Java Current and Future

Georges Saab
Vice President, Java Platform Group, Oracle
Chair, OpenJDK Governing Board

Mikael Vidstedt
Director, Java Virtual Machine

April 29, 2019

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.

Why is Java SE Important?

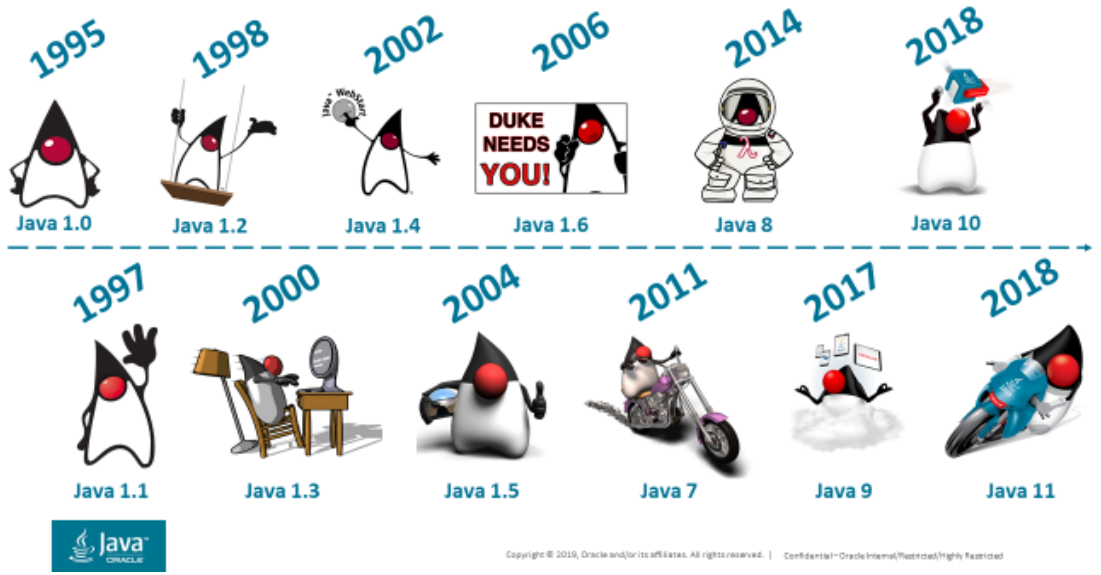
- **Ease of Use –** Language fundamentals based on improving C++ to provide powerful, simple to use language.
- **Reliability –** Java is strongly typed with heavily tested libraries and extensive tooling making Java robust.
- **Security –** Java's architecture, originally designed for embedded devices, was designed with security in mind.
- **Platform Independence –** Write once, run anywhere. OS/Hardware agnostic.

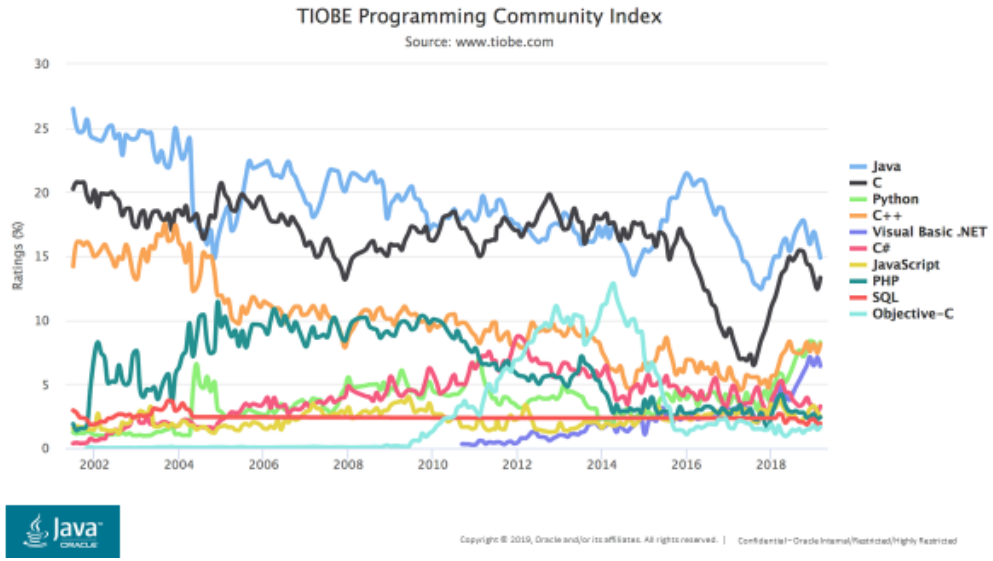
- **#1 programming language**
- **12 million developers run Java**
- **80%** of enterprises run Java SE on desktop, servers, and cloud deployments
- **38 billion** active Java Virtual Machines globally
- **#1 developer choice for cloud**

Java™ ORACLE

Java Continues to Thrive

- **Two Decades of Innovation –** Java continues to evolve with important language features such as Generics in Java 5, Lambdas in Java 8 and Modules in Java 9 – boosting performance, stability and security of the platform along the way.
- **Ready for the Cloud –** Release cadence changes introduced in 2017 mean important features are delivered faster. Many of Java's humble embedded roots make it ideal for Cloud – low memory foot print, fast startup and data isolation.





