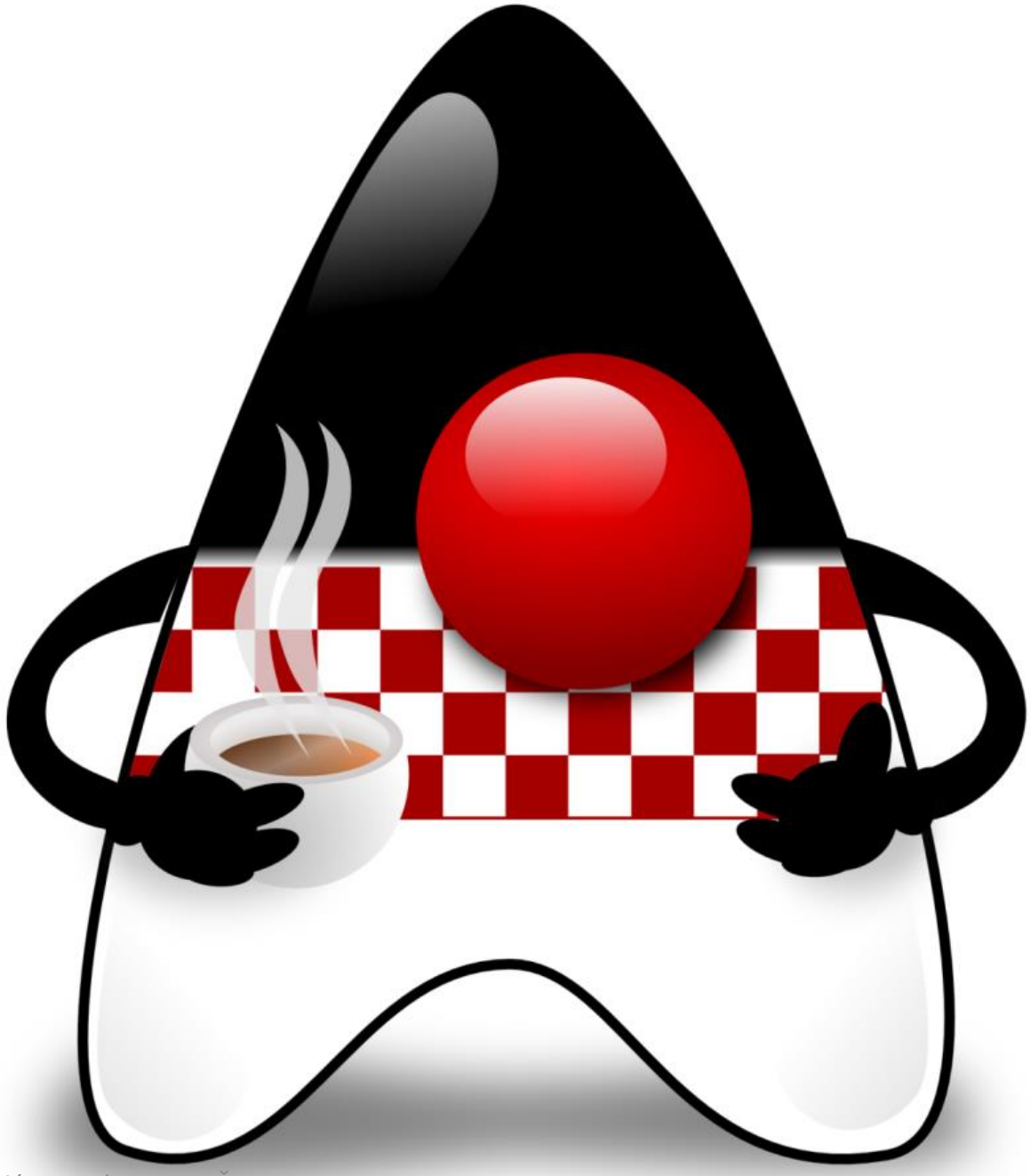
The State of Java and Software Development (in Croatia)

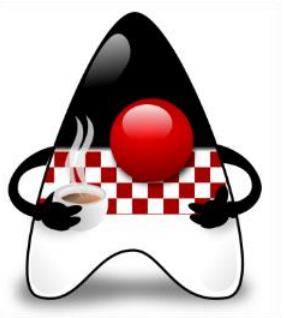
dr. sc. **Branko Mihaljević**

Aleksander Radovan

doc. dr. sc. **Martin Žagar**

HUJAK





Current State of Java?

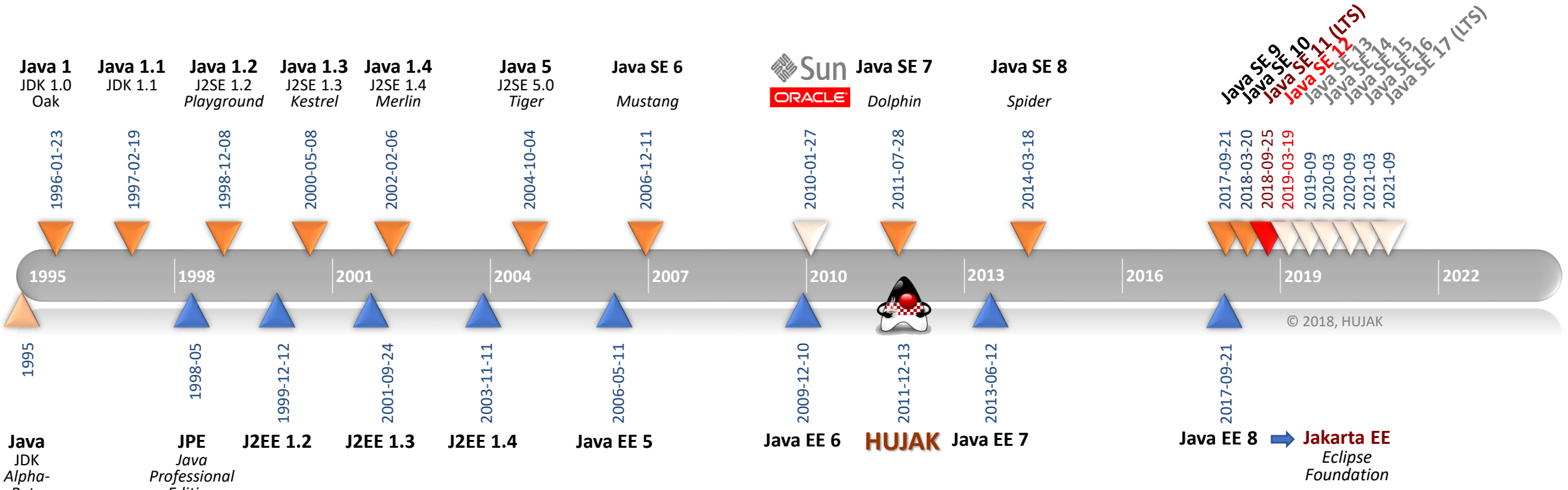
- Are you still using **Java 8**?
- Did you use **Java 9 / 10**?
- Maybe – you started using the **LTS** version **Java 11**?
- Or – upgraded to the latest **Java 12**?
- What about ~~Java~~ **Jakarta EE**?
- Well… let's explain – in our humble opinion

Java Platform today is:

Stable **Secure** **Free** ?

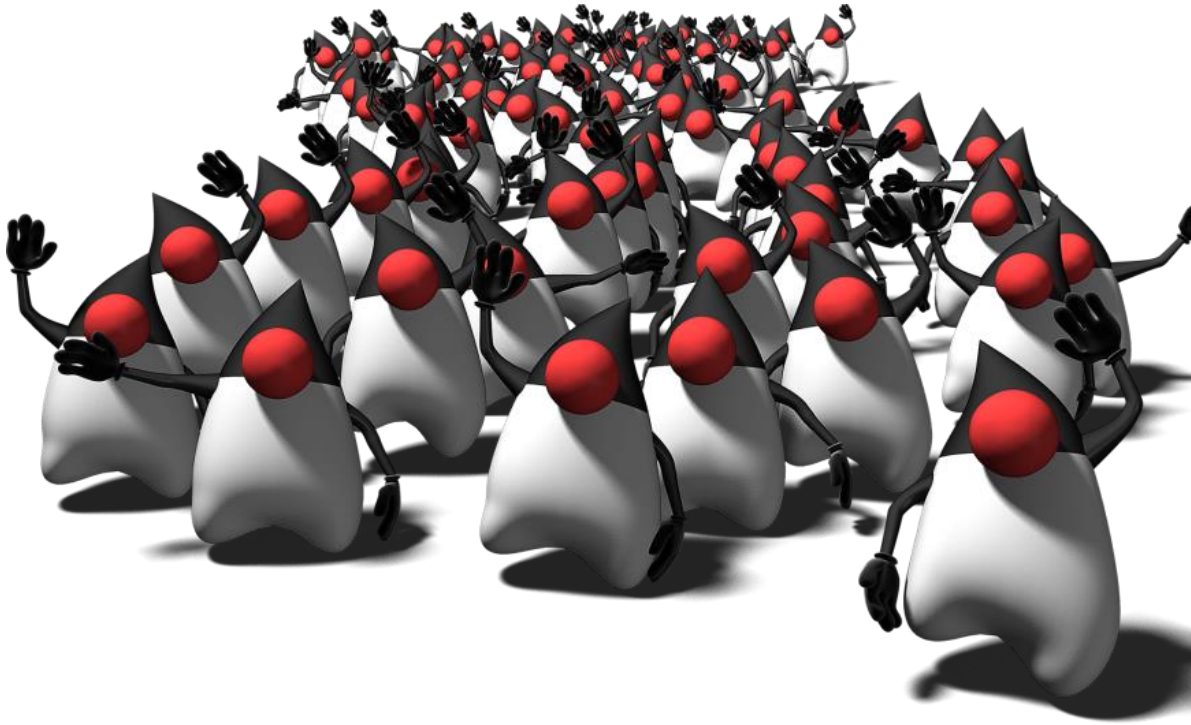
However, commonly choose two out of three

"Moving Java Forward Faster"

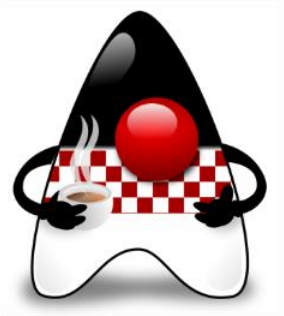


So, what is Java … for us, developers?

