Java 8 Update

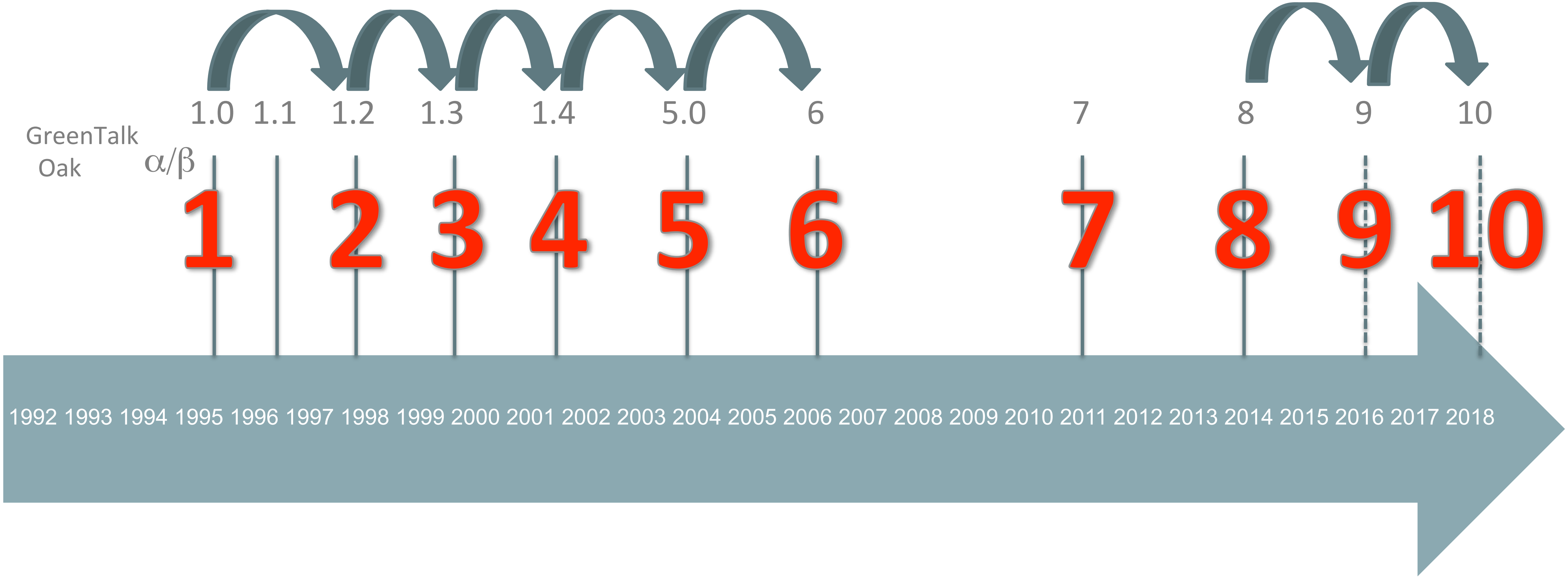
Steve Elliott
Oracle UK
July 2014

Safe Harbor Statement

The following 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.

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. | Oracle Confidential – Internal/Restricted/Highly Restricted

Java Timeline



ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java SE EOL / Lifetime Support Policy

	GA Date	EoPU	Premier Support	Extended Support
Java SE 1.4.2	Feb 2002	Oct 2008	Feb 2010	Feb 2013
Java SE 5	May 2004	Oct 2009	May 2011	May 2015
Java SE 6	Dec 2006	Feb 2013	~~Dec 2013~~ Dec 2015	~~Jun 2017~~ Dec 2018
Java SE 7	Jul 2011	Mar 2015 *	~~Jul 2016~~ Jul 2019	~~Jul 2019~~ Jul 2022
Java SE 8	Mar 2014	Mar 2017 *	Mar 2022	Mar 2025

For details see, http://www.oracle.com/technetwork/java/eol-135779.html

* Or later. Exact date TBD.

Deployment technologies (browser based) : Java 6 Premier – Jun 2017, Java 7+ Premier – 5yrs after GA, No Extended Support (moves to Sustaining)

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java 8 & Java Mission Control 5.3 – GA 18th March 2014

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

HTTP URL Permissions

Base64

Enhanced Verification Errors

Improve Contended Locking

DocTree API

Prepare for Modularization

Lambda (JSR 335)

Remove the Permanent Generation

Generalized Target-Type Inference

Date/Time API (JSR 310)

Bulk Data Operations

Java 8

Parallel Array Sorting

Limited doPrivileged

Repeating Annotations

Compact Profiles

Parameter Names

Nashorn

Unicode 6.2

Configurable Secure-Random Number Generation

TLS Server Name Indication

Type Annotations (JSR 308)

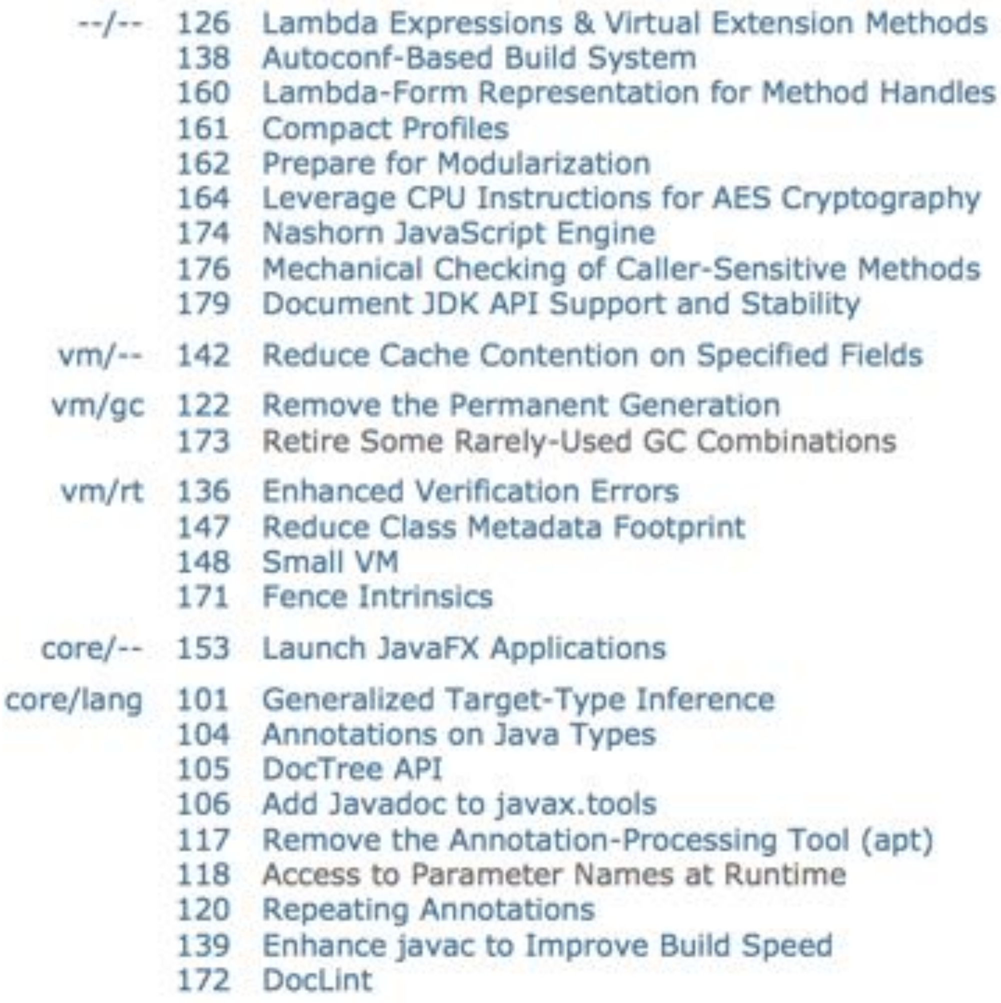
Lambda-Form Representation for Method Handles