Brought to you by Oracle

- Leading Author of Java Technology
- Leading Sponsor of Java Ecosystem
- Driving platform innovation

OpenJDK

ORACLE Academy

Java Community Process Executive Committee Member

Java User Group

Java magazine

Issues fixed in JDK 11 per organization

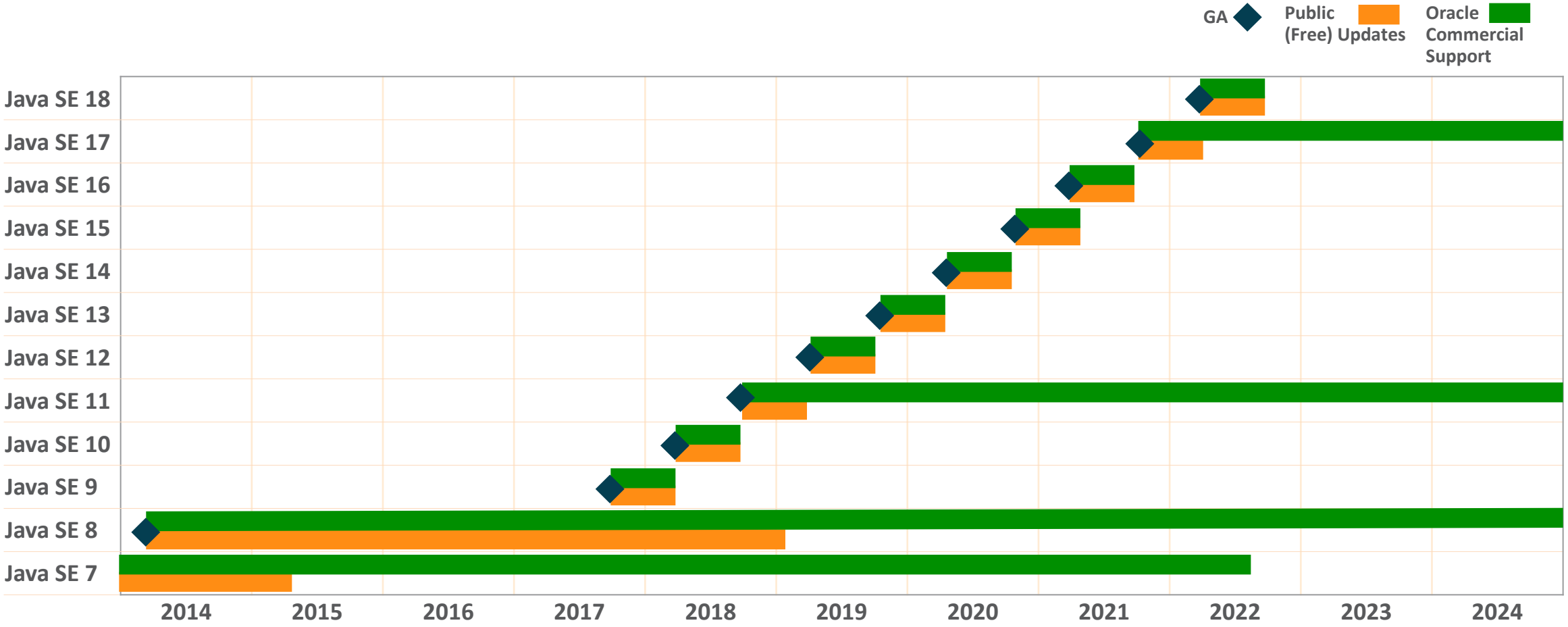
Java ORACLE

Innovating for the Future

- **Portola –** Keep Java #1 on cloud and modern devops by better leveraging containerized environments.
- **Panama –** Higher performance and easier development of I/O intensive applications through Java-native platform enhancements.
- **Valhalla –** Higher density and performance of machine learning and big data applications through the introduction of Value Types.
- **Loom –** Massively scale lightweight threads, making concurrency simply again.
- **Amber –** Continuously improve developer productivity through evolutions of the Java language.