- **#1 Development Platform**
  - **Continued growth of Java** for **23+ years**
- **A Few Dozen Billion Devices** run **Java**
- **10 Million Java Developers** in
  - Many have **Java Certificates**



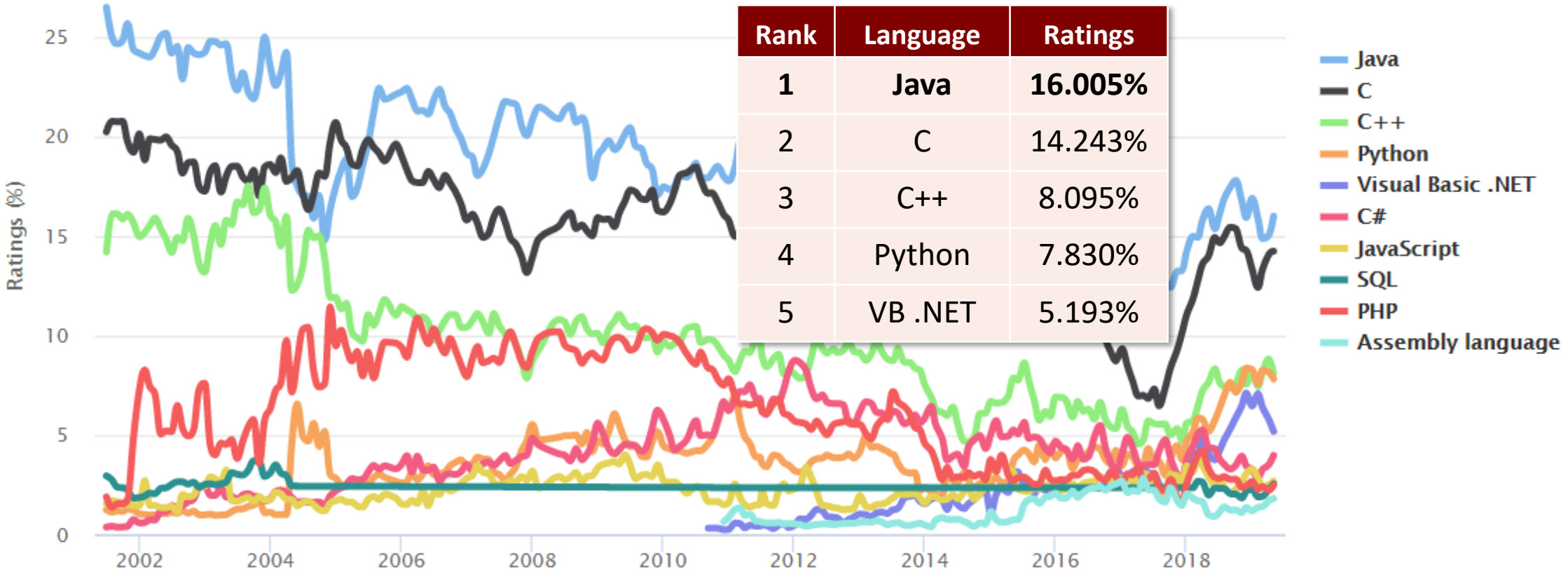
- But not only Java – **50+ JVM languages** (or even **more with GraalVM**)
  - Including **Clojure, Groovy, Scala, JRuby, Jython, Fantom, Kotlin, Ceylon, Xtend, X10, LuaJ, Golo, Frege, Mirah, Eta**… and **JavaScript**



How is Java currently holding?

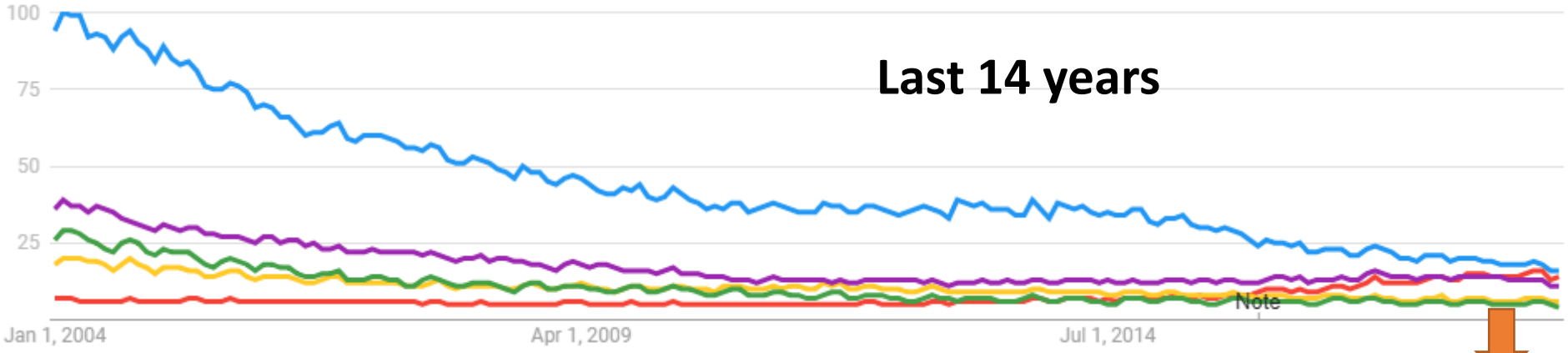
- **TIOBE index** for May 2019

Rank	Language	Ratings
1	**Java**	**16.005%**
2	C	14.243%
3	C++	8.095%
4	Python	7.830%
5	VB .NET	5.193%

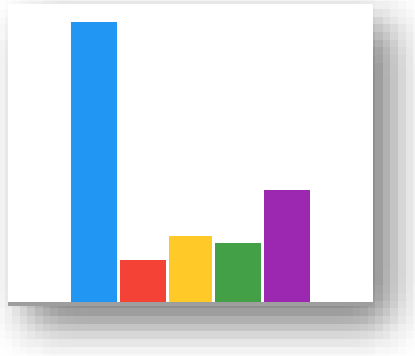


What about historical trends?

- Google Trends – Java vs Python vs C vs C++ vs JavaScript

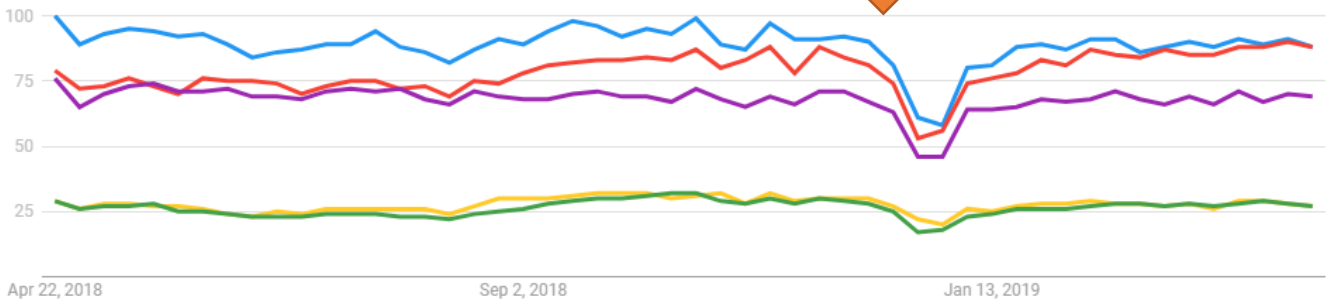


Last 14 years



Average

Last year (2018/2019)



What about Java download?

- When you type "Java download" in Google you'll probably get **www.java.com**
- And you can download the "latest" (?!) JRE **Java 8 Update 211**
- What about the **latest JDK download(s)**?

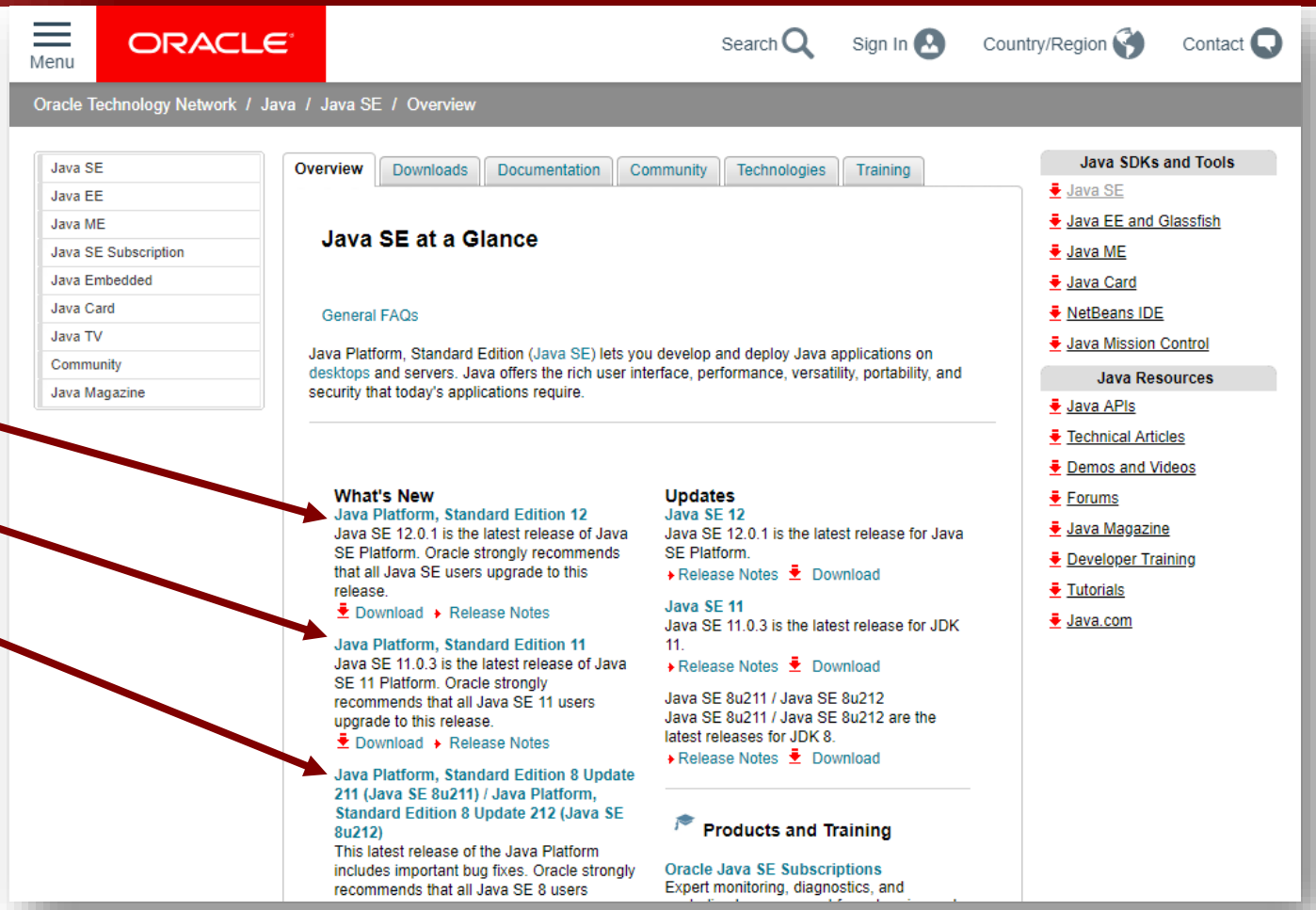
Available JDKs (and Licenses)

- **Oracle JDK** www.oracle.com/technetwork/java/javase/downloads/
  - **Oracle Binary Code License (BCL)** with FoU (Field of Use) restrictions **$$$**
- Many **OpenJDKs:**
- **Oracle OpenJDK** jdk.java.net
  - **GNU General Public License version 2**, with the Classpath Exception (**GPLv2cpe**)
- **AdoptOpenJDK's OpenJDK** adoptopenjdk.net
  - OpenJDK **8** (LTS), OpenJDK **11** (LTS) or OpenJDK **12** on **Hotspot** JVM or **OpenJ9** JVM (former IBM commercial JVM, open-sourced to Eclipse foundation)
- **Azul's Zulu OpenJDK** www.azul.com/downloads/zulu/
  - From JDK **6** to JDK **12**, wide platform support (Windows, Linux, macOS…)
- Others: **Amazon's Corretto OpenJDK**, **RedHat's OpenJDK**, **SAP's SapMachine OpenJDK**, **Linux distribution's OpenJDKs** …

OpenJDK or Commercial JDK?