Fence Intrinsics

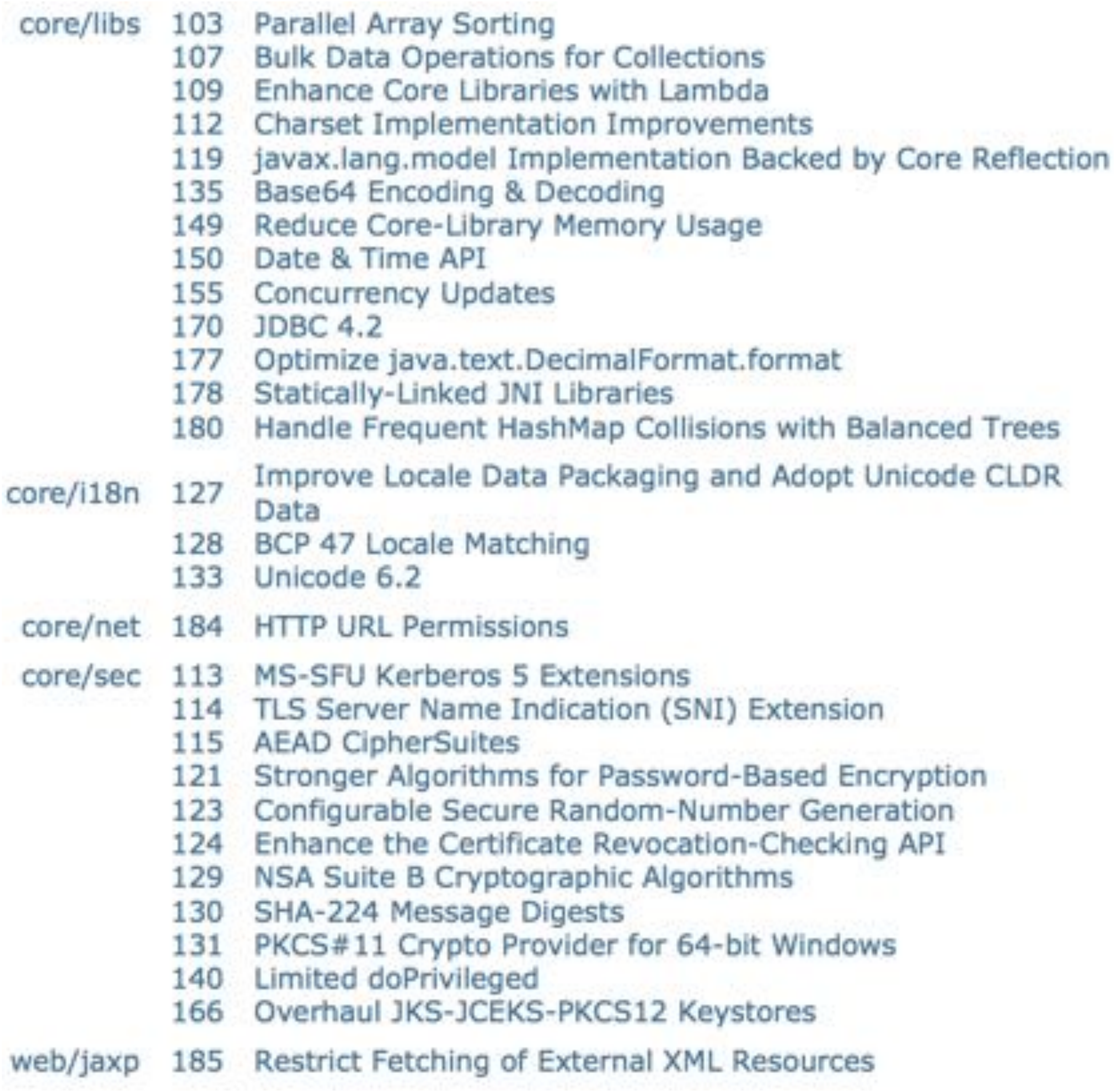
Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java 8

http://openjdk.java.net/projects/jdk8/features



--/--	126	Lambda Expressions & Virtual Extension Methods
	138	Autoconf-Based Build System
	160	Lambda-Form Representation for Method Handles
	161	Compact Profiles
	162	Prepare for Modularization
	164	Leverage CPU Instructions for AES Cryptography
	174	Nashorn JavaScript Engine
	176	Mechanical Checking of Caller-Sensitive Methods
	179	Document JDK API Support and Stability
vm/--	142	Reduce Cache Contention on Specified Fields
vm/gc	122	Remove the Permanent Generation
	173	Retire Some Rarely-Used GC Combinations
vm/rt	136	Enhanced Verification Errors
	147	Reduce Class Metadata Footprint
	148	Small VM
	171	Fence Intrinsics
core/--	153	Launch JavaFX Applications
core/lang	101	Generalized Target-Type Inference
	104	Annotations on Java Types
	105	DocTree API
	106	Add Javadoc to javax.tools
	117	Remove the Annotation-Processing Tool (apt)
	118	Access to Parameter Names at Runtime
	120	Repeating Annotations
	139	Enhance javac to Improve Build Speed
	172	DocLint



core/libs	103	Parallel Array Sorting
	107	Bulk Data Operations for Collections
	109	Enhance Core Libraries with Lambda
	112	Charset Implementation Improvements
	119	javax.lang.model Implementation Backed by Core Reflection
	135	Base64 Encoding & Decoding
	149	Reduce Core-Library Memory Usage
	150	Date & Time API
	155	Concurrency Updates
	170	JDBC 4.2
	177	Optimize java.text.DecimalFormat.format
	178	Statically-Linked JNI Libraries
	180	Handle Frequent HashMap Collisions with Balanced Trees
core/i18n	127	Improve Locale Data Packaging and Adopt Unicode CLDR Data
	128	BCP 47 Locale Matching
	133	Unicode 6.2
core/net	184	HTTP URL Permissions
core/sec	113	MS-SFU Kerberos 5 Extensions
	114	TLS Server Name Indication (SNI) Extension
	115	AEAD CipherSuites
	121	Stronger Algorithms for Password-Based Encryption
	123	Configurable Secure Random-Number Generation
	124	Enhance the Certificate Revocation-Checking API
	129	NSA Suite B Cryptographic Algorithms
	130	SHA-224 Message Digests
	131	PKCS#11 Crypto Provider for 64-bit Windows
	140	Limited doPrivileged
	166	Overhaul JKS-JCEKS-PKCS12 Keystores
web/jaxp	185	Restrict Fetching of External XML Resources

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java SE 8

- Biggest changes to the Java language since Java SE 5
- Coordinated co-evolution of language, libraries, and VM
  - Lambda expressions and interface evolution
  - Bulk data operations on collections, more library support for parallelism
  - Streams API
- The main goals of these changes are:
  - Better developer productivity
  - More reliable code
  - Better performance (especially on multi core hardware)

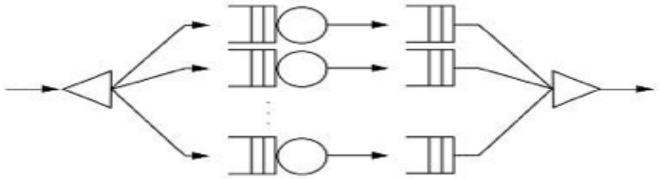
Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Concurrency in Java

java.lang.Thread

java.util.concurrent
(jsr166)

Phasers, etc
(jsr166)



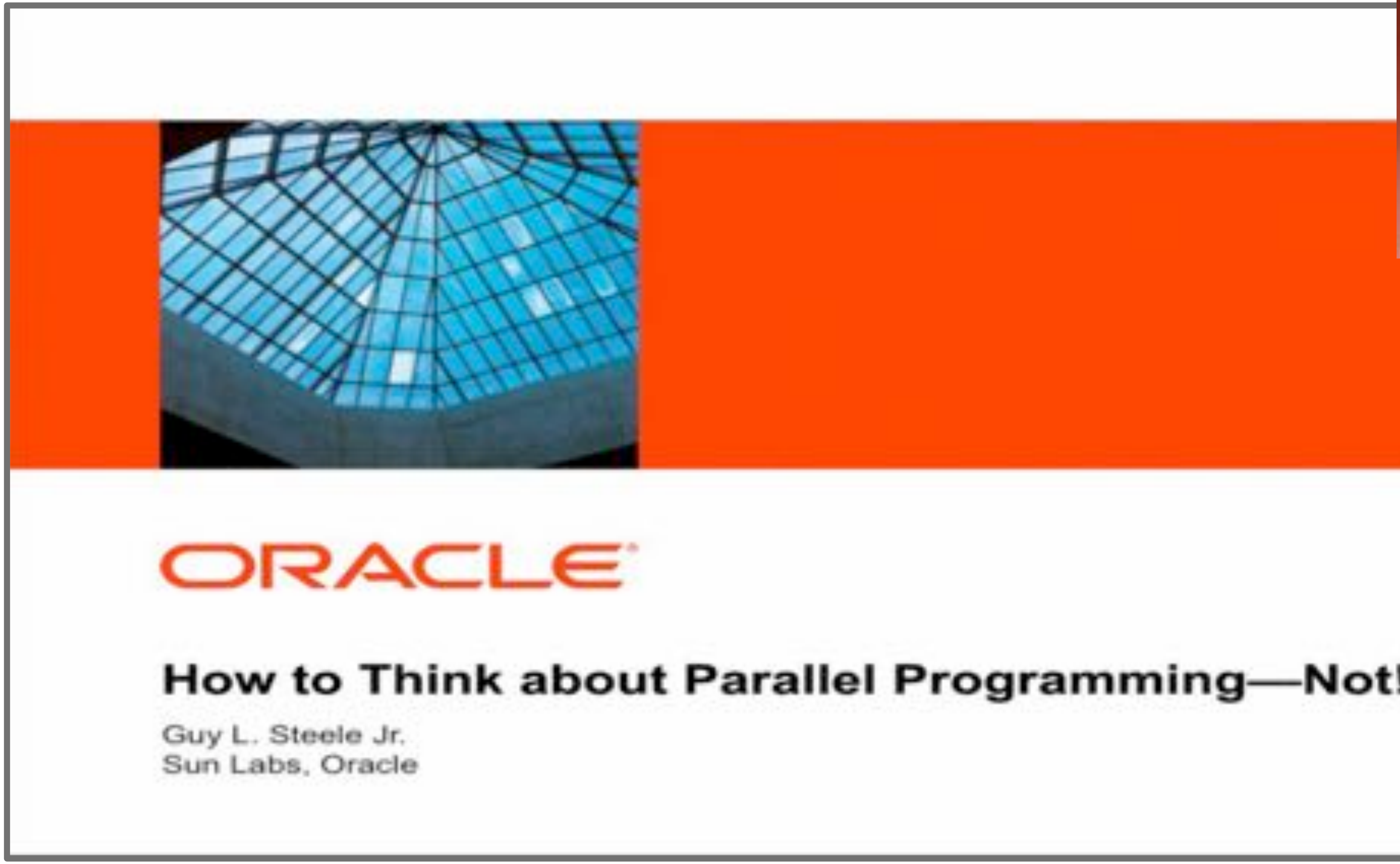
Fork/Join Framework
(jsr166y)