Oracle Java SE Binaries Release Cadence

Supporting choice of migration paths - Every 6 months or every 3 years



Introducing Oracle Java SE Subscriptions

Don’t buy up front. Buy what you need, when you need it, only as long as you need.

- **Oracle Java SE Subscription –** $25/processor/month and lower with tiered discounts. License and support in a simple, single product. Easily buy online at the Oracle Store, or through Oracle Sales.
- **Oracle Java SE Desktop Subscription –** Same Oracle Java SE runtime, low desktop pricing. $2.50/users/month and lower with tiered discounting.

RedMonk

“The subscription model for updates and support has been long established in the Linux ecosystem. Meanwhile people are increasingly used to paying for services rather than products. It’s natural for Oracle to offer a monthly Java SE subscription to suit service-based procurement models for enterprise customers.”

– James Governor, RedMonk co-founder and analyst

Oracle Java SE Subscription

Comprehensive solution for the Java Powered Landscape

- Java SE Licensing and Support for use on **Desktops, Servers and Cloud deployments**
- Access to performance, stability and security updates direct from Oracle
  - Including bundled patch releases for interim fixes that can't wait!
- My Oracle Support, 24x7, 27 languages supported, fast fix turnarounds
- A Oracle Java SE Subscription gives you control of when and how you migrate to newer versions
- Access to all the enterprise management, monitoring and deployment features
- Support and access to all the new and continuously delivered innovations going forward

Java™ ORACLE

Java Current and Future

Mikael Vidstedt
Director, Java Virtual Machine

April 29, 2019

Panama

Loom

Metropolis

Valhalla

Skara

Amber

openjdk.java.net

Valhalla

http://openjdk.java.net/projects/valhalla/

Stateless?

```
final class Point {
    final int x;
    final int y;
}
```

Stateless? Not quite…

```
final class Point {
    final int x;
    final int y;
}


Point p = …
synchronized (p) {
   //
}
```

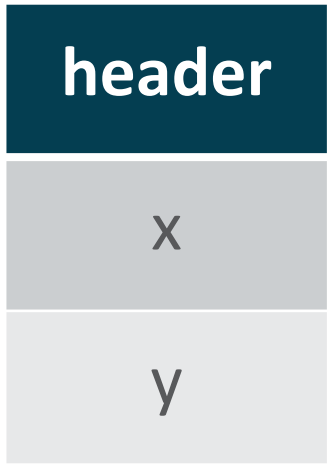
Stateless? Not quite…

```
final class Point {
    final int x;
    final int y;
}


Point p = …
synchronized (p) {
   // ...
}

// identity
p1 == p2
Objects.identityHashCode(p);
```

