 \times 8 = 64$

SPARC T3 (2011)

$16 \times 8 = 128$

Project Lambda

- Closures are important to simplify making applications more parallel
 - Fork-join is good, but not enough
- Inner classes provide some functionality of closures
 - Too restrictive
 - Too bulky in syntax
 - Not powerful enough
- Java SE 8 will bring lambda statements to the Java language
 - Simple, powerful syntax

```
Collection<Student> students = ...;

double max = Double.MIN_VALUE;

for (Student s : students) {
    if (s.gradYear == 2011)
        max = Math.max(max, s.score);
}
```



```
Collection<Student> students = ...;
```

```
max = students.filter(new Predicate<Student>() {  
    public boolean op(Student s) {  
        return s.gradYear == 2011;  
    }  
}).map(new Extractor<Student, Double>() {  
    public Double extract(Student s) {  
        return s.score;  
    }  
}).reduce(0.0, new Reducer<Double, Double>() {  
    public Double reduce(Double max, Double score) {  
        return Math.max(max, score);  
    }  
});
```

```
Collection<Student> students = ...;
```

```
max = students.filter((Student s) -> s.gradYear == 2011)  
    .map((Student s) -> s.score)  
    .reduce(0.0,  
        (Double max, Double score) ->  
            Math.max(max, score));
```

```
max = students.filter(s -> s.gradYear == 2011)  
    .map(s -> s.score)  
    .reduce(0.0, Math#max);
```

```
max = students.parallel()  
    .filter(s -> s.gradYear == 2011)  
    .map(s -> s.score)  
    .reduce(0.0, Math#max);
```