Project Lambda

1.4 5.0 6 7 8

2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |



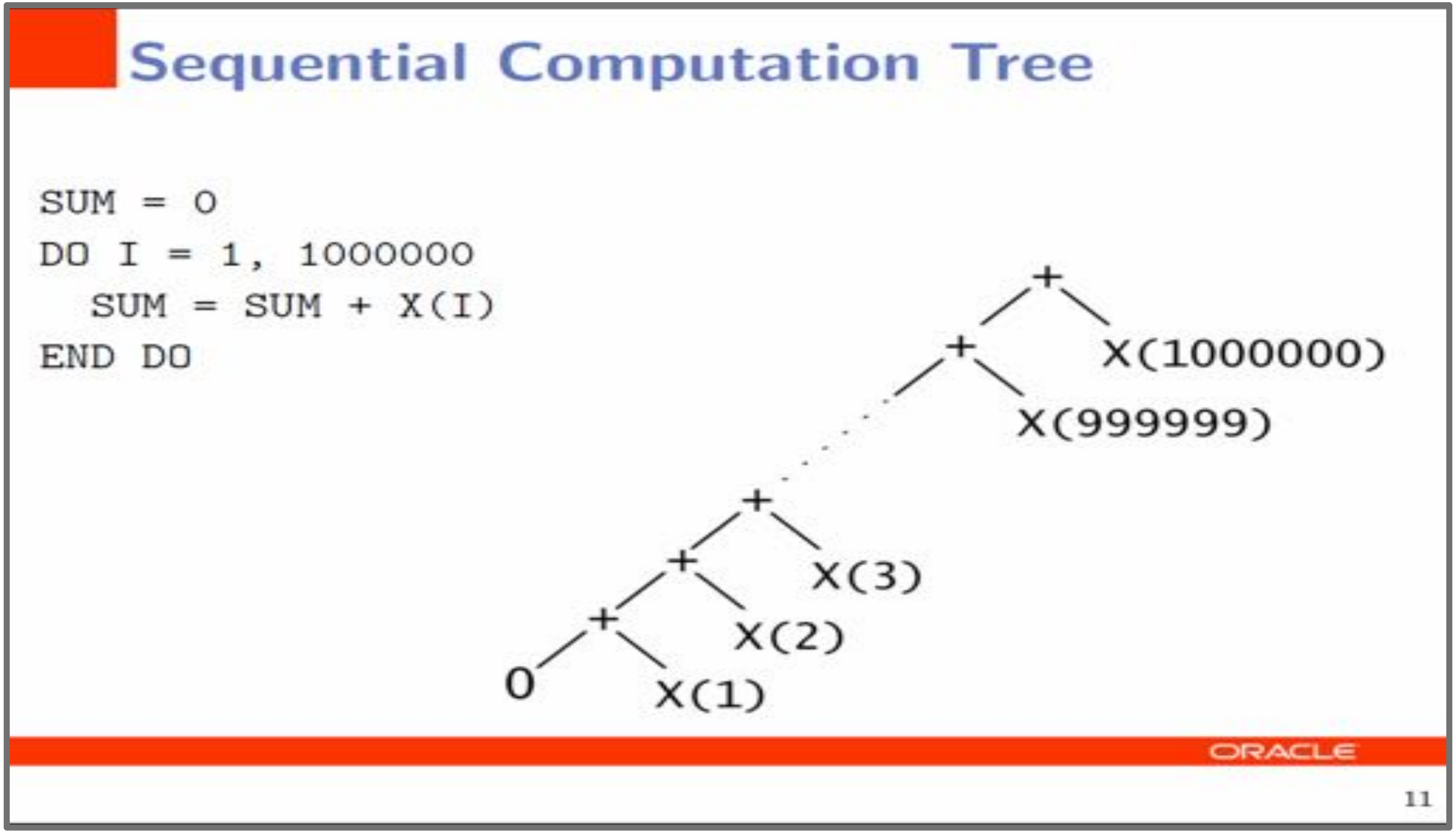
ORACLE

How to Think about Parallel Programming—Not!

Guy L. Steele Jr.
Sun Labs, Oracle

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

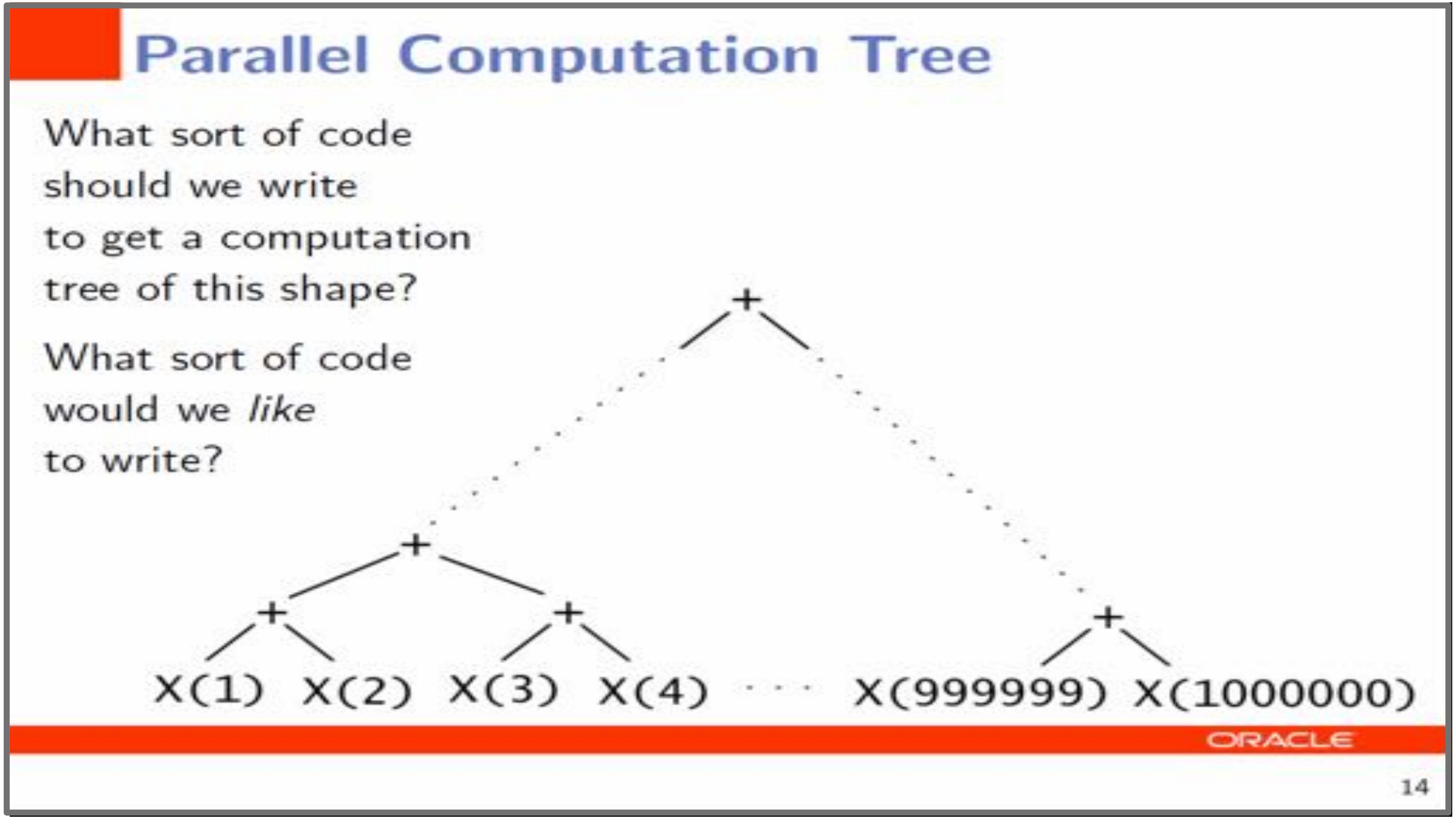


Sequential Computation Tree

```
SUM = 0
DO I = 1, 1000000
  SUM = SUM + X(I)
END DO
```

+
+
X(1000000)
X(999999)
+
+
X(3)
+
X(2)
0
X(1)