Java instance representation

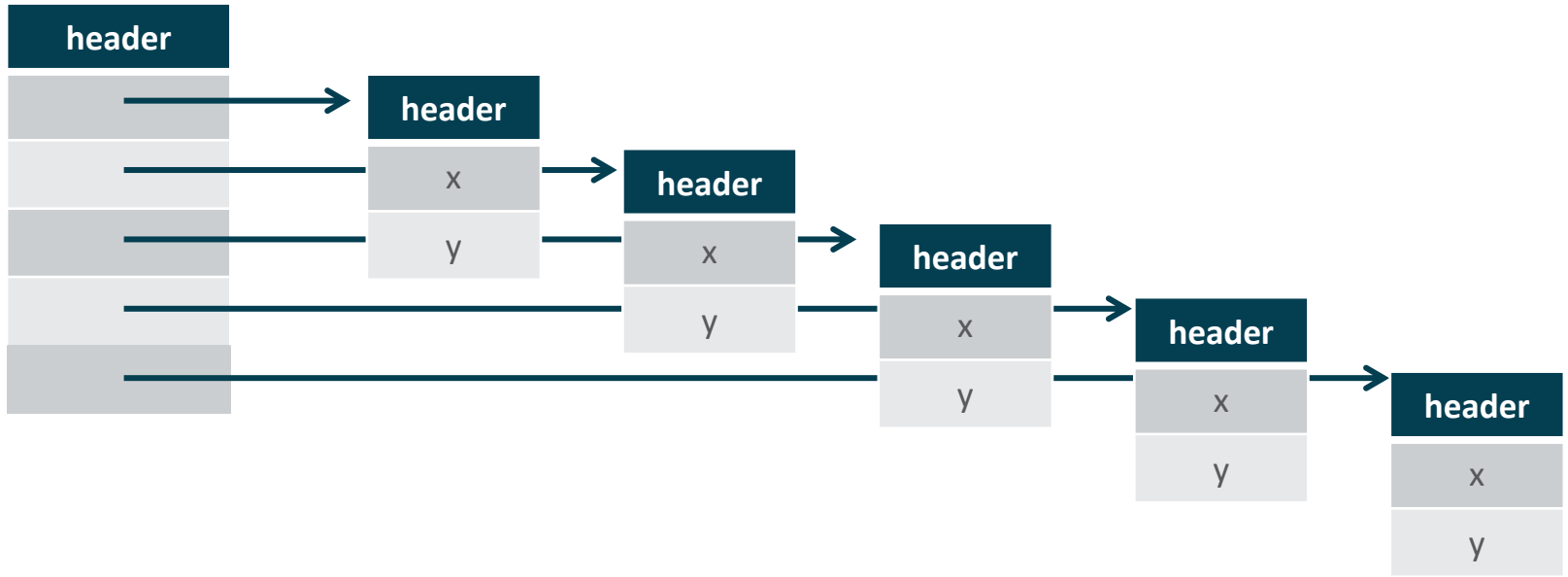


Valhalla

The data layout we have now

```
final class Point {
    final int x;
    final int y;
}

Point[] pts =
```

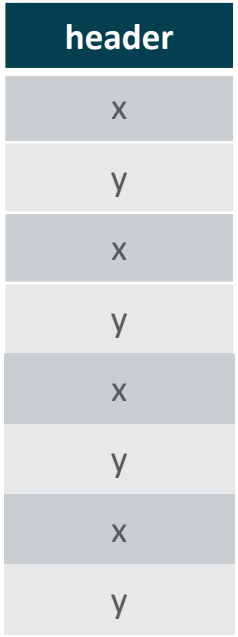


Valhalla

The data layout we want

```
__inline__ class Point {
    int x;
    int y;
}

Point[] pts =
```



Codes like a class, works like a long

Basically: Everything will get faster ☺

Valhalla: Performance

- Example: Matrix multiplication with complex elements

```
public class Complex {

    private final double re;
    private final double im;

    ...

    public Complex add(Complex that) {
        return new Complex(this.re + that.re,
                           this.im + that.im);
    }

    public Complex mul(Complex that) {
        return new Complex(
            this.re * that.re - this.im * that.im,
            this.re * that.im + this.im * that.re);
    }
}
```

Valhalla: Performance

- We can multiply these in the obvious way
  - But with lots of allocation and indirection

```
Complex[][] multiply(Complex[][] A, Complex[][] B) {
    int size = A.length;
    Complex[][] R = new Complex[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                Complex s = new Complex(0, 0);
                for (int k = 0; k < size; k++) {
                    s = s.add(A[i][k].mul(B[k][j]));
                }
                R[i][j] = s;
            }
        }
        return R;
    }
}
```

Valhalla: Performance

- JMH benchmark results (i7 laptop)
  - Take with appropriate skepticism

Metric	Boxed	Value	Factor
Time/op (ms)	3609	298	12.1
Allocation/op (MB)	3823	3.8	1006
Instructions	7.8G	2.5G	3.1
Instructions/cycle	1.02	2.63	2.6

Amber

https://openjdk.java.net/projects/amber/

Amber: Right-sizing Language Ceremony

- Set of Java language level features, incrementally delivered
- Features
  - Local Variable Type Inference (aka. the `var` keyword)
  - Switch expressions
  - Pattern matching
  - Multi-line and raw strings
  - Records and sealed types
  - ...

JEP 286: Local-Variable Type Inference (**JDK 10**)

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

try (InputStream is = socket.getInputStream();
    InputStreamReader isr = new InputStreamReader(is, charsetName);
    BufferedReader buf = new BufferedReader(isr)) {
    return buf.readLine();
}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

try (InputStream is = socket.getInputStream();
     InputStreamReader isr = new InputStreamReader(is, charsetName);
     BufferedReader buf = new BufferedReader(isr)) {
     return buf.readLine();
}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

try (InputStream is = socket.getInputStream();
     InputStreamReader isr = new InputStreamReader(is, charsetName);
     BufferedReader buf = new BufferedReader(isr)) {
     return buf.readLine();
}


// AFTER

try (var inputStream = socket.getInputStream();
     var reader = new InputStreamReader(inputStream, charsetName);
     var bufReader = new BufferedReader(reader)) {
     return bufReader.readLine();
}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE
void removeMatches(Map<? extends String, ? extends Number> map, int max) {




}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

void removeMatches(Map<? extends String, ? extends Number> map, int max) {
   for (Iterator<? extends Map.Entry<? extends String, ? extends Number>> iterator =
map.entrySet().iterator(); iterator.hasNext();) {
      Map.Entry<? extends String, ? extends Number> entry = iterator.next();
      if (max > 0 && matches(entry)) {
         iterator.remove();
         max--;
      }
   }
}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

void removeMatches(Map<? extends String, ? extends Number> map, int max) {
   for (Iterator<? extends Map.Entry<? extends String, ? extends Number>> iterator =
map.entrySet().iterator(); iterator.hasNext();) {
      Map.Entry<? extends String, ? extends Number> entry = iterator.next();
      if (max > 0 && matches(entry)) {
         iterator.remove();
         max--;
      }
   }
}
```

JEP 286: Local-Variable Type Inference (**JDK 10**)