Java Download at Oracle

- **Oracle's Java SE Downloads**
- Currently available downloads of Oracle's JDK:
  - Java SE **12.0.1**
  - Java SE **11.0.3**
  - Java SE **8u211**



JDK 9 – very old news?

- **JDK 9 was in General Availability in September 2017**
  - **109 new features** and **APIs!**
- **Java Platform Module System (JPMS)**
  - All core Java libraries are now modules (JEP 220)
  - 97 modules: 28 Java SE, 8 JavaFX, 59 JDK, 2 Oracle…
- **Encapsulation**
  - Most internal APIs encapsulated (JEP 260)
  - Finding encapsulated API use
  - jdeps – analyses dependencies on APIs
- **New release model**
- **90 JEPs included …**

BTW, how did we do JPMS migration?

- The easy way to **migrate application** to **JPMS**

a) Leave everything on the classpath ☺

b) Put everything on the classpath in the **unnamed module**
   - All public packages exported, this unnamed module depends on all modules

c) **Migrate to modules** as required
   - Automatic modules
   - Move existing jar files from classpath to modulepath

- And that should be it
- ... or not

OpenJDK Release Model

- **New Features** included (only) **when ready**
- **Feature release versions** released every **6 months** (in **March** & **September**)
- **Update releases** shipped **quarterly** (in **January**, **April**, **July**, and **October**)
- **Long-term support (LTS) feature release** every **3 years**
  - LTS for all releases is **not practical**
  - Starting with JDK 11, updates available **for at least 3 years**
  - LTS Plan: **JDK 11** (September 2018), **JDK 17** (September 2021), then **JDK 23**…
  - For Oracle's **commercial customers** updates available **for at least 3 years** or longer
- **Time-Based Release Versioning** (JEP 322) openjdk.java.net/jeps/322
  - Revise the version-string scheme of the Java SE Platform and the JDK

JDK Version Numbering

- **$FEATURE.$INTERIM.$UPDATE.$EMERG**
  - **$FEATURE** is **incremented** every **six months**
    - Previously MAJOR
    - JDK **11** in September 2018, JDK **12** in March 2019, JDK **13** in September 2019…
  - **$INTERIM** is always **zero**, reserved for flexibility and future use
    - Previously MINOR
  - **$UPDATE** is **incremented** every **three months**
    - The first one is **one month** after $FEATURE, previously SECURITY
    - JDK **12.0.1** in April 2019, JDK **12.0.2** in July 2019, JDK **13.0.1** in October 2019…
  - **$PATCH** is emergency patch-release counter
    - Outside of planned schedule, incremented only when it's necessary to fix a critical issue

JDK 10 – old news?

- **JDK 10 was in General Availability on March 20, 2018**
  - **109 new features** and **APIs!**
- **12 JEPs** included
  - 286: Local-Variable Type Inference
  - 296: Consolidate the JDK Forest into a Single Repository
  - 304: Garbage-Collector Interface
  - 307: Parallel Full GC for G1
  - 310: Application Class-Data Sharing
  - 312: Thread-Local Handshakes
  - 313: Remove the Native-Header Generation Tool (javah)
  - 314: Additional Unicode Language-Tag Extensions
  - 316: Heap Allocation on Alternative Memory Devices
  - 317: Experimental Java-Based JIT Compiler
  - 319: Root Certificates
  - 322: Time-Based Release Versioning

Local Variable Type Inference (JEP 286)

- Extending **type inference** to declarations of **local variables** and **initializers**
  - Static type safety with reduced ceremony associated with writing Java
  - Examples:

```
var list = new ArrayList<String>();  // infers ArrayList<String>
var stream = list.stream();          // infers Stream<String>
var m = new HashMap <String, List<BigDecimal>>();
```

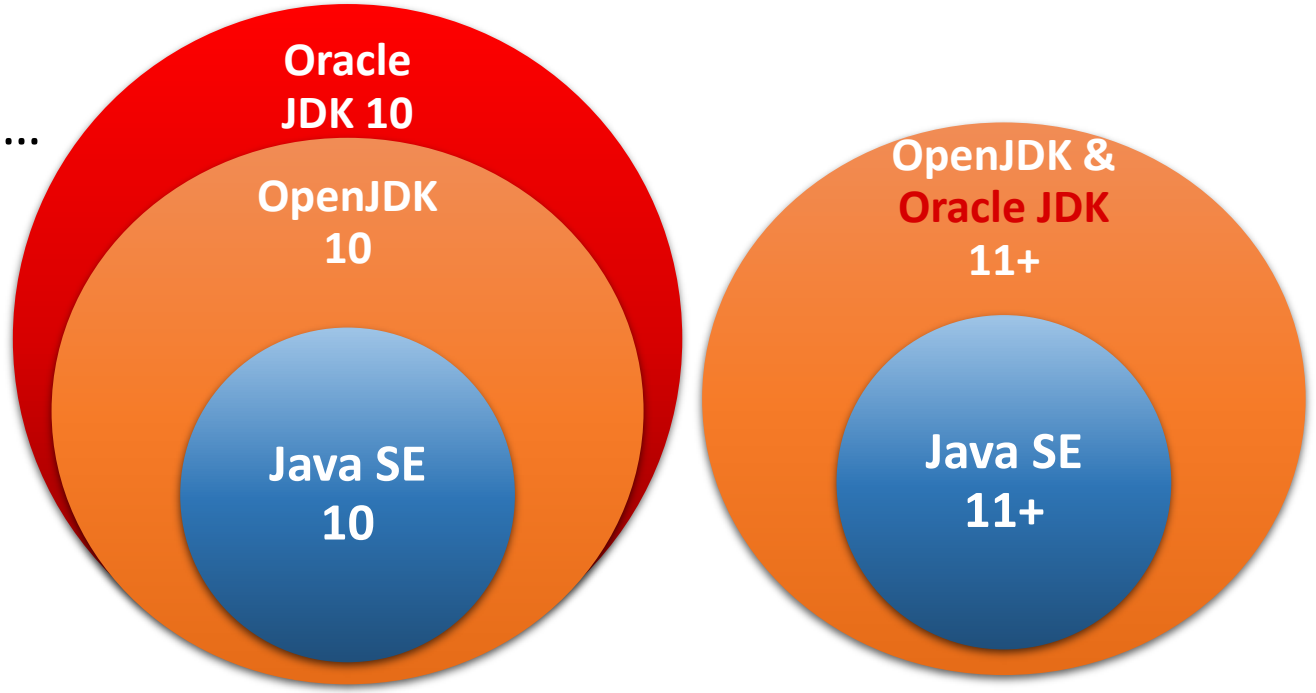
- Restricted to: **local variables** with initializers, **indexes** in the enhanced for-loop, **locals** declared in a traditional for-loop
- Not available for: **method** parameters, **constructor** parameters, **method return types**, **fields, catch formals** or any other kind of variable declaration
- *Don't blame language features for making developers write sh**y code* – Simon Maple

JDK 11 – LTS version (finally)

- **JDK 11 was in General Availability on September 25, 2018**
  - **90 new features** and **APIs!**
- **17 JEPs** included:
  - 181: Nest-Based Access Control
  - 309: Dynamic Class-File Constants
  - 315: Improve Aarch64 Intrinsics
  - 318: Epsilon: A No-Op Garbage Collector
  - 320: Remove the Java EE and CORBA Modules
  - 321: HTTP Client (Standard)
  - 323: Local-Variable Syntax for Lambda Parameters
  - 324: Key Agreement with Curve25519 and Curve448
  - 327: Unicode 10
  - 328: Flight Recorder
  - 329: ChaCha20 and Poly1305 Cryptographic Algorithms
  - 330: Launch Single-File Source-Code Programs
  - 331: Low-Overhead Heap Profiling
  - 332: Transport Layer Security (TLS) 1.3
  - 333: ZGC: A Scalable Low-Latency Garbage Collector (Experimental)
  - 335: Deprecate the Nashorn JavaScript Engine
  - 336: Deprecate the Pack200 Tools and API

Open Sourcing and Converged Binaries

- **No functional difference** between **OpenJDK** and **Oracle JDK** in **JDK 11+**
- **Open sourcing** most of the closed-source parts of JDK
  - Flight recorder, Mission control …
- **Removing** some closed-source parts
  - Browser Plugin, Java Web Start, JavaFX …
- **Backwards Compatibility**
  - Java SE Applications should work
- **Docker container aware** JVM
  - Container CPU count and memory size
- **Open sourcing Java EE**
  - **Jakarta EE** (jakarta.ee)
    as a part of Eclipse Foundation



Garbage Collectors

- Many GCs to choose from:
- **Serial GC**
- **Parallel GC** and **Parallel Old GC**
- **CMS (Concurrent Mark and Sweep) GC** (deprecated in Java 9)
- **G1 (Garbage-First) GC** (default since Java 9)
  - **Parallel Full GC** for G1 (updated JEP 307 in Java 10)
  - **Abortable Mixed Collections** for G1 (JEP 344 in Java 12)
  - **Promptly Return Unused Committed Memory from G1** (JEP 346 in Java 12)
- **Epsilon GC** (no-op GC, experimental in Java 11)
- **ZGC** (experimental in Java 11)
  - Improved in Java 12
- **Shenandoah GC** (experimental in Java 12)
- Others: **Azul's C4 GC …**