Project Jigsaw

Modular applications, and Java runtime

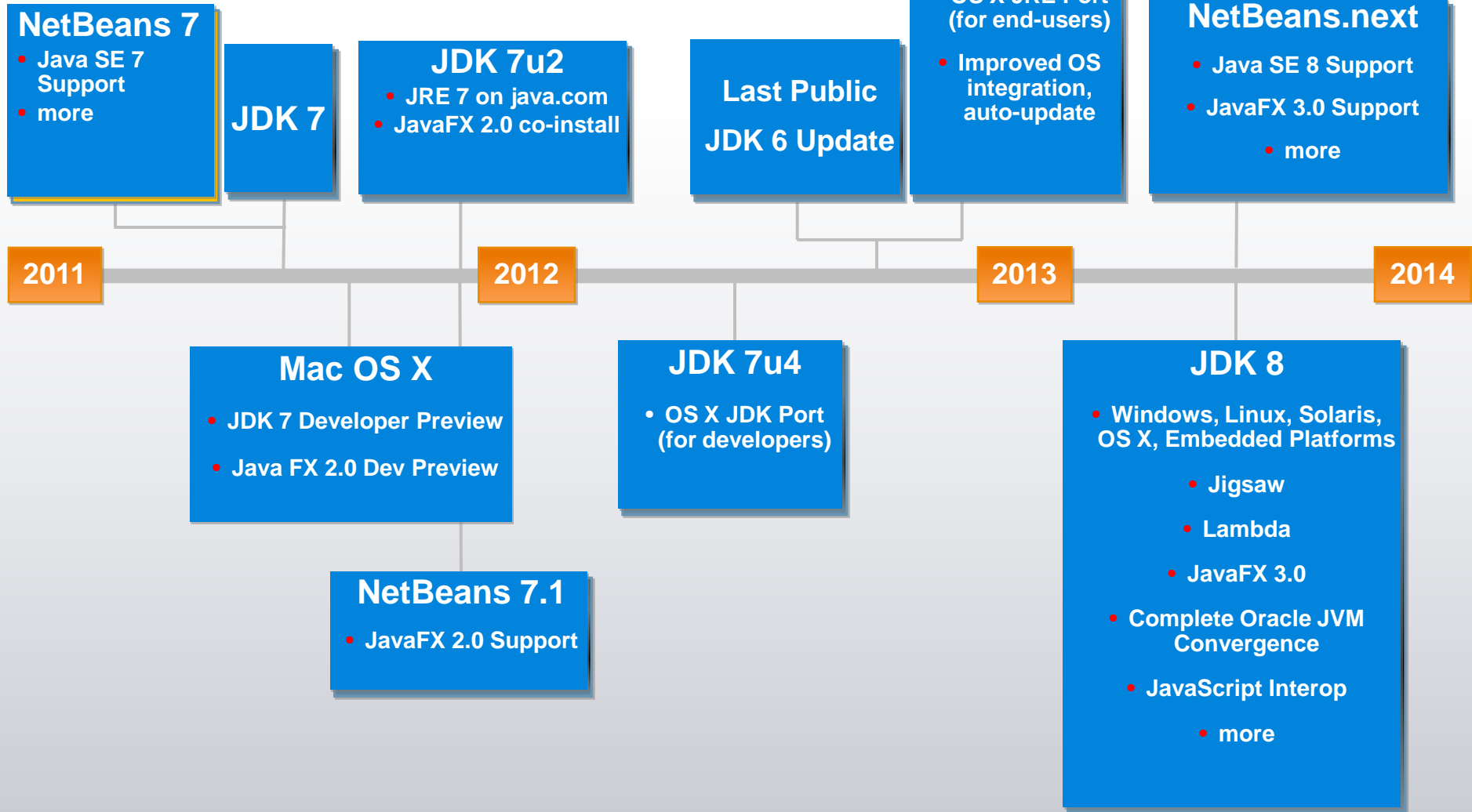
~~classpath~~



JDK 8 – Proposed Content

Theme	Description/Content
Project Jigsaw	<ul style="list-style-type: none">• Module system for Java applications and for the Java platform
Project Lambda	<ul style="list-style-type: none">• Closures and related features in the Java language (JSR 335)• Bulk parallel operations in Java collections APIs (filter/map/reduce)
Oracle JVM Convergence	<ul style="list-style-type: none">• Complete migration of performance and serviceability features from JRockit, including Mission Control and the Flight Recorder
JavaFX 3.0	<ul style="list-style-type: none">• Next generation Java client, Multi-touch
JavaScript	<ul style="list-style-type: none">• Next-gen JavaScript-on-JVM engine (Project Nashorn)• JavaScript/Java interoperability on JVM
Device Support	<ul style="list-style-type: none">• Camera, Location, Compass and Accelerometer
Developer Productivity	<ul style="list-style-type: none">• Annotations onTypes (JSR 308), Minor language enhancements
API and Other Updates	<ul style="list-style-type: none">• Enhancements to Security, Date/Time (JSR 310), Networking, Internationalization, Accessibility, Packaging/Installation

JDK Roadmap



Additional Disclaimers

- Some *ideas* for the Java Platform are shown on the following slides
- Large R&D effort required
- Content and timing highly speculative
- Some things will turn out to be bad ideas
- New ideas will be added
- Still, Java's future is bright (in our humble opinion)!

Java SE 9 (and beyond...)

Interoperability	<ul style="list-style-type: none">• Multi-language JVM• Improved Java/Native integration
Cloud	<ul style="list-style-type: none">• Multi-tenancy support• Resource management
Ease of Use	<ul style="list-style-type: none">• Self-tuning JVM• Language enhancements
Advanced Optimizations	<ul style="list-style-type: none">• Unified type system• Data structure optimizations
Works Everywhere and with Everything	<ul style="list-style-type: none">• Scale down to embedded, up to massive servers• Support for heterogeneous compute models

Vision: Interoperability

- Improved support for non-Java languages
 - Invokedynamic (done)
 - Java/JavaScript interop (in progress – JDK 8)
 - Meta-object protocol (JDK 9)
 - Long list of JVM optimizations (JDK 9+)
- Java/Native
 - Calls between Java and Native without JNI boilerplate (JDK 9)

Vision: Cloud

- Multi-tenancy (JDK 8+)
 - Improved sharing between JVMs in same OS
 - Per-thread/threadgroup resource tracking/management
- Hypervisor aware JVM (JDK 9+)
 - Co-operative memory page sharing
 - Co-operative lifecycle, migration