```
// BEFORE

void removeMatches(Map<? extends String, ? extends Number> map, int max) {
   for (Iterator<? extends Map.Entry<? extends String, ? extends Number>> iterator =
map.entrySet().iterator(); iterator.hasNext();) {
      Map.Entry<? extends String, ? extends Number> entry = iterator.next();
      if (max > 0 && matches(entry)) {
         iterator.remove();
         max--;
      }
   }
}

// AFTER

void removeMatches(Map<? extends String, ? extends Number> map, int max) {
   for (var iterator = map.entrySet().iterator(); iterator.hasNext();) {
      var entry = iterator.next();
      if (max > 0 && matches(entry)) {
         iterator.remove();
         max--;
      }
   }
}
```

JEP 325: Switch Expressions (**Preview: JDK 12**)

```
javac --enable-preview --release 12 ...; java --enable-preview ...
```

JEP 325: Switch Expressions (**Preview: JDK 12**)

```
javac --enable-preview --release 12 ...; java --enable-preview ...
```

```
enum Day {
   MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
SUNDAY
}
```

JEP 325: Switch Expressions (**Preview: JDK 12**)

```
javac --enable-preview --release 12 ...; java --enable-preview ...
```

```
// BEFORE

int lettersInDayName(Day day) {
   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("…");
   }
   return numLetters;
}
```

```
enum Day {
   MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
SUNDAY
}
```

JEP 325: Switch Expressions (**Preview: JDK 12**)

`javac --enable-preview --release 12 ...; java --enable-preview ...`

```
// BEFORE

int lettersInDayName(Day day) {
   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("…");
   }
   return numLetters;
}
```

```
// AFTER

int lettersInDayName(Day day) {

   return switch (day) {
      case MONDAY, FRIDAY, SUNDAY -> 6;
      case TUESDAY -> 7;
      case THURSDAY, SATURDAY -> 8;
      case WEDNESDAY -> 9;
   };
}




enum Day {
   MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
SUNDAY
}
```

JEP 325: Switch Expressions (**Preview: JDK 12**)

`javac --enable-preview --release 12 ...; java --enable-preview ...`

```
// BEFORE

int lettersInDayName(Day day) {
   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("…");
   }
   return numLetters;
}
```

```
// AFTER

int lettersInDayName(Day day) {

   return switch (day) {
      case MONDAY, FRIDAY, SUNDAY -> 6;
      case TUESDAY -> 7;
      case THURSDAY, SATURDAY -> 8;
      case WEDNESDAY -> 9;
   };
}
```

```
enum Day {
   MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY,
SUNDAY
}
```

JEP 305: Pattern Matching (**Future**)

```
// BEFORE

if (o instanceof Rectangle) {
    Rectangle rectangle = (Rectangle)o;
    use(rectangle);
}
```

JEP 305: Pattern Matching (**Future**)

```
// BEFORE

if (o instanceof Rectangle) {
    var       rectangle = (Rectangle)o;
    use(rectangle);
}
```

JEP 305: Pattern Matching (**Future**)

```
// BEFORE

if (o instanceof Rectangle) {
    var       rectangle = (Rectangle)o;
    use(rectangle);
}



// AFTER

if (o instanceof Rectangle rectangle) {
    use(rectangle);
}
```

JEP <tbd>: Records and Sealed Types (**Future**)

```
// BEFORE

class Point {
    private final int x, y;

    public Point(int x, int y) { ... }

    public int getX() { ... }

    public int getY() { ... }

    public int hashCode() { ... }
    public int equals(Object o) { ... }
    public int compareTo(T o) { ... }
    ...
}
```

JEP <tbd>: Records and Sealed Types (**Future**)