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

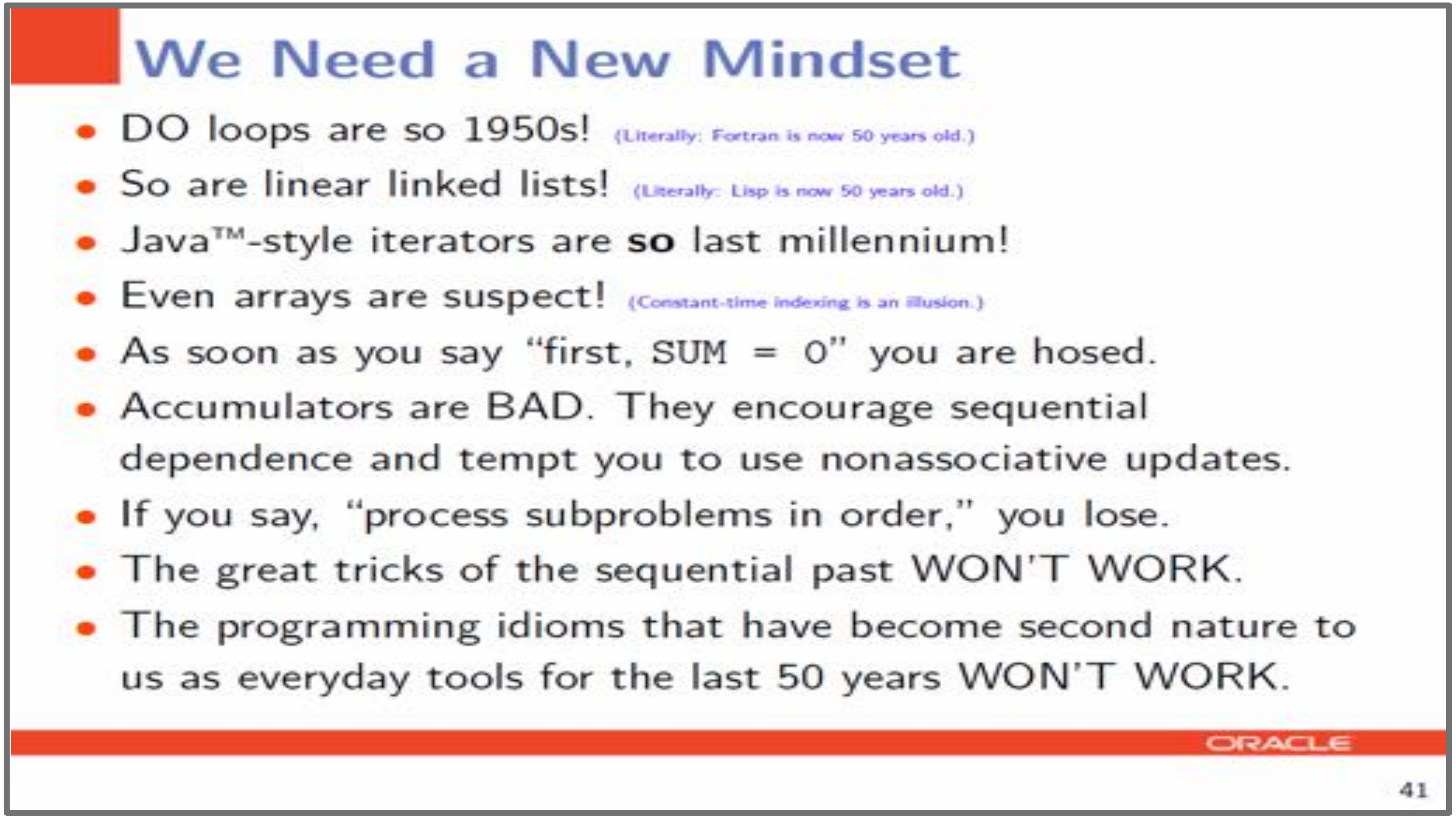


Parallel Computation Tree

What sort of code should we write to get a computation tree of this shape?

What sort of code would we *like* to write?

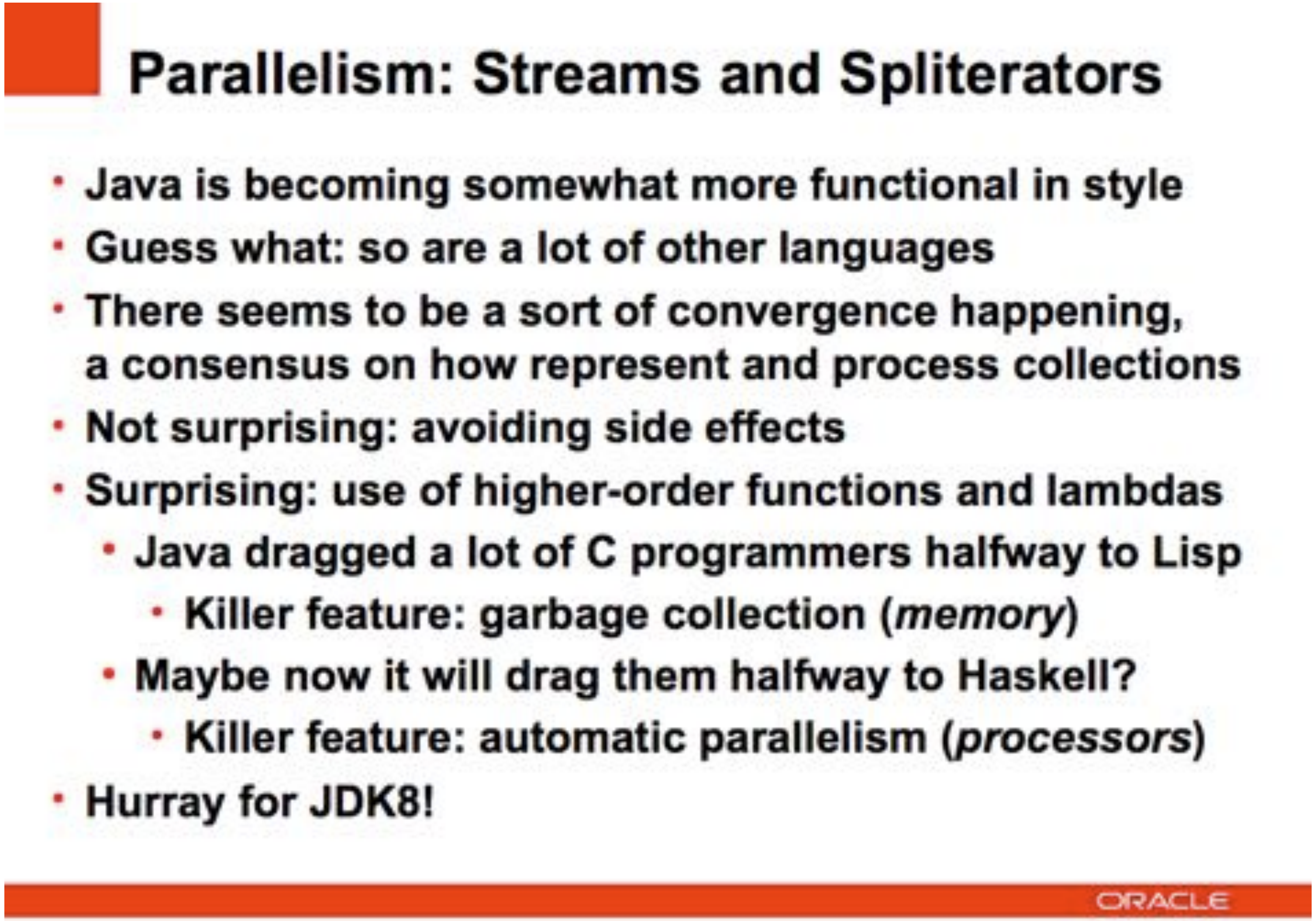
Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |



We Need a New Mindset

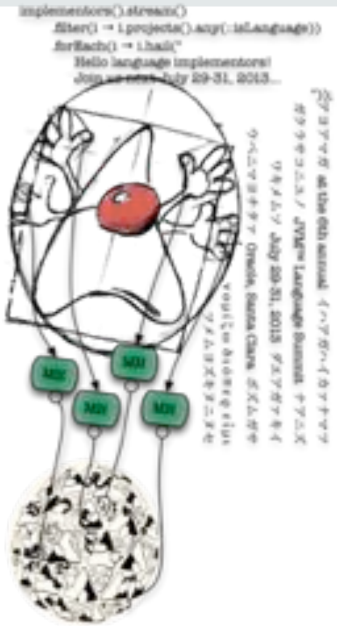
- DO loops are so 1950s! (Literally: Fortran is now 50 years old.)
- So are linear linked lists! (Literally: Lisp is now 50 years old.)
- Java™-style iterators are **so** last millennium!
- Even arrays are suspect! (Constant-time indexing is an illusion.)
- As soon as you say "first, `SUM = 0`" you are hosed.
- Accumulators are BAD. They encourage sequential dependence and tempt you to use nonassociative updates.
- If you say, "process subproblems in order," you lose.
- The great tricks of the sequential past WON'T WORK.
- The programming idioms that have become second nature to us as everyday tools for the last 50 years WON'T WORK.

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |



Parallelism: Streams and Spliterators

- Java is becoming somewhat more functional in style
- Guess what: so are a lot of other languages
- There seems to be a sort of convergence happening, a consensus on how represent and process collections
- Not surprising: avoiding side effects
- Surprising: use of higher-order functions and lambdas
  - Java dragged a lot of C programmers halfway to Lisp
    - Killer feature: garbage collection (*memory*)
  - Maybe now it will drag them halfway to Haskell?
    - Killer feature: automatic parallelism (*processors*)
- Hurray for JDK8!