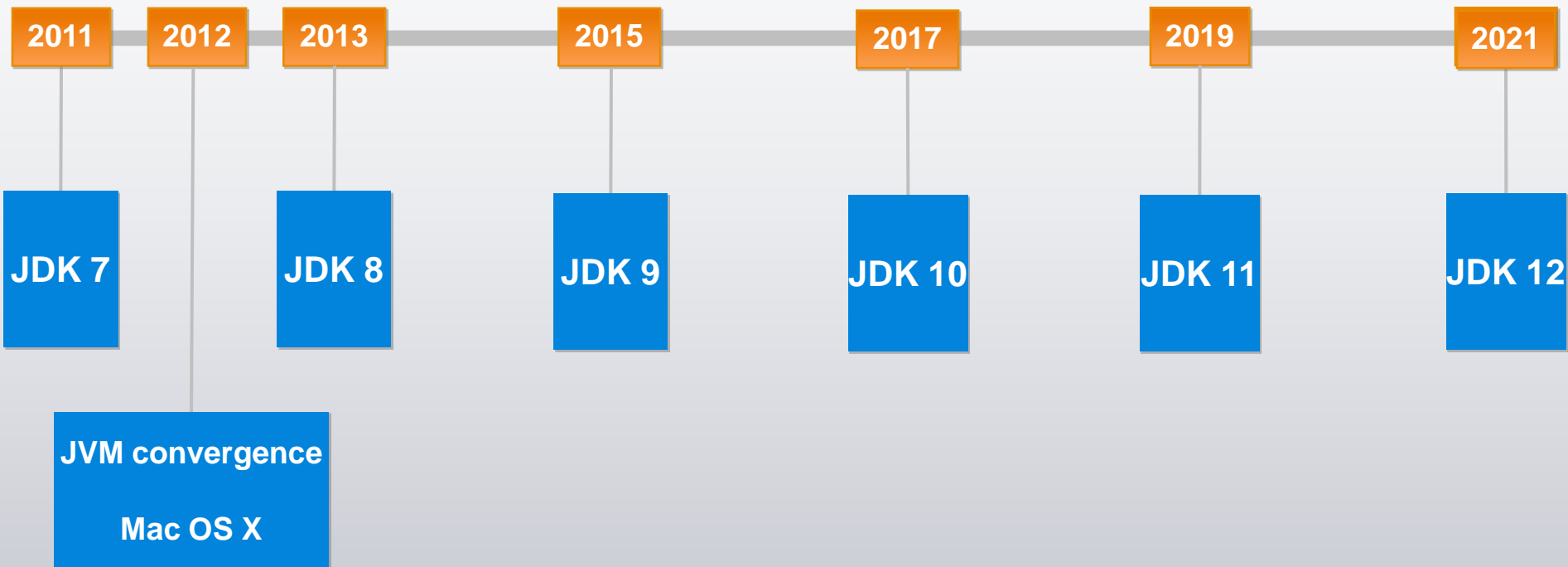
Vision: Language Features

- Large data support (JDK 9)
 - Large arrays (64 bit support)
- Unified type system (JDK 10+)
 - No more primitives, make everything objects
- Other type reification (JDK 10+)
 - True generics
 - Function types
- Data structure optimizations (JDK 10+)
 - Structs, multi-dimensional arrays, etc
 - Close last(?) performance gap to low-level languages

The Path Forward

- Open development
 - Prototyping and R&D in OpenJDK
 - Cooperate with partners, academia, greater community
- Work on next JDK, future features in parallel
- 2-year cycle for Java SE releases

Java SE 2012 to Java 12



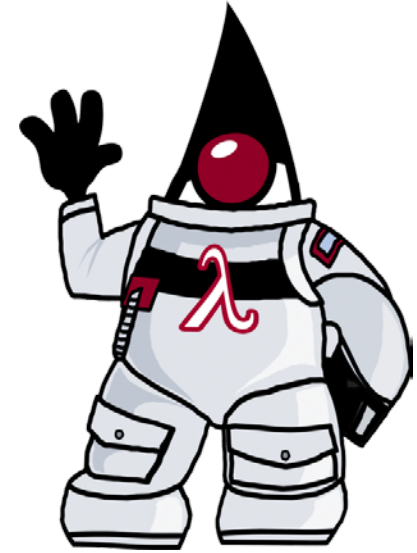
Conclusions


- The Java platform will continue to evolve
- Java SE 8 will add some nice, big features
- Expect to see more in Java SE 9 and beyond
- Java is not the new Cobol

Further Information

- Project Lambda
 - openjdk.java.net/projects/lambda

- Project Jigsaw
 - openjdk.java.net/projects/jigsaw





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.

ORACLE®