```
// BEFORE

class Point {
    private final int x, y;

    public Point(int x, int y) { ... }

    public int getX() { ... }

    public int getY() { ... }

    public int hashCode() { ... }
    public int equals(Object o) { ... }
    public int compareTo(T o) { ... }
    ...
}
```

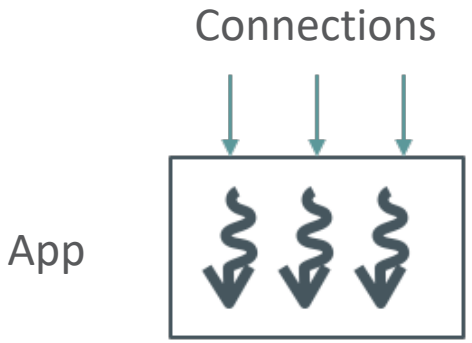
```
// AFTER

record Point(int x, int y) { }
```

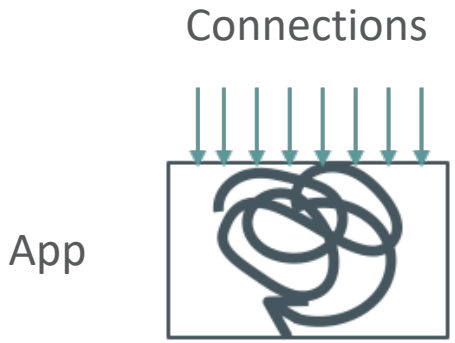
Loom

https://openjdk.java.net/projects/loom/

Scalability

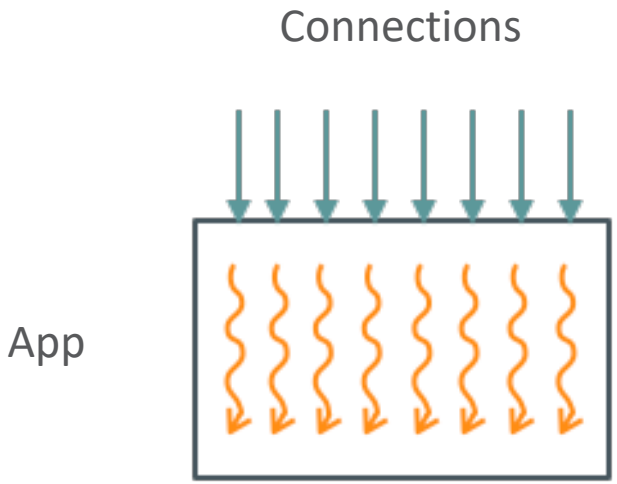


- Sync
  - Simple
  - Less scalable



- Async
  - Scalable
  - Complex
  - Non-interoperable
  - Hard to debug/profile

Loom



- Making blocking calls virtually free
  - Fibers
    - Lightweight user-mode threads
  - Delimited continuations
  - Tail-call elimination

- **Codes Like Sync, Works Like Async**
- **Concurrency Made Simple**

Panama

https://openjdk.java.net/projects/panama/

Panama

- Sometimes you just have to “go native”
  - Off-CPU computing (Cuda, OpenCL)
  - Deep learning (Blas, cuBlas, cuDNN, Tensorflow, ...)
  - Graphics processing (OpenGL, Vulkan, DirectX)
  - Others (OpenSSL, SQLite, V8, ...)

Panama

- Safe, Fast, Simple access to “foreign” functions and data