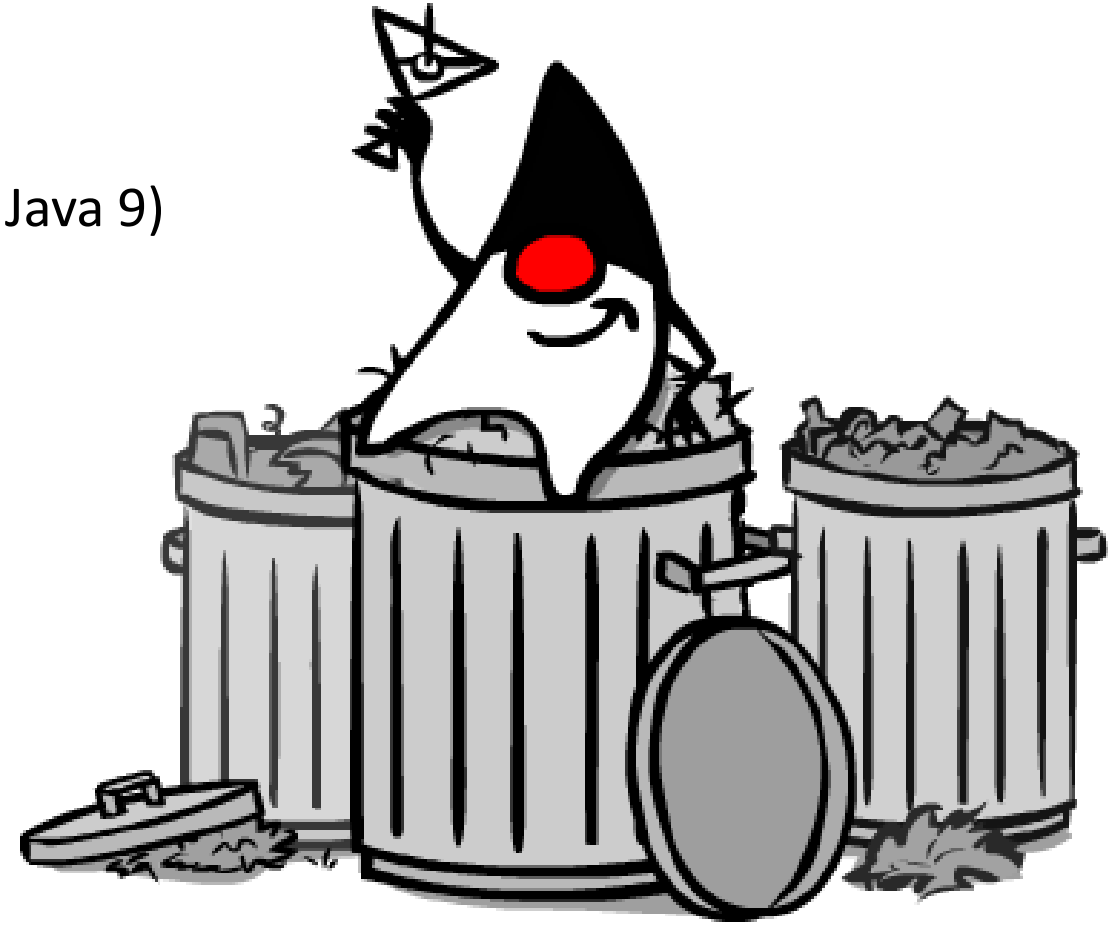


Image Source: 5 Coding Hacks to Reduce GC Overhead, Tall Weiss, OverOps, 2013

JDK 12 – the latest one

- **JDK 12 was in General Availability on March 19, 2019**
  - **Many new features** and **APIs**
- **8 JEPs** included:
  - 189: Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)
  - 230: Microbenchmark Suite
  - 325: Switch Expressions (Preview)
  - 334: JVM Constants API
  - 340: One AArch64 Port, Not Two
  - 341: Default CDS Archives
  - 344: Abortable Mixed Collections for G1
  - 346: Promptly Return Unused Committed Memory from G1

Switch Expressions (*Preview*, JEP 325)

- "Simplified" **switch** form with "**case L ->**" switch labels
- If a label is matched, then only the expression or statement to the right of an arrow label is executed – there is no fall through
- Example:

```
static void howMany(int k) {
    switch (k) {
        case 1 -> System.out.println("one");
        case 2 -> System.out.println("two");
        case 3 -> System.out.println("many");
    }
}
```

Switch Expressions (JEP 325) – example

```
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("Hmm: " + day);
};
```

```
int numLetters = switch (day) {
   case MONDAY, FRIDAY, SUNDAY -> 6;
   case TUESDAY                -> 7;
   case THURSDAY, SATURDAY     -> 8;
   case WEDNESDAY              -> 9;
   // no default!!!
};
```

Microbenchmarking Suite (JEP 230)

- **Basic suite of microbenchmarks** for JDK
  - Initial set of 100+ benchmarks
  - Easy to run existing microbenchmarks and create new ones
  - https://openjdk.java.net/jeps/230
- Based on **Java Microbenchmarking Harness (JMH)**
  - Rich framework for developing performance benchmarks for Java applications
  - Version 1.12 or later (http://openjdk.java.net/projects/code-tools/jmh)
  - Originally developed by Aleksey Shipilëv
- Goals:
  - Stable (non-moving) and tuned benchmarks, targeted for continuous performance testing (new releases supported)
  - Simple – easy to find and run a benchmark, built, add new benchmarks, update tests as APIs
  - Support comparison to previous JDK releases for applicable tests

One AArch64 Port, Not Two (JEP 340)

- Remove all of the sources related to the 64-bit ARM **arm64 port** while retaining the 32-bit ARM port and the 64-bit **aarch64 port**
  - Allows all contributors to focus their efforts on a single 64-bit ARM implementation, and eliminate the duplicate work required to maintain two ports
  - https://openjdk.java.net/jeps/340
- OpenJDK had two 64-bit ARM ports
  - One by Oracle (referred as arm 64) and the other by Red Hat (referred as aarch64)
  - Both ports produce aarch64 implementations
- Unnecessary
  - Oracle stopped supporting ARM port for their JDK binaries
  - Decision was made to use only the Red Hat's port - still maintained and developed

Default CDS Archives (JEP 341)

- Generate **class data-sharing (CDS) archive** on 64-bit platforms using the default class list
  - Class Data Sharing (CDS) used to be a commercial feature in the Oracle JDK, now included in OpenJDK
  - https://openjdk.java.net/jeps/341
- Goals:
  - Improve out-of-the-box startup time
  - Eliminate the need for users to run `-Xshare:dump` to benefit from CDS
- To use CDS, an archive is required that has been generated for classes that are loaded when an application starts
- In JDK 12 (for 64-bit platforms) there is **classes.jsa** file in lib/server directory as the CDS archive for the "default classes"
- Since CDS is turned on by default (equivalent to the `-Xshare:auto` option) users will benefit from improved startup time for applications
  - Measurements on 64-bit platforms show a **32% or more startup time reduction** running HelloWorld

JDK 12 – short conclusion

- JDK 12 provides a **small number** of new features and APIs ☹
- **Switch expressions** is the most interesting to developers
- G1 users will appreciate the **performance improvements**
- And that's about it…

- However, there are some other nice things to consider! ☺

Programming Polyglotism

- **Polyglot programming problems**:
  - Cross-language interoperability
  - General-purpose programming languages not-so-good performance
  - Language tools – configuration, debugging…
- New ideas:
  - Project **Maxine** – OOP compiler
  - OpenJDK's Project **Metropolis** – presented **GraalVM**

GraalVM™

- **GraalVM** – high-performance embeddable polyglot virtual machine that enables you to combine different programming languages that incur almost no overhead
  - In the JVM, into standalone native image, or embedded into large application

GraalVM – components

- **JVM** – HotSpot VM or other
- **JVM Compiler Interface (JVMCI)** – supports custom optimizing compiler
- **Graal Compiler** – new optimized compiler for JVM languages
  - Written in **Java**, uses Graal Intermediate Representation (Graal IR)
  - Optimized inlining and escape analysis algorithms
- **Truffle** framework – any other language
  - Uses Abstract Syntax Trees (ASTs), partial evaluation for interpreters, same interoperability protocols
- **Sulong** (LLVM) – high-performance LLVM (Low-Level Virtual Machine) bitcode interpreter
- Additional: Native images with **Substrate VM**

GraalVM – architecture

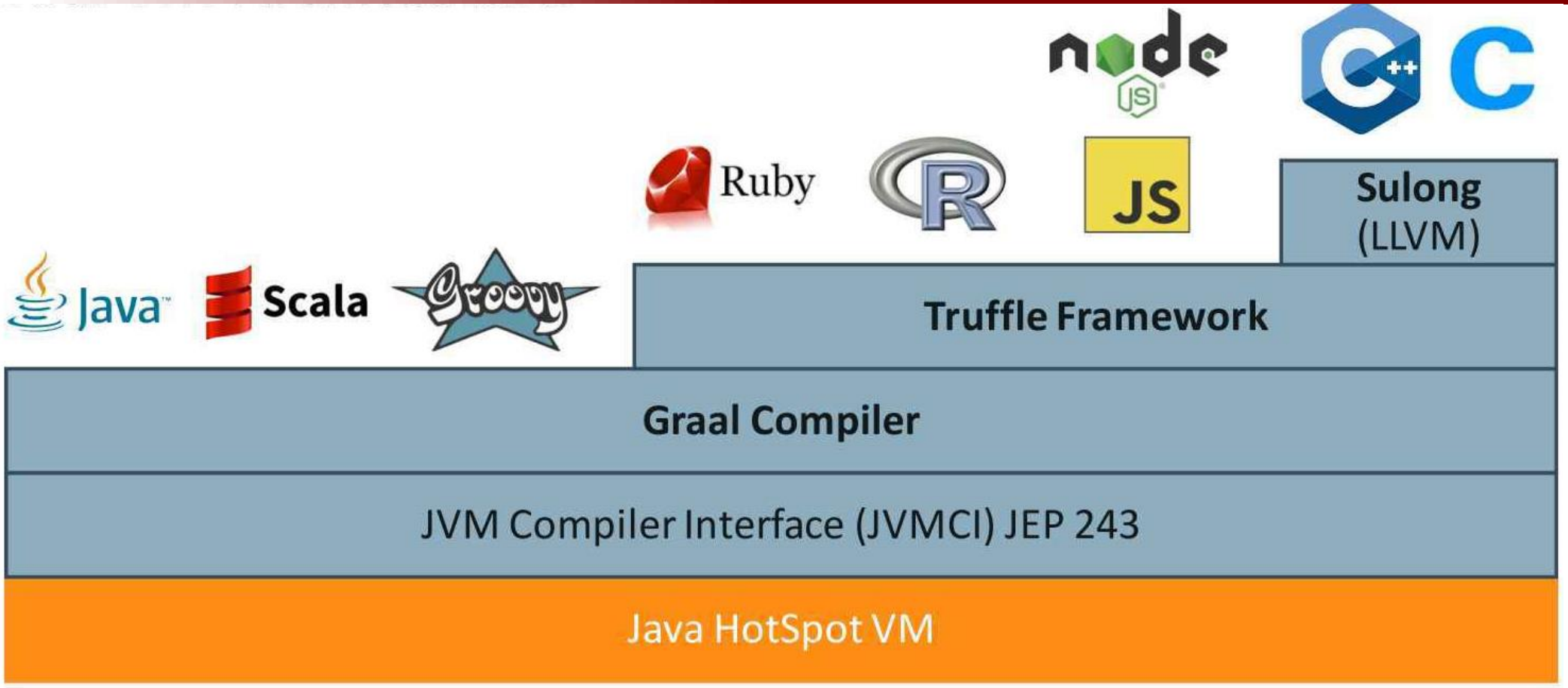


Image Source: Oracle, 2017

Quarkus (quarkus.io)

QUARKUS

- **Supersonic Subatomic Java** – **Kubernetes** Native Java stack tailored for **GraalVM** & OpenJDK **HotSpot** (with common Java libraries and standards)