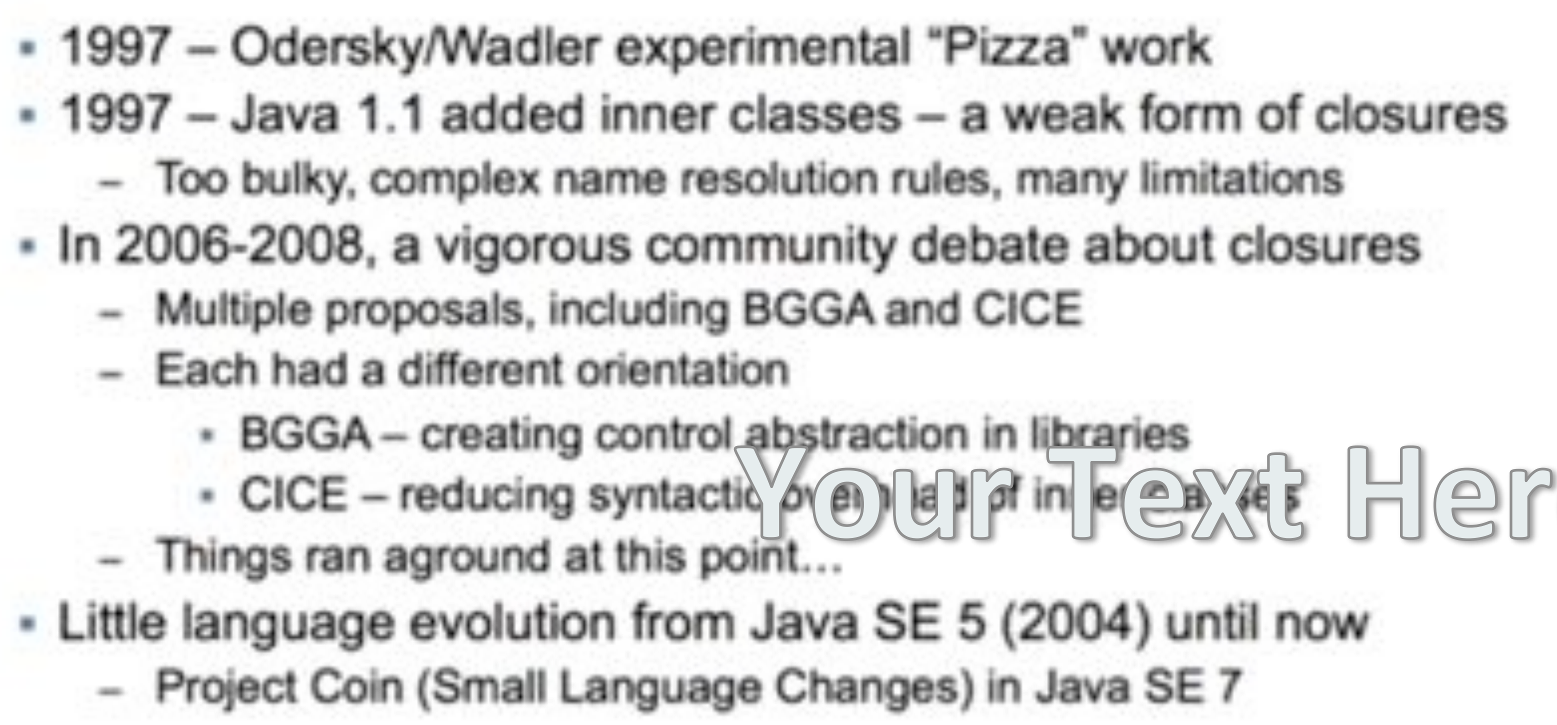


```
implementors().stream()
      .filter(i -> i.projects().any(::isLanguage))
      .forEach(i -> i.hail("
          Hello language implementors!
          Join us next July 29-31, 2013...
```

Guy Steele
JVM Language Summit 2013

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

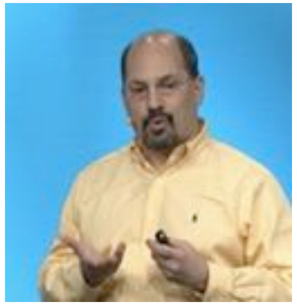
Closures / Lambdas for Java – a long and winding road…



- 1997 – Odersky/Wadler experimental "Pizza" work
- 1997 – Java 1.1 added inner classes – a weak form of closures
  - Too bulky, complex name resolution rules, many limitations
- In 2006-2008, a vigorous community debate about closures
  - Multiple proposals, including BGGA and CICE
  - Each had a different orientation
    - BGGA – creating control abstraction in libraries
    - CICE – reducing syntactic overhead of inner classes
  - Things ran aground at this point…
- Little language evolution from Java SE 5 (2004) until now
  - Project Coin (Small Language Changes) in Java SE 7

Your Text Here

- Dec 2009 – OpenJDK Project Lambda formed
- Nov 2010 – JSR-335 filed
  - Lambda Expressions + Interface Evolution + Bulk Collection Operations



Brian Goetz
JSR335 Spec Lead

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Lambdas In Java

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

The Problem: External Iteration

```
List<Student> students = ...
double highestScore = 0.0;
for (Student s : students) {
  if (s.gradYear == 2011) {
    if (s.score > highestScore) {
      highestScore = s.score;
    }
  }
}
```

- Our code controls iteration
- *Inherently serial:* iterate from beginning to end
- Not thread-safe because business logic is stateful (mutable accumulator variable)

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Internal Iteration With Inner Classes

More Functional, Fluent

```
List<Student> students = ...
double highestScore = students.
  filter(new Predicate<Student>() {
    public boolean op(Student s) {
      return s.getGradYear() == 2011;
    }
  }).
  map(new Mapper<Student,Double>() {
    public Double extract(Student s) {
      return s.getScore();
    }
  }).
  max();
```

- Iteration handled by the library
- Not inherently serial – traversal *may* be done in parallel
- Traversal *may* be done lazily – so one pass, rather than three
- Thread safe – client logic is stateless
- High barrier to use
  - Syntactically ugly

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Internal Iteration With Lambdas

```
SomeList<Student> students = ...
double highestScore = students.
           filter(Student s -> s.getGradYear() == 2011).
           map(Student s -> s.getScore()).
           max();
```

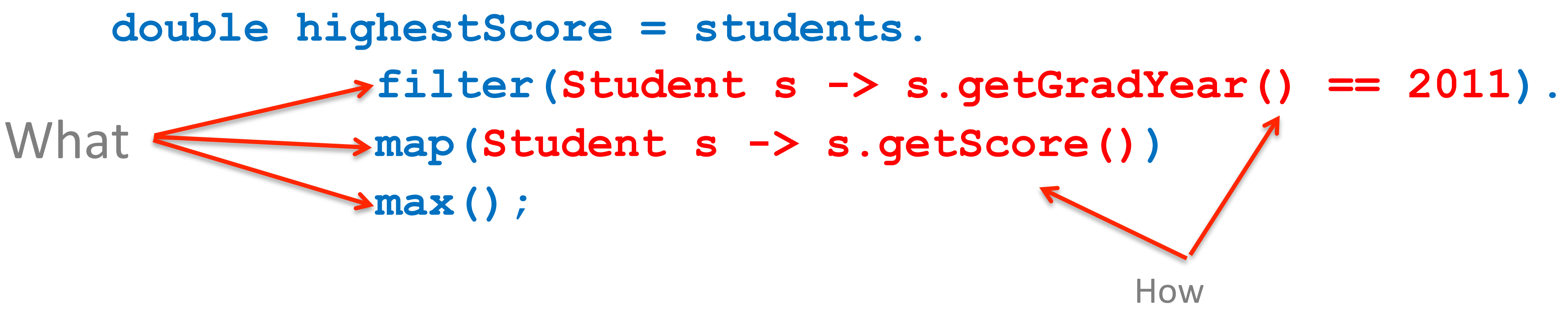
- More readable
- More abstract
- Less error-prone

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Lambda Expressions

Some Details

- Lambda expressions represent anonymous functions
  - Same structure as a method
    - typed argument list, return type, set of thrown exceptions, and a body
  - Not associated with a class
- We now have parameterised behaviour, not just values



```
double highestScore = students.
                      filter(Student s -> s.getGradYear() == 2011).
                      map(Student s -> s.getScore())
                      max();
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Library Evolution Goal

- Requirement: aggregate operations on collections
  - New methods required on Collections to facilitate this

```
int heaviestBlueBlock = blocks.
           filter(b -> b.getColor() == BLUE).
           map(Block::getWeight).
           reduce(0, Integer::max);
```

- This is problematic
  - Can’t add new methods to interfaces without modifying all implementations
  - Can’t necessarily find or control all implementations

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Solution: Extension Methods

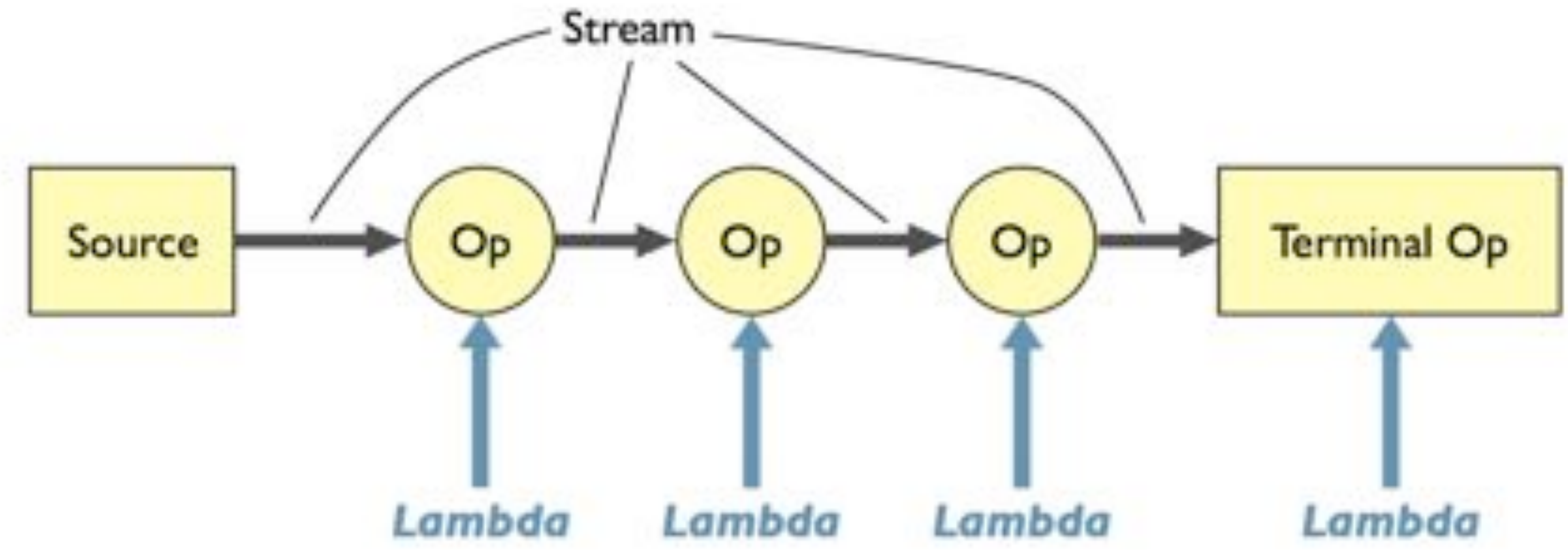
AKA Defender or Default Methods

- Specified in the interface
- From the caller’s perspective, just an ordinary interface method
- Provides a default implementation
  - Default only used when implementation classes do not provide a body for the extension method
  - Implementation classes can provide a better version, or not