  - Foreign: not Java code/heap data
  - Call native functions/libraries
  - Access off-heap data
- **Without having to write a single line of native code**
- Think: JNI replacement

JNI

JNI

Panama

- Tool generates Java classes
- Calling native code indistinguishable from calling normal Java methods
- New APIs for operating on off-heap data

Metropolis

https://openjdk.java.net/projects/metropolis/

Metropolis: “Java-on-Java”

- Implement more of the JVM in Java
  - Starting with the JIT compiler
- Currently: C1 ”client” + C2 “server” optimizing compiler
  - C2 reaching its complexity budget
  - Many (most) projects have significant impact on the JIT compiler
    - Valhalla, Panama, …

Graal

- JIT compiler implemented in Java
- Lower barrier to entry
- More flexible, easier to experiment

Skara

Modernizing the developer experience for the JDK itself

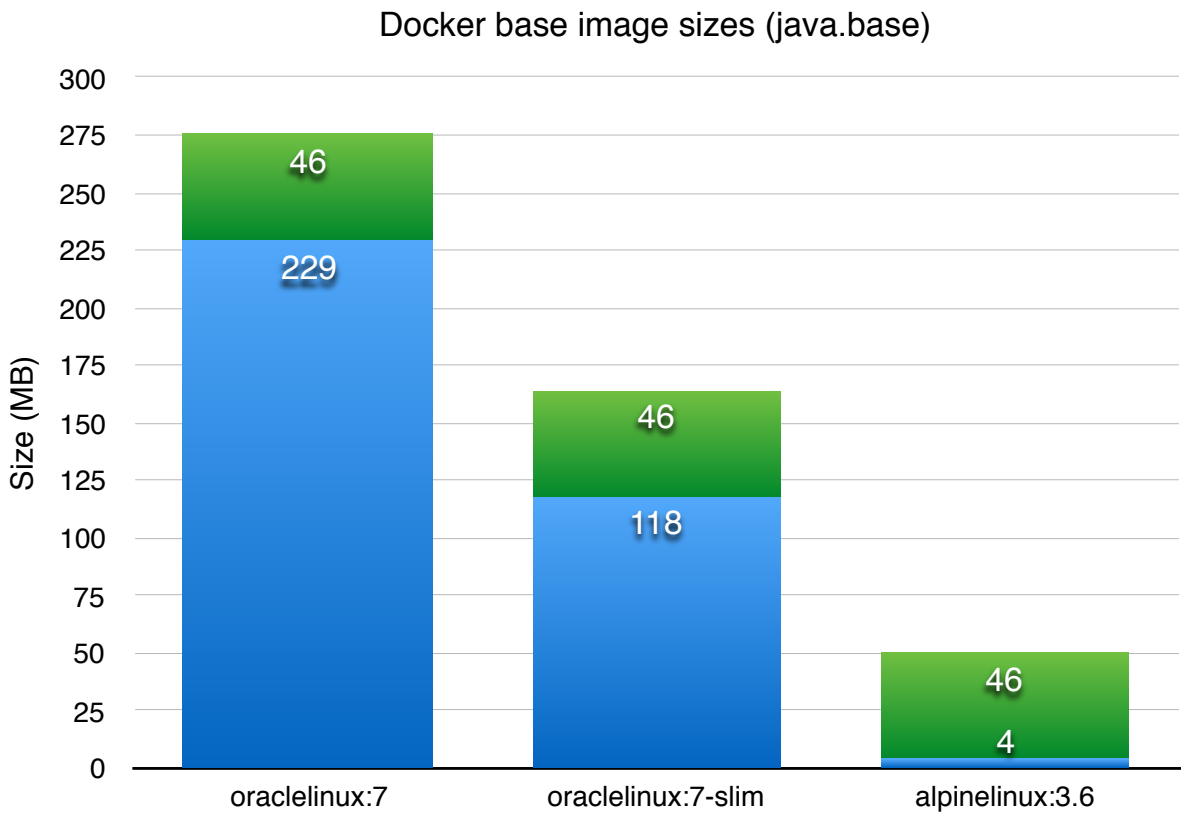
- Replacing the aging OpenJDK code infrastructure
  - SCM and code review
- Investigating
  - Moving to git (currently: mercurial/hg)
  - Leveraging hosted options (e.g. GitLab, GitHub, Bitbucket, etc.)

https://openjdk.java.net/projects/skara/