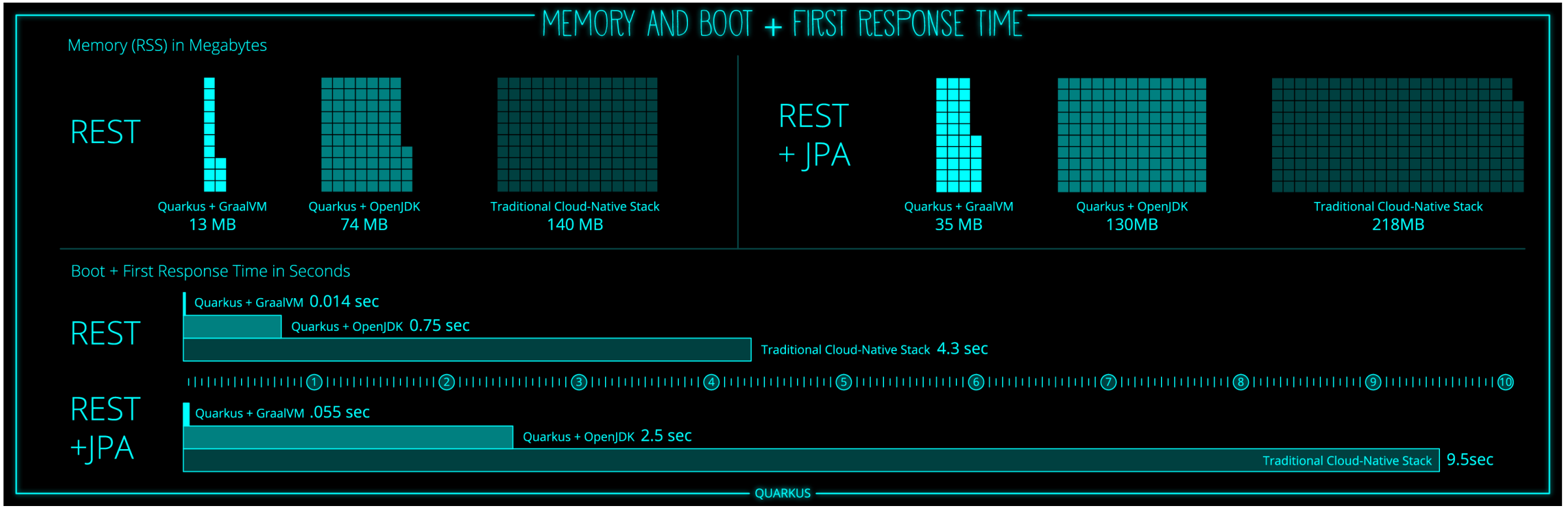
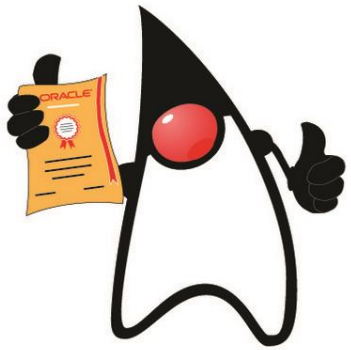


Image Source: Quarkus, 2019

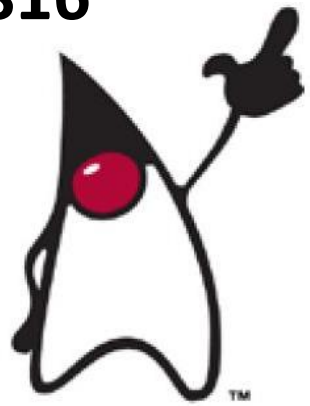
New Java 11 Certifications

- Finally (March 20), **OCP Java SE 11 Developer Certification** ☺
  - New Oracle Certified Professional: Java SE 11 Developer Certification
- **Java SE 11 Programmer I** – Exam **1Z0-815**
  - Questions: 80, Duration: 180 minutes, Passing score: 63%
- **Java SE 11 Programmer II** – Exam **1Z0-816**
  - Questions: 80, Duration: 180 minutes, Passing score: 63%
- Guide: **Java SE 11 Certification Questions Answered**
  - www.oracle.com/a/ocom/docs/dc/ou-5021-java-se11-faq-4.pdf
- To get a certificate you need **both exams** passed!
  - Price: each exams is approx. 1500 HRK



Upgrade to Java 11 Certifications

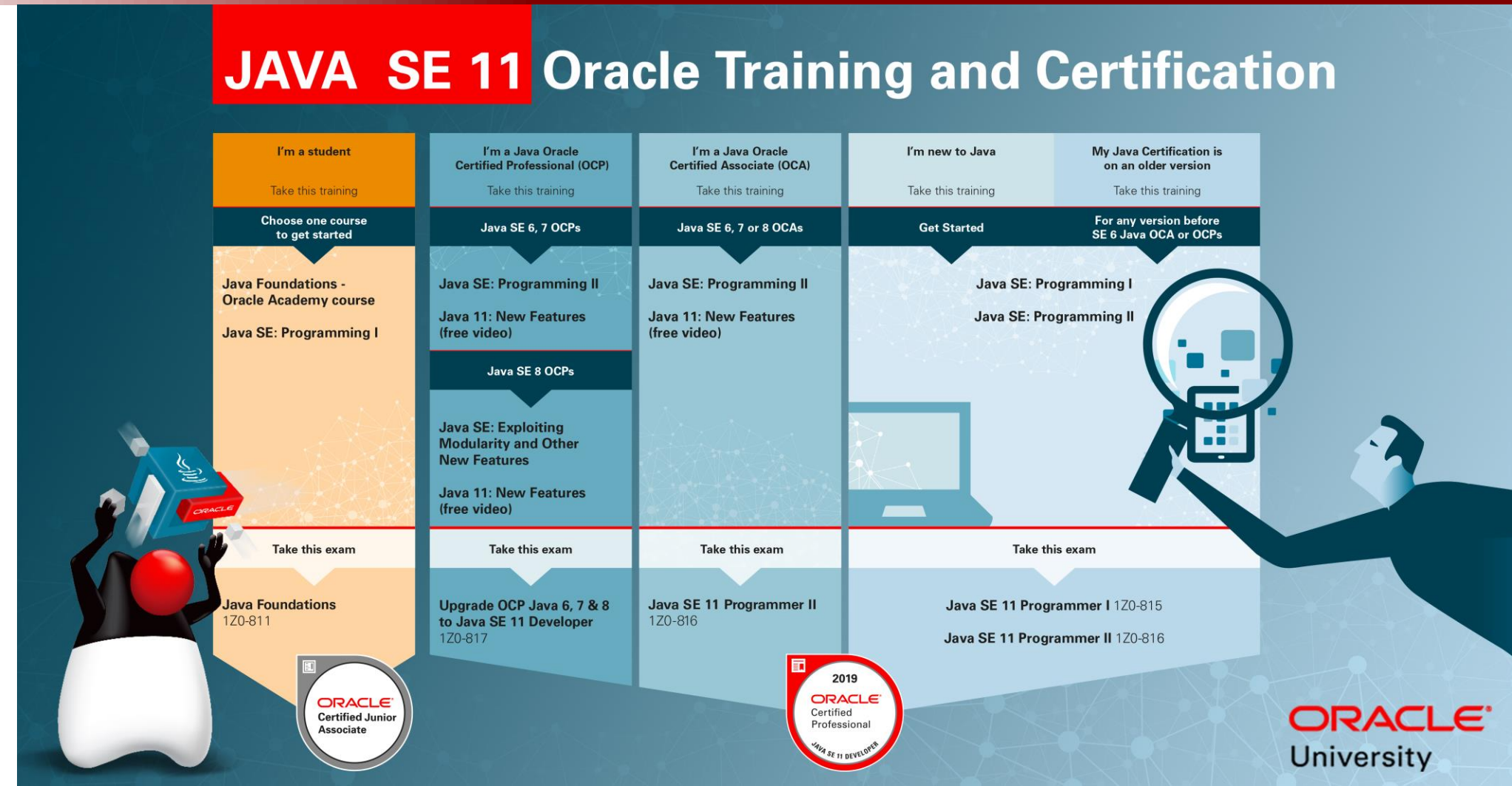
- If you have **OCP/OCA/SCP Java SE 6, 7 or 8**:
- **Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer** – Exam **1Z0-817**
  - Questions: 58, Duration: 120 minutes, Passing score: 61%
- **Upgrade OCA Java 5, 6, 7 & 8 to Java SE 11 Developer** – Exam **1Z0-816**
  - Questions: 80, Duration: 180 minutes, Passing score: 63%
- + Free video: **Java 11 – New Features**



- For **students** only (JDK 8 only):
- **Java Foundations Certified Junior Associate Certification** – Exam **1Z0-811**
  - Questions: 75, Duration: 150 minutes, Passing score: 65%, Price: approx. 600 HRK

Various ways to get Java certificate

- **Student**
- From **OCA**
- From **OCP**
- From the **scratch**



Red Hat stewards Java 8 and 11?

- On April 17, **Red Hat** (with new parent company **IBM**) took the role of managing OpenJDK 8 and OpenJDK 11
  - Under the guidance of Technical Lead **Andrew Haley**

- **Nondisruptive** change
  - Users of Java 8 (on Oracle JDK) may want to locate another distribution
  - Most likely does not impact users of Java 11 who seek updates
    - Oracle clarified what is free (OpenJDK) and what is not free (Oracle JDK)
  - More at: www.redhat.com/en/about/press-releases/leadership-openjdk-8-and-openjdk-11-transitions-red-hat
- N.B. IBM bought Red Hat for $34B in 2018

Is Java still "Free"?

- **$free** as in **free beer** (the cost) vs **free** as in **free speech** (what can you do)
- For **$free** use of **OpenJDK binaries**
- For **free** use of **OpenJDK** with **GPLv2+CE license**
- **Updates** refers to **code patches** – typically **$free**
- **Support** means **fixing bugs** and **answering questions** – was **never $free**
- LTS release **every 3 years** – **does not mean** 3 years of **$free updates**
- **Oracle JDK 11** (and onward) in **production** (only) with **commercial Java SE subscription**
  - Free JDK 11 (and later) are only OpenJDK binaries
- However, **Oracle JDK 8** can be used **indefinitely for free**
  - **Without** any further **security patches** and **bug fixes**

How much $$$?

Is Java really "Moving Forward Faster"?