```
interface Collection<E> {
  default Stream<E> stream() {
    return StreamSupport.stream(spliterator());
  }
}
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Lambdas In Full Flow: Streams



Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Aggregate Operations

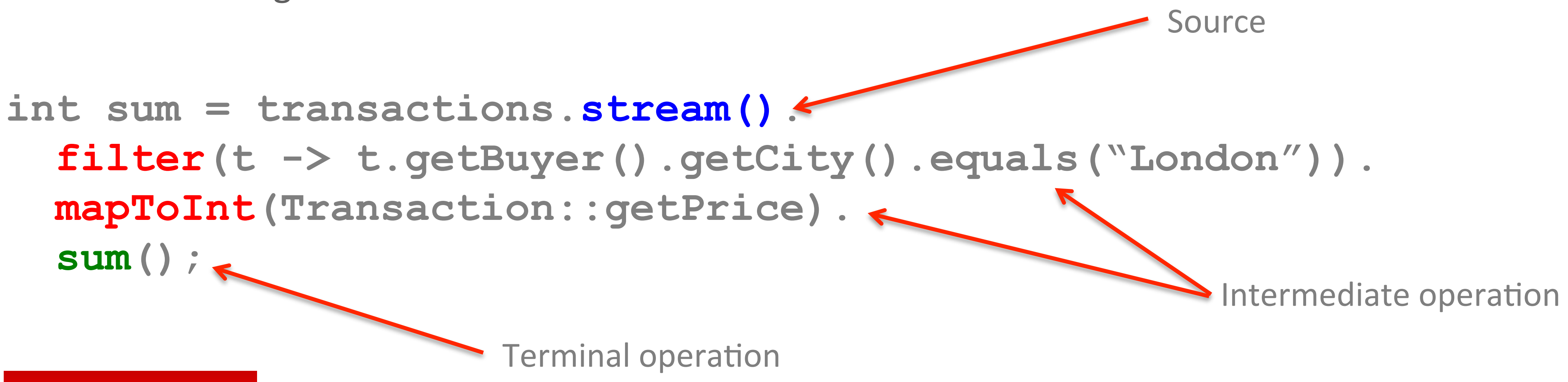
- Most business logic is about aggregate operations
  - “Most profitable product by region”
  - “Group transactions by currency”
- As we have seen, up to now, Java uses external iteration
  - Inherently serial
  - Frustratingly imperative
- Java SE 8’s answer: The `Stream` API
  - With help from Lambdas

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Stream Overview

Pipeline

- A stream pipeline consists of three types of things
  - A source
  - Zero or more intermediate operations
  - A terminal operation
    - Producing a result or a side-effect



```
int sum = transactions.stream().
  filter(t -> t.getBuyer().getCity().equals(“London”)).
  mapToInt(Transaction::getPrice).
  sum();
```

Source

Intermediate operation

Terminal operation

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Stream Sources

Many Ways To Create

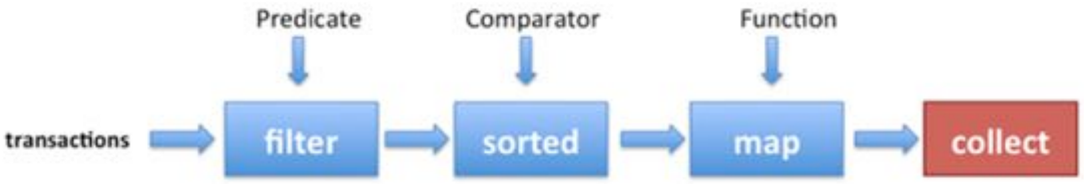
- From collections and arrays
  - `Collection.stream()`
  - `Collection.parallelStream()`
  - `Arrays.stream(T array)` or `Stream.of()`
- Static factories
  - `IntStream.range()`
  - `Files.walk()`
- Roll your own
  - `java.util.Spliterator`

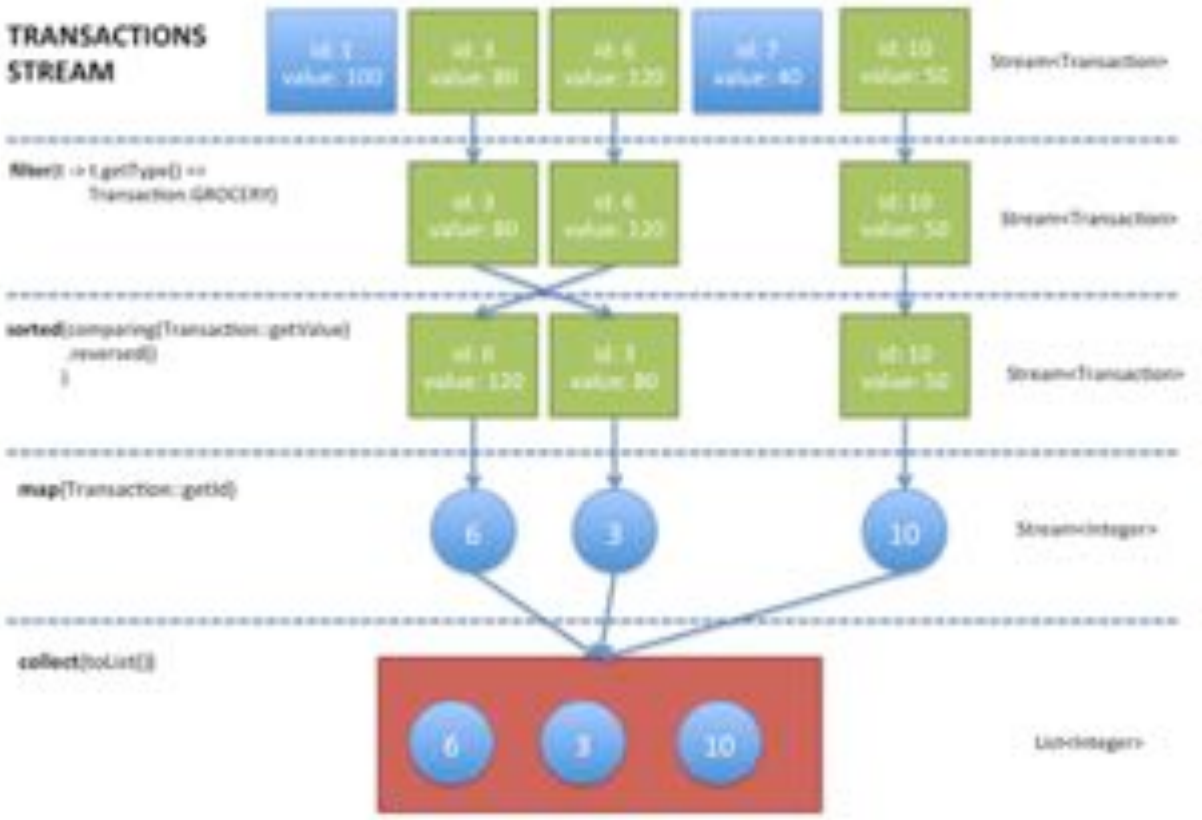
Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Streams

http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

```
List<Integer> transactionsIds =
    transactions.stream()
                .filter(t -> t.getType() == Transaction.GROCERY)
                .sorted(comparing(Transaction::getValue).reversed())
                .map(Transaction::getId)
                .collect(toList());
```





Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 1

Convert words in list to upper case

```
List<String> output = wordList.
  stream().
  map(String::toUpperCase).
  collect(Collectors.toList());
```

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 1

Convert words in list to upper case (in parallel)

```
List<String> output = wordList.
  parallelStream().
  map(String::toUpperCase).
  collect(Collectors.toList());
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 2

Find words in list with even length

```
List<String> output = wordList.
  parallelStream().
  filter(w -> (w.length() & 1 == 0).
  collect(Collectors.toList());
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 3

Count lines in a file

- BufferedReader has new method
  - `Stream<String> lines()`

```
long count = bufferedReader.
  lines().
  count();
```

ORACLE®

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 4

Join lines 3-4 into a single string

```
String output = bufferedReader.
  lines().
  skip(2).
  limit(2).
  collect(Collectors.joining());
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 6

Collect all words in a file into a list

```
List<String> output = reader.
  lines().
  flatMap(line -> Stream.of(line.split(REGEXP))).
  filter(word -> word.length() > 0).
  collect(Collectors.toList());
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Example 7