Portola: Java in a World of Containers

- Container resource awareness
  - Java heap & internal structures
- Port of the JDK to Alpine/musl
- The Alpine Linux base image weighs in at **4MB**
  - Uses the “musl” C library

https://openjdk.java.net/projects/portola/



ZGC: A Scalable Low-Latency Garbage Collector

Experimental (JDK 11+): `java -XX:+UnlockExperimentalVMOptions -XX:+UseZGC …`

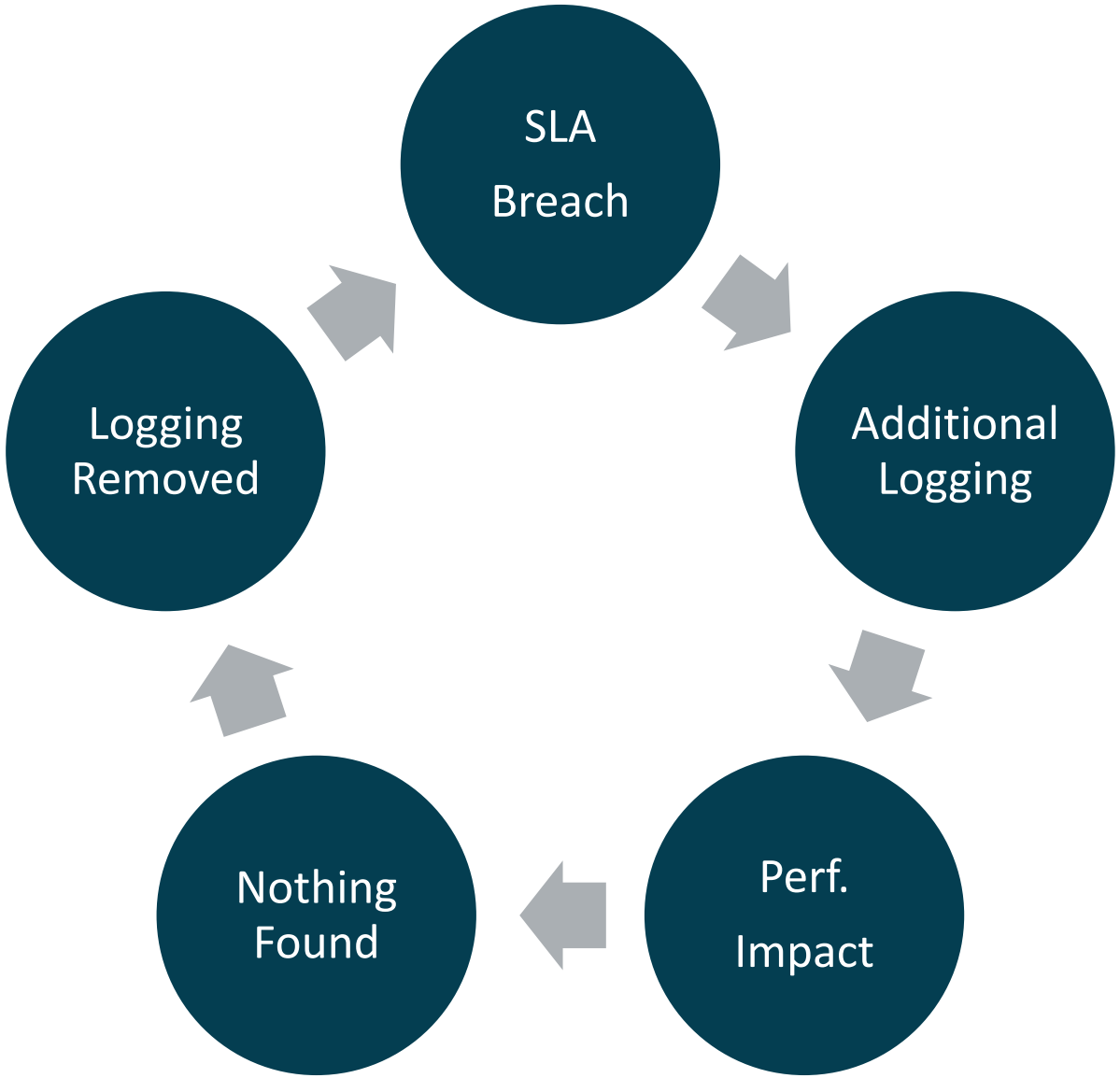
TB Up to multi-terabyte heaps

10ms Max GC pause time

Easy to tune

15% Max application throughput reduction

https://openjdk.java.net/projects/zgc/



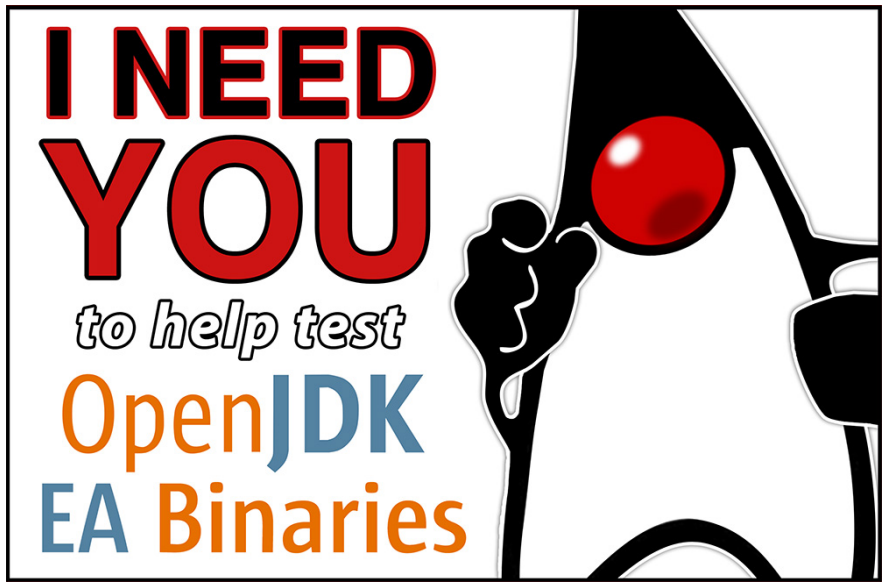
Flight Recorder (**JDK 11+**)

```
java -XX:+FlightRecorder ...
```

- Fine-grained event logging
  - Garbage collection, JIT compilation, locks, class loading, …
  - Events: Java API allows adding custom, user-defined events
- Designed to be **always on in production**
  - Negligible (low single-digit %) overhead
  - Piggy-backing on existing JVM functionality

We need your feedback!

Project	Early-Access (EA) Binaries
JDK 13	http://jdk.java.net/13/
Valhalla	http://jdk.java.net/valhalla/
Panama	http://jdk.java.net/panama/
jpackage	http://jdk.java.net/jpackage/




Java™
ORACLE®