- **Community opinion**: well… **yeah**! ☺
- **Much more frequent Java releases**
- **Faster access to new features**
- **Many new improvement ideas**
- **A lot of maintenance and housekeeping**
- **Java still remains free**

- **Looking forward to new things!**

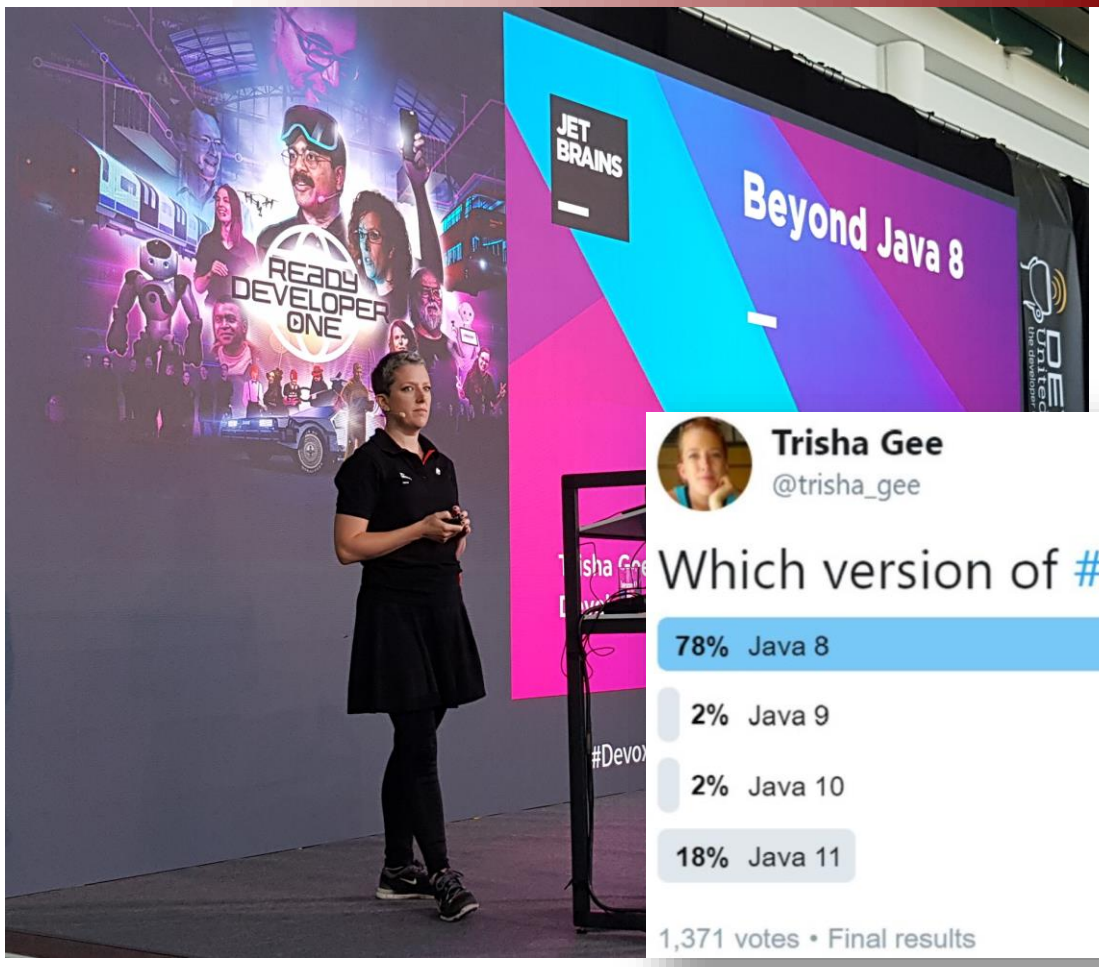
What's still missing?

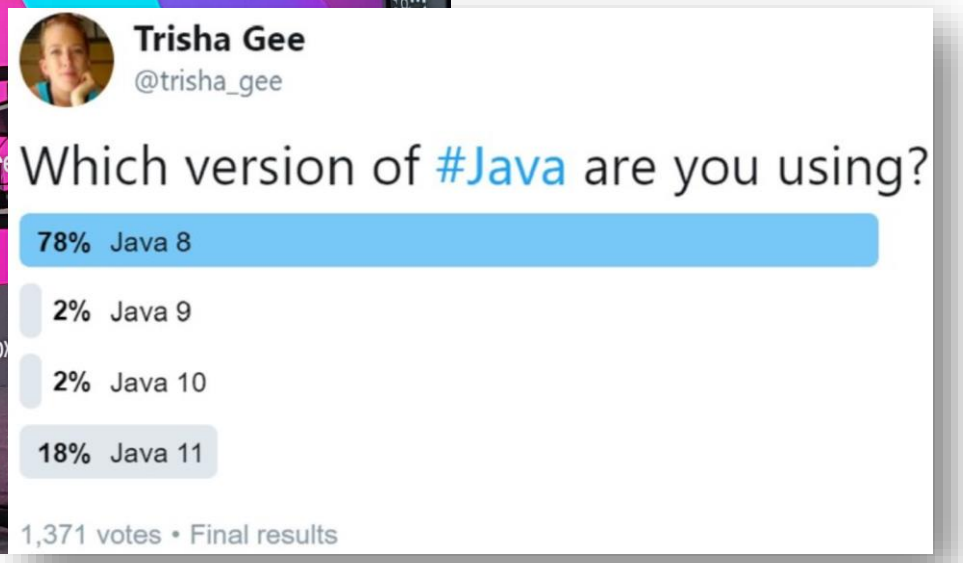
- (Unconfirmed) **JEP candidates** for **JDK 13**
  - JEP 348: **Java Compiler Intrinsics for JDK APIs**
  - JEP 349: **JFR Event Streaming**
  - JEP 350: **Dynamic CDS Archives**
  - JEP 351: **ZGC: Uncommit Unused Memory**
  - JEP 352: **Non-Volatile Mapped Byte Buffers**
  - JEP 353: **Reimplement the Legacy Socket API**
  - JEP 354: **Switch Expressions** – non-preview
  - JEP 326: **Raw String Literals** ?

Latest news from Devoxx UK

- We were at **Devoxx UK** ☺
  - May 9-11, London
- Listening interesting talks
  - **Martijn Werburg, Mark Reinhold, Trisha Gee** and many many others
- Well… from the *core* Java news **not to many new things**
- But a lot of **other technologies and products**

Beyond Java 8 by Trisha Gee





- jshell
- vars
- List.of(), Set.of() …
- toUnmodifiableList()
- Predicate.not()
- Http Client
- Modules
- JEP 302, JEP 305, Data Classes…

More Long-term Java Future

- Project **Amber** – incubator for smaller, productivity-oriented **language features** and **simplifying syntax**
  - Local variable type inference, local variable syntax for lambdas, lambda leftovers, raw string literals, pattern matching, switch expressions…
- Project **Valhalla** – incubator project for **advanced JVM and language feature** candidates
  - Value types and specialized generics
- Project **Panama** – to interconnect JVM and native code
  - Foreign function interface (FFI) as a replacement for JNI
- Project **Loom** – to reduce complexity in writing concurrent applications
  - Fibres (JVM-level threads) and continuations
- Project **Metropolis** – JVM re-written in Java, i.e. "***Java on Java***"
  - Using Graal experience, easier porting, performance to be explored (AOT compiler)
- Project **Skara** – alternative SCM & code review for JDK
  - Git instead of Mercurial

Project Amber

- **Right-sizing language** ceremony
- Includes:
  - **Local variable type inference** (JEP 286) – in JDK 10
  - **Local variable syntax for lambda parameters** (JEP 323) – in JDK 11
  - **Pattern matching for instanceof** (JEP 305) – switch statement with case for different types of objects
  - **Switch Expressions** (JEP 325) – expressions in switch statements and lambdas – (in JDK 12 as preview)
  - **Raw string literals** (JEP 326) – use of single backquote
  - **Java Compiler Intrinsics for JDK APIs** (JEP 348)
  - **Enhanced Enums** (JEP 301) – generic enums with type parameters (on hold)
- More at openjdk.java.net/projects/amber/

Evolution of Java type inference

- From List<String> empty = Collections.<String>eList();
  to (Java **5**) **List<String> e = Collections.eList();**
- From List<String> list = new ArrayList<String>();
  to (Java **7**) **List<String> list = new ArrayList<>();** *diamond operator*
- From Predicate<String> isEmpty = (String s) -> s.length() == 0;
  to (Java **8**) **Predicate<String> isEmpty = s -> s.length() == 0;**
- From BufferedReader reader = new BufferedReader(new FileReader(file);
  to (Java **10**) **var reader = new BufferedReader(new FileReader(file));**

- What about **string literals**?

Source: Java, Today and Tomorrow, Mark Reinhold, Devoxx, 2018

Raw string literals (JEP 326) – examples

```
Runtime.getRuntime().exec
("\"C:\\Program Files\\foo\" bar");
```

```
System.out.println
("this".matches("\\w\\w\\w\\w"));
```

```
String html =
"<html>\n" +
"      <body>\n" +
"            <p>Hello World.</p>\n" +
"      </body>\n" +
"</html>\n";
```

```
Runtime.getRuntime().exec
(`"C:\Program Files\foo" bar`);
```

```
System.out.println
("this".matches(`\w\w\w\w`));
```

```
String html =
"""<html>
    <body>
        <p>Hello World.</p>
    </body>
 </html>
""";
```

→ or """.align();

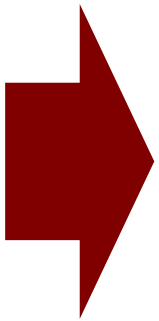
More type inference…

```
String format(Object ob) {
   String s =  "unknown";
   if (ob instanceof Integer) {
      int i = (Integer) ob;
      s = String.format("int %d", i);
   } else if (ob instanceof Double) {
      double d = (Double) ob;
      s = String.format("double %f", d);
   } else if (ob instanceof Point) {
      Point p = (Point) ob;
      s = String.format("point %f %f",
          p.x(), p.y());
   }
   return s;
}
```

```
String format(Object ob) {
   String s =  "unknown";
   if (ob instanceof Integer i) {
      s = String.format("int %d", i);
   } else if (ob instanceof Double d) {
      s = String.format("double %f", d);
   } else if (ob instanceof Point(x, y)) {
      s = String.format("point %f %f",
      x(), y());
   }
   return s;
}
```

Even more type casting

```
String format(Object ob) {
    String s =  "unknown";
    if (ob instanceof Integer i) {
        s = String.format("int %d", i);
    } else if (ob instanceof Double d) {
        s = String.format("double %f", d);
    } else if (ob instanceof Point(x, y)) {
        s = String.format("point %f %f",
        x(), y());
    }
    return s;
}
```



```
String format(Object ob) {
    return switch(ob) {
    case Integer i -> String.format(
        "int %d", i);
    case Double d) -> String.format(
        "double %f", d);
    case Point(x, y)) -> String.format(
        "point %f %f", x, y);
    }
    default -> "unknown";
}
```

Project Valhalla

- Incubator project for **advanced Java VM and language feature** candidates
- Problem:
  - **Primitives** for performance and **objects** for OO, encapsulation, polymorphism, inheritance…
  - But still, there is no **ArrayList<int>** ☹
  - If we use Integer than (un)boxing, creation of object, heap, indirection reference…
- **Value Objects (JEP 169)** – "*codes like a class, works like a primitive*"
  - Supports methods, fields, implements interface, encapsulation, generic…
  - Doesn't support mutation or sub-classes
- **Generics over Primitive Types (JEP 218)** – extends generic types to support the **specialization of generic classes** and interfaces over primitive types
- More at openjdk.java.net/projects/valhalla/