List of unique words in lowercase, sorted by length

```
List<String> output = reader.
  lines().
  flatMap(line -> Stream.of(line.split(REGEXP))).
  filter(word -> word.length() > 0).
  map(String::toLowerCase).
  distinct().
  sorted((x, y) -> x.length() - y.length()).
  collect(Collectors.toList());
```

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Conclusions

- Java needs lambda statements
  - Significant improvements in existing libraries are required
- Require a mechanism for interface evolution
  - Solution: virtual extension methods
- Bulk operations on Collections
  - Much simpler with Lambdas
- Java SE 8 evolves the language, libraries, and VM together

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Date And Time APIs

Developed and integrated via JSR 310

http://www.threeten.org

- 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

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

LocalDate	**2010-12-03**	
LocalTime	**11:05:30**	
LocalDateTime	**2010-12-03T11:05:30**	
ZonedDateTime	**2010-12-03T11:05:30+01:00 Europe/Paris**	
Instant	***2576458258.266 seconds after 1970-01-01***	
Duration	**PT30S**	***(30 seconds)***
Period	**P1Y6M**	***(1 year and 6 months)***

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

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, `jjs` to run JavaScript
- Internationalised error messages and documentation

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java Virtual Machines

- HotSpot and JRockit Convergence (and CDC)

  Remove permgen

  JIT Compilers (C1/C2 Tiered Compilation)

  GC improvements / G1 / Rationalisation

  Ergonomics

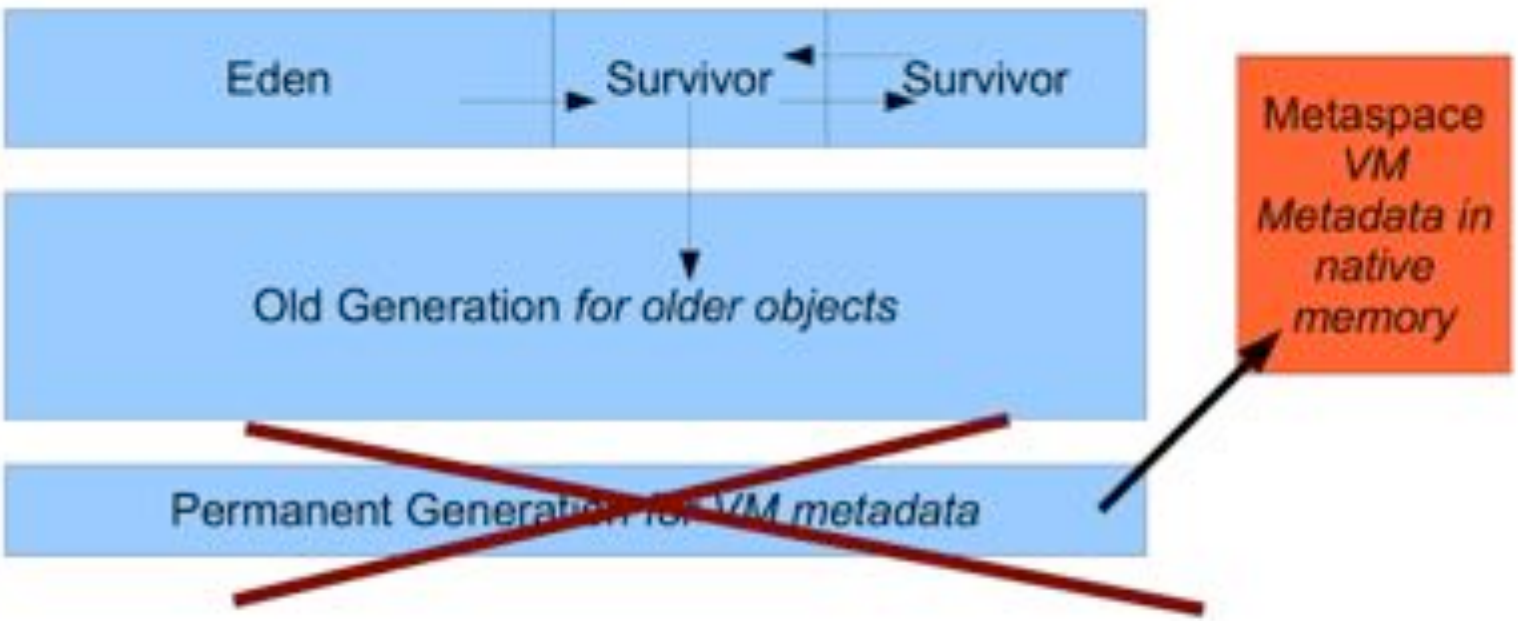
  Instrumentation / Tuning / Performance

  Multi language support (Indy, Nashorn…)

Isolation / Multi-Tenancy / Cloud

Low Latency

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |



Eden

Survivor

Survivor

Old Generation *for older objects*

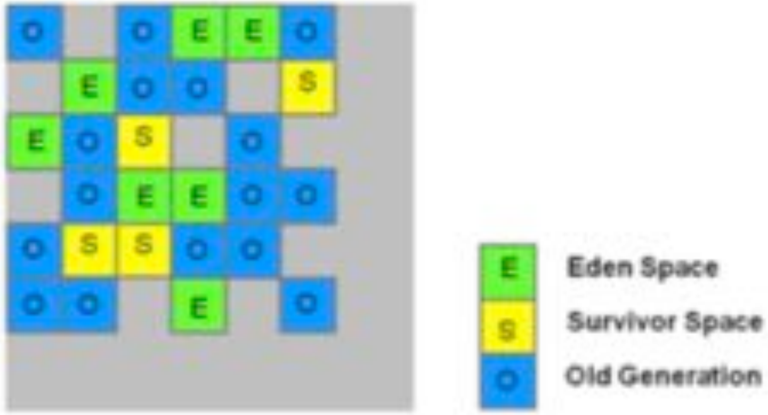
~~Permanent Generation *for VM metadata*~~

Metaspace *VM Metadata in native memory*

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

G1 – Garbage First

- Came from Sun Labs, been in development for the last few years
- Will replace CMS in some near-future release
- Officially supported as of 7u4
- Region based heap
  - Dynamic young generation sizing
  - Partial heap compaction using evacuation



- Pause target
  - Select number of regions in young and mixed collections that fits target
- Garbage First
  - Select regions that contain mainly garbage

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java SE Advanced (Commercial Product)

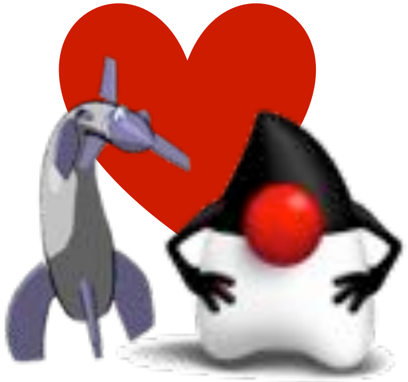
- Java Mission Control & Flight Recorder
  - Real-time profiling and diagnostics without performance overheads
- Enterprise JRE features
  - Usage tracking
  - auto update off

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

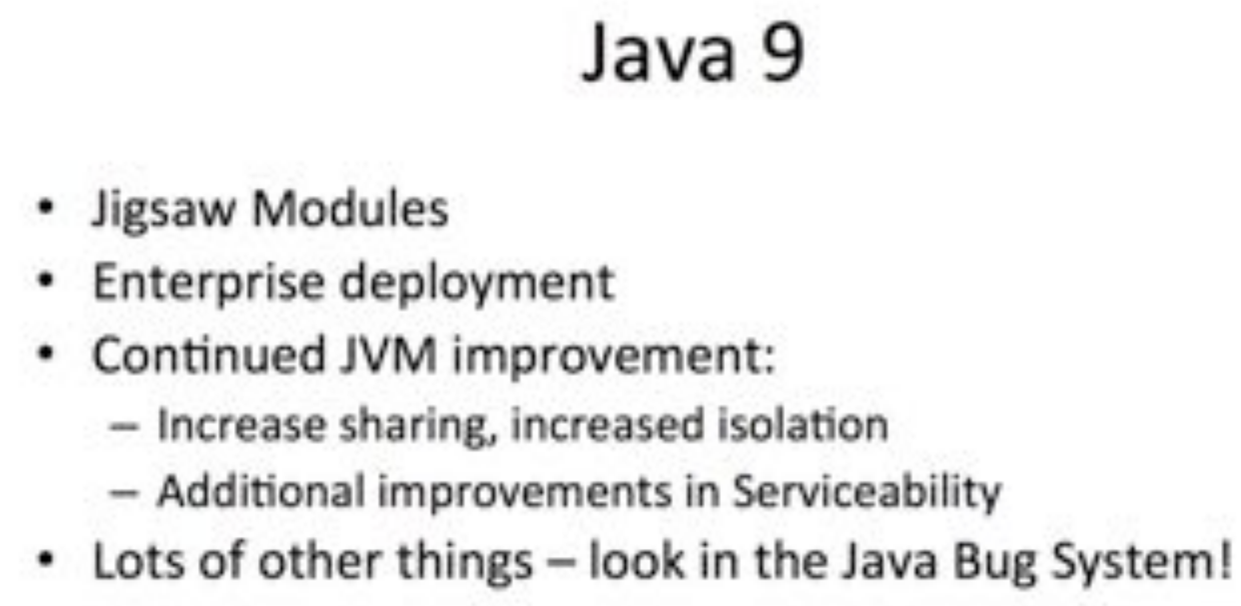
Java Mission Control

JVM Convergence

- The HotSpot version of the JRockit tools suite - JRockit Mission Control
- First release was JMC 5.2, released with JDK 7u40
- JMC 5.3 released with JDK 8
  - Minor release
  - Mostly bugfixes and incremental improvements
  - Now supports Eclipse 4.3.x
- Free for development use