Project Panama

- Interconnecting JVM and **native code**
  - Featuring native function calling from the JVM and native data access from the JVM
- **Foreign function interface (FFI)** as a replacement for JNI, includes:
  - Native function calling from JVM (C, C++), specifically per JEP 191
  - Native data access from JVM or inside JVM heap
  - New data layouts in JVM heap, native metadata definition for JVM
  - Header file API extraction tools
  - Native library management APIs and native-oriented JIT optimizations
  - Native-oriented interpreter and runtime “hooks”, class and method resolution “hooks”

Project Loom

- **Threads** cannot match the scale of the domain's unit of concurrency
  - Millions of transactions, users or sessions – number of OS threads is much less
- Most concurrent applications need some synchronization between threads for every request
  - An expensive context switch happens between OS threads
- Project **Loom** – reducing complexity in writing concurrent applications
  - Alternative, **user-mode thread implementations**, **delimited continuations**, and other constructs involving **call-stack manipulation**
  - Proposal for lightweight **fibres** (JVM-level threads) as alternative implementation of threads, managed by schedulers like `ForkJoinPool`, written in Java
- Ordinary Java threads preserved, performance improved, and footprint reduced
  - Less memory and almost zero overhead when task switching

Fibres – preliminary results in JDK 13

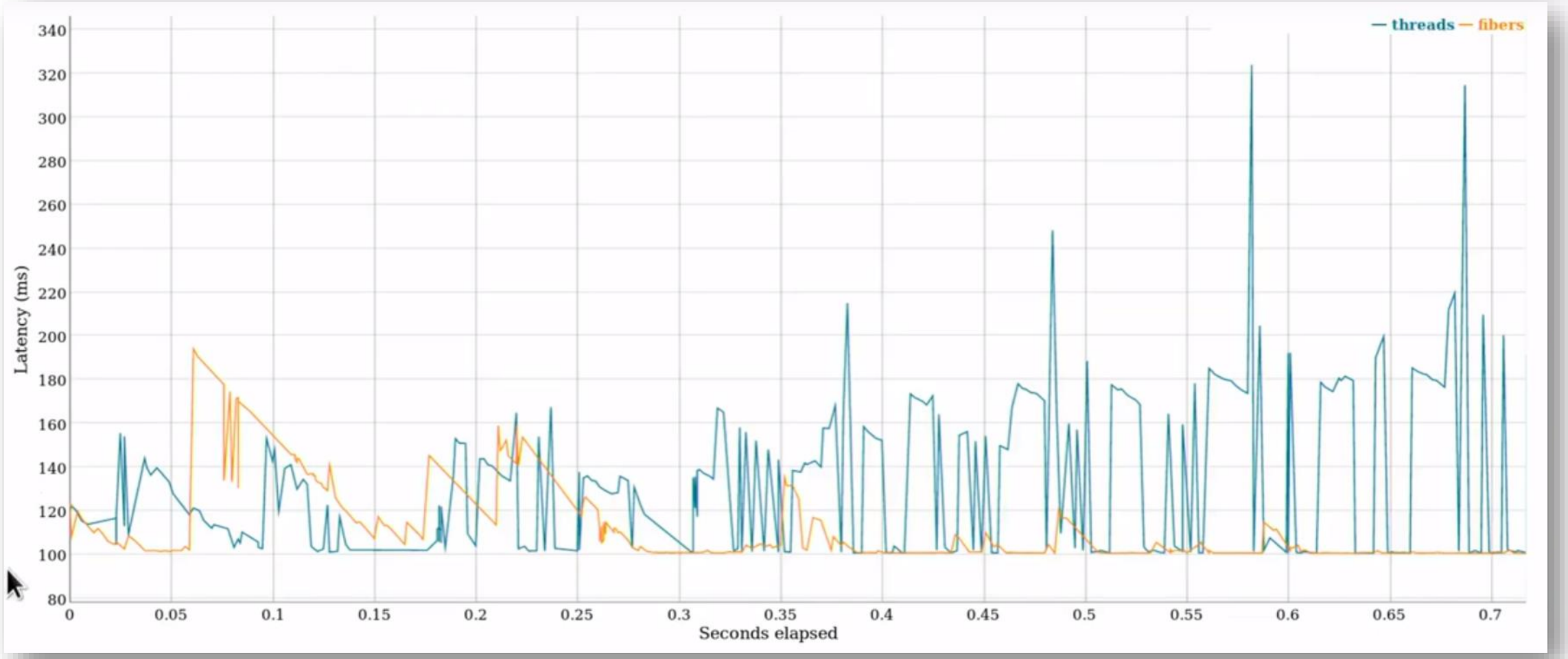


Image Source: Mark Reinhold, Oracle, 2019

Data Classes and Sealed Types for Java

- Have you heard about **Data Classes** (**Records**)?

```
class Point {

   final double x;
   final double y;

   Point (double x, double y) {
      this.x = x;
      this.y = y;
   }

   double x() { return x; }

   double y() { return y; }
```

```
   double equals (Object o) {
      if (not (o instance of Point)
         …
      return …
   }

   double hashCode () {
      return …
   }

   double toString() {
      return …
   }
```

Source: Mark Reinhold, Devoxx UK, 2019

Data Classes and Sealed Types for Java

- Have you heard about **Data Classes** (**Records**)?

```
record Point(double x, double y) {

   final double x;
   final double y;

   Point (double x, double y) {
      this.x = x;
      this.y = y;
   }

   double x() { return x; }

   double y() { return y; }

   double equals (Object o) {
      if (not (o instance of Point)
         return …
   }

   double hashCode () {
      return …
   }

   double toString() {
      return …
   }
```

You can still override those methods ☺

Sometimes data is just … data.

Mark Reinhold

Source: Mark Reinhold, Devoxx UK, 2019

More at: https://cr.openjdk.java.net/~briangoetz/amber/datum.html

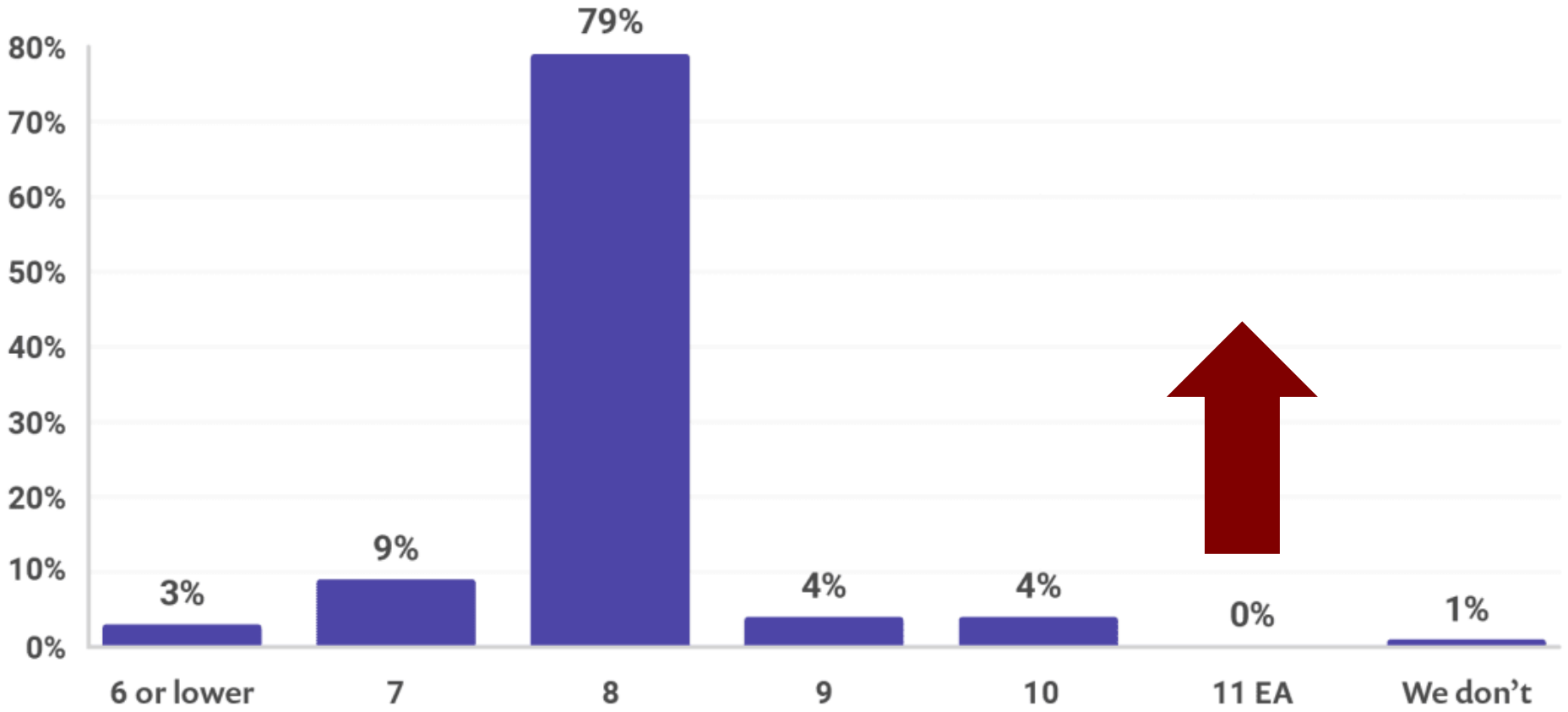
OK, but what do we use in reality?

- **JVM Ecosystem Report 2018** by **Snyk** and **Java Magazine**, October 2018
  snyk.io/blog/jvm-ecosystem-report-2018

Which Java SE version in production?