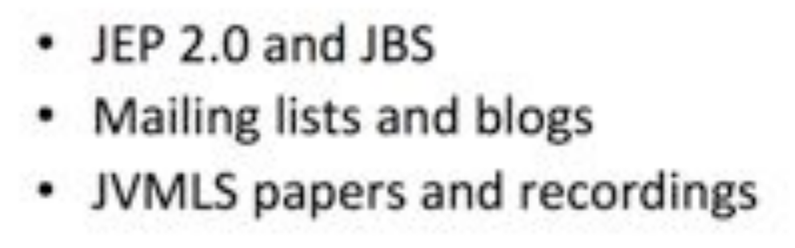


Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

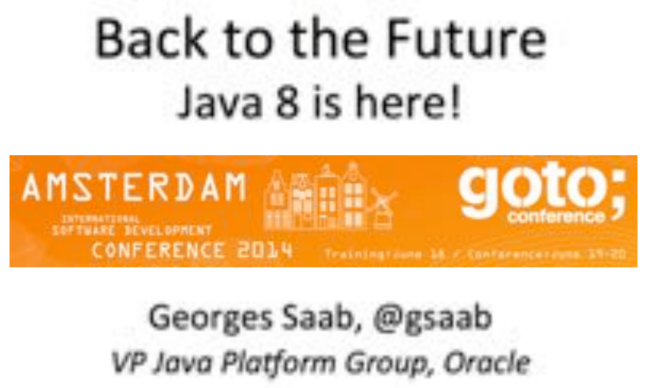


Java 9

- Jigsaw Modules
- Enterprise deployment
- Continued JVM improvement:
  - Increase sharing, increased isolation
  - Additional improvements in Serviceability
- Lots of other things – look in the Java Bug System!



- JEP 2.0 and JBS
- Mailing lists and blogs
- JVMLS papers and recordings



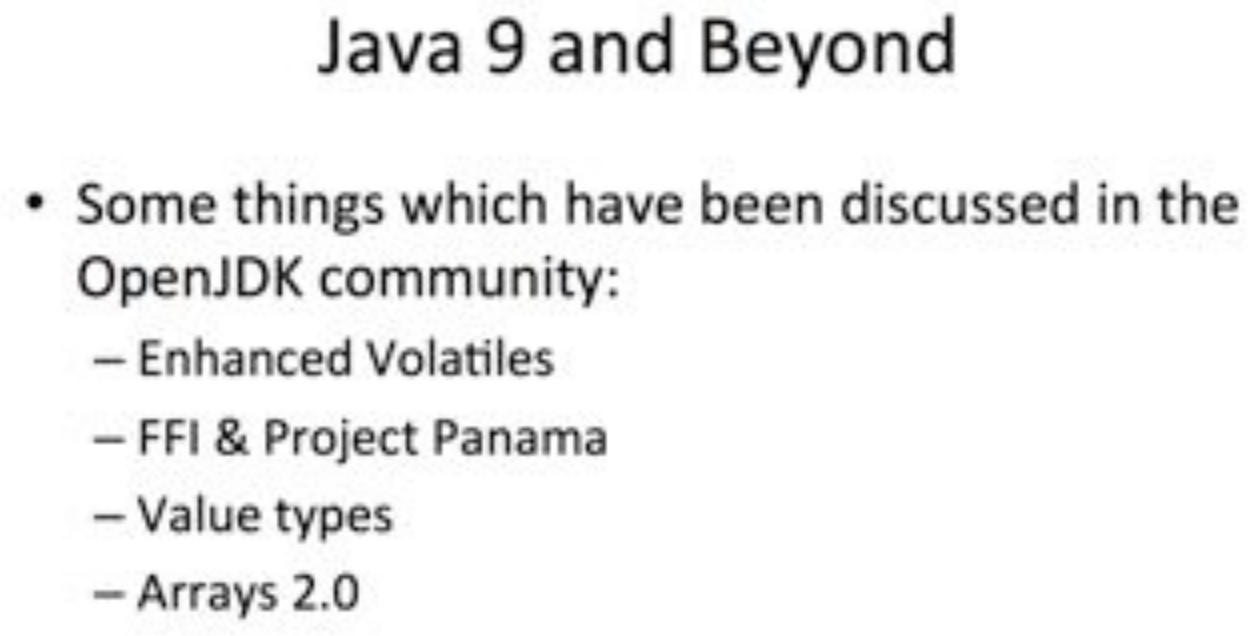
Back to the Future

Java 8 is here!

AMSTERDAM
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE 2014
goto; conference

Georges Saab, @gsaab

VP Java Platform Group, Oracle



Java 9 and Beyond

- Some things which have been discussed in the OpenJDK community:
  - Enhanced Volatiles
  - FFI & Project Panama
  - Value types
  - Arrays 2.0

Java 8

Learn More & Resources

- **Download:** java.oracle.com
- Documentation: docs.oracle.com/javase
- Training: education.oracle.com/java
- Java 8 Central: www.oracle.com/java8
- Java Magazine: www.oracle.com/javamagazine

@java @javaembedded

Facebook.com/ilovejava

Nighthacking.com

Youtube.com/java

blogs.oracle.com/java

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java 8 Launch Event (March 2014)

Videos - http://www.oracle.com/events/us/en/java8/index.html

Java 8 Launch

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

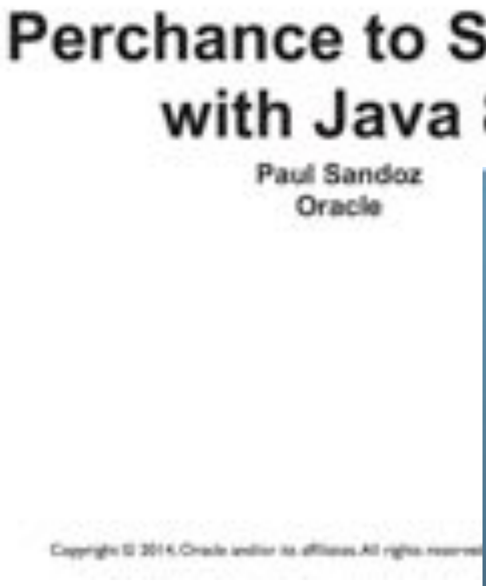
NetBeans IDE 8: Toolbox for Java SE 8
Geertjan Wielenga

Nashorn: JavaScript on the JVM
Jim Laskey

Java 8 Security Highlights
Milton Smith

Introducing Java Mission Control 5.3.0
Marcus Hirt

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |



Perchance to Stream with Java 8

Paul Sandoz
Oracle

Copyright © 2014 Oracle and/or its affiliates. All rights reserved.

Java

Fundamentals of GC Tuning

Charlie Hunt
JVM & Performance Junkie

CREATE
THE FUTURE

Lambda: A Peek Under the Hood

Brian Goetz

Jul 13

JavaOne

What's New in HotSpot JVM 8

Vladimir Ivanov
HotSpot JVM Compiler team
Oracle Corp.

MAKE THE
FUTURE
JAVA
ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

// MARCH/APRIL 2014 /

Java™ magazine

By and for the Java community

JAVA 8 EXPLORE THE POSSIBILITIES

IN THIS ISSUE: Lambda Expressions / Nashorn / Date and Time / Embedded Java

ORACLE.COM/JAVAMAGAZINE

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

https://blogs.oracle.com/java-platform-group

Java Platform Group, Product Management blog

Thoughts on Java SE, Java Security and Usability

« Welcome! | Main | Code signing: Unders... »

Introducing Deployment Rule Sets

By costlow on Aug 20, 2013

As the Java security model has hardened for browser-based applets, desktop administrators have asked for ways to manage version compatibility and security updates for their end-users.

A new feature is being introduced in Java 7 update 40 called "Deployment Rule Set," designed to address the issue of security and compatibility in browser applets without affecting normal back-end Java programs like Eclipse, Freemind, or Tomcat. Specifically this deployment rule set addresses two major points:

1. The desktop administrator's ability to control Java version compatibility, and default choices on the end-user's desktop. For example your users may use most recent security updates for most browser applets but still use an old Java 1.6 for that one legacy application that is no longer maintained.
2. The end-user's awareness of who created the application and their default interaction (ask, run, or block). By seeing the actual company or signer, the user is protected from running code by someone that they do not know. For example, I would trust "My University" or "Erik Costlow" but not "Unknown publisher" or someone else claiming to be me.

This feature is geared towards two types of users:

Desktop Administrators, who manage a number of users and need to control version compatibility and default dialogs to specific company applets. Desktop Administrators should learn how to control Java across these user systems. For example, "automatically run browser applets signed by our company" or "run all our browser applets with the latest secure version, except for this one internal system that we know needs Java 1.6."

Developers, who create Java applets and Web Start applications should be aware of the role that deployment rule sets play on their end-user's desktop.

How to create a deployment rule set

About

This blog contains topics related to Java SE, Java Security and Usability. The target audience is developers, sysadmins and architects that build, deploy and manage Java applications. Contributions come from the Java SE Product Management team.

Search

Enter search term:

☑ Search only this blog

Recent Posts

7u45 Caller-Allowable-Codebase and

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

The Pragmatic Programmers
Functional Programming in Java
Harnessing the Power of Java 8 Lambda Expressions
Venkat Subramaniam
Edited by Jacquelyn Carter

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Q & A

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Java™