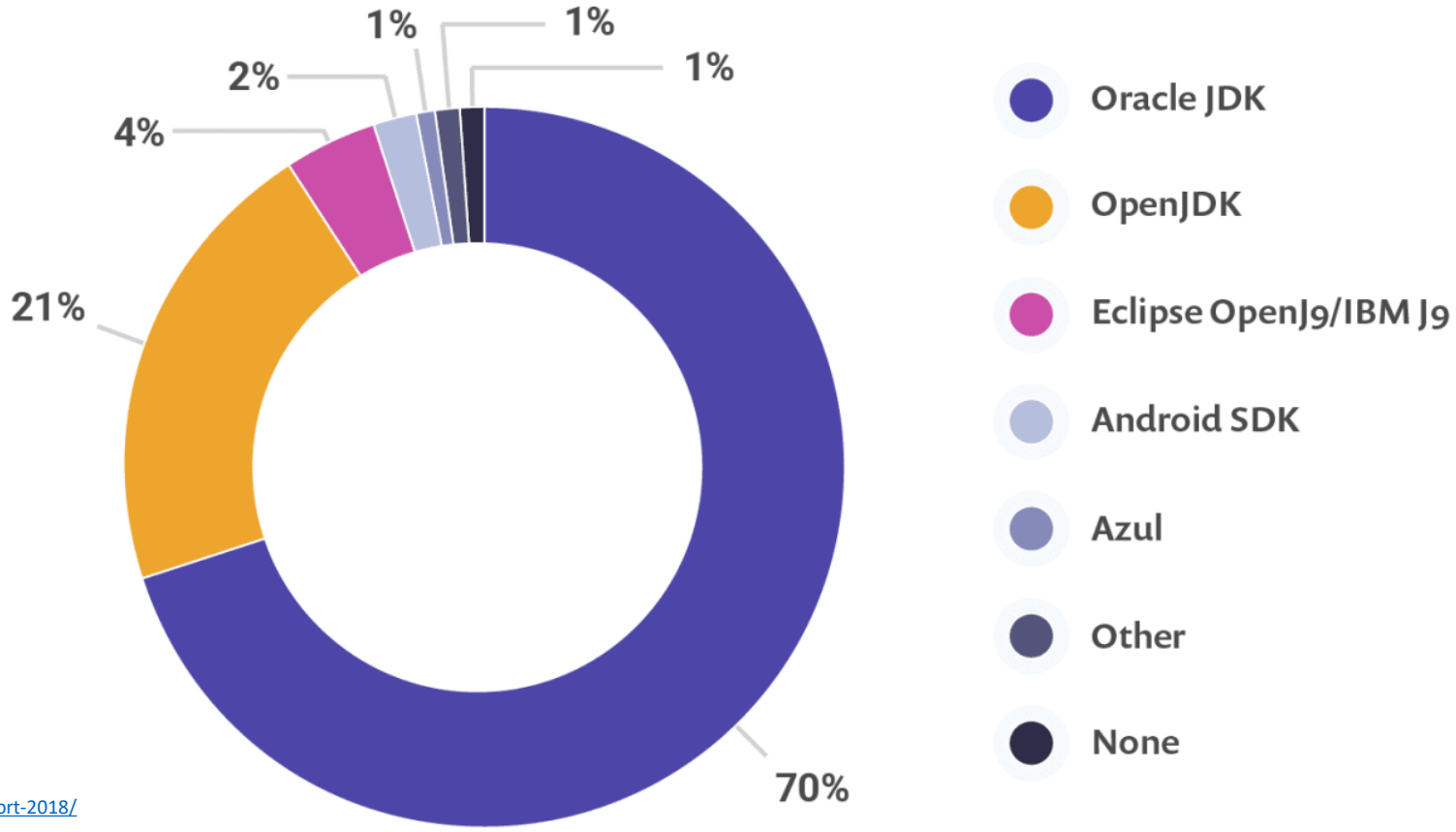
- Which Java SE **version** do you use in production for your main application?

- Note:
  This was **before** Java 11



Which JDKs are in production?

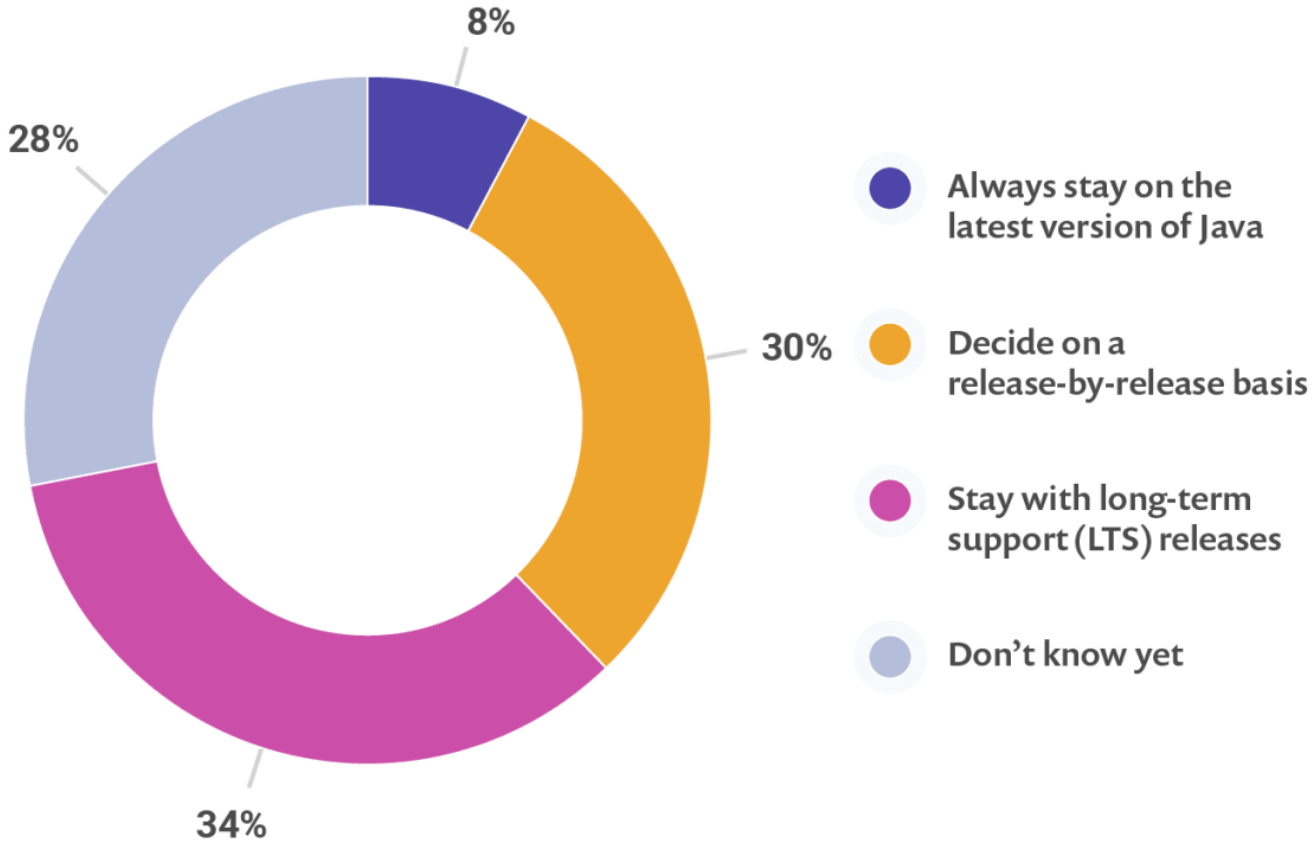
- Which Java vendor's **JDK** do you use in production for main applications?



Source: JVM Ecosystem Report, Snyk, 2018, https://snyk.io/blog/jvm-ecosystem-report-2018/

Which Java SE version in the future?

- How do you plan to respond to Java's new **release cycle**?



Where can you learn Java?

- On **every major university** in the world

MIT, STANFORD, Caltech, HARVARD, ETH zürich

- On **all major online learning** and MOOC platforms

edX & RITX, lynda.com, ORACLE UNIVERSITY, Udemy, treehouse, coursera, PLURALSIGHT, UDACITY

What about learning Java in Croatia?

- You can **learn Java** practically in **each and every computing / computer science / information technology university and/or college study program** in Croatia
  - Java is **#1 introductory programming language** for decades! ☺
- In **18+ cities**: Bjelovar, Čakovec, Dubrovnik, Krapina, Križevci, Osijek, Pula, Rijeka, Sisak, Split, Šibenik, Varaždin, Velika Gorica, Virovitica, Zabok, Zadar, Zagreb, and Zaprešić
- On **37 or more educational institutions** including 7 public universities and 15 private colleges
- In **80 or more educational programs** (BS, MS, spec, PhD)

Source: Gdje studirati računarstvo i informatiku u Hrvatskoj?, www.bug.hr/obrazovanje/gdje-studirati-informatiku-u-hrvatskoj-2018-4185, Bug, 2018.

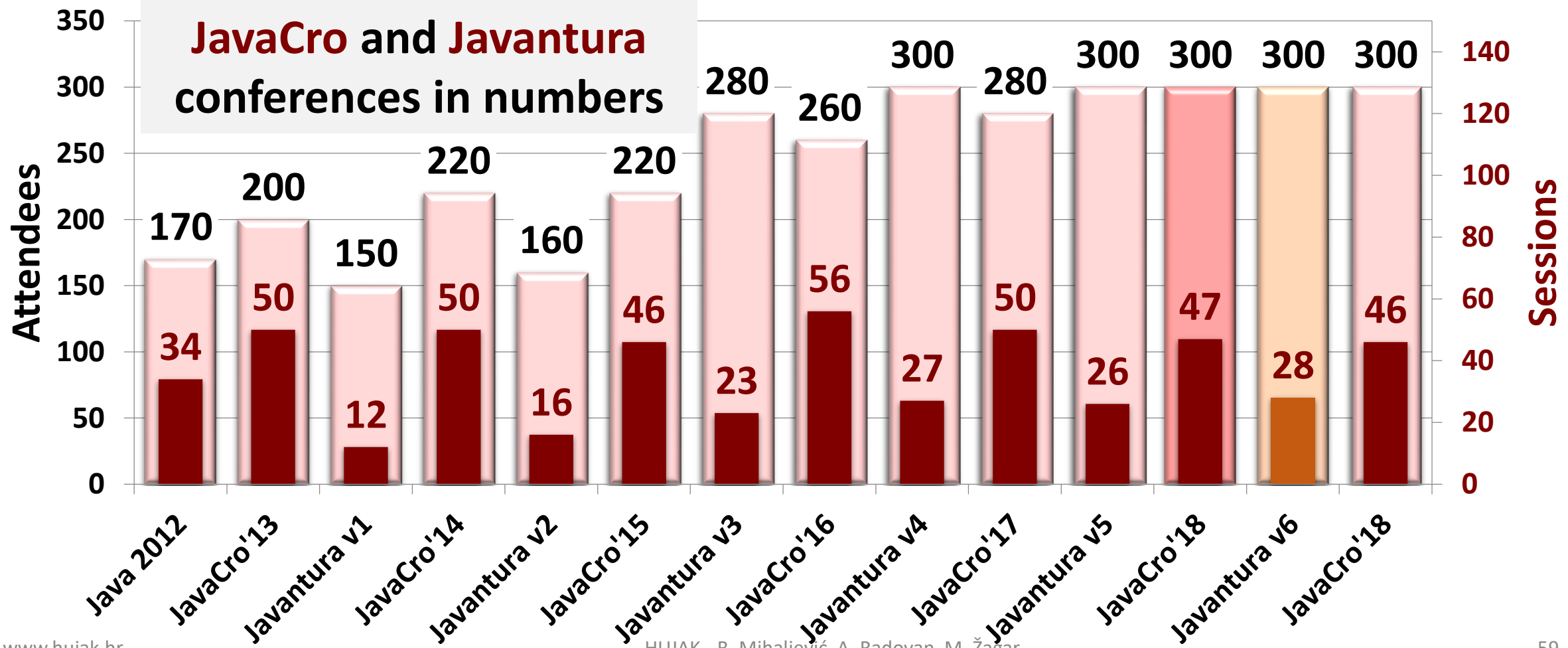
Java-related Conferences in Croatia

Conference	Location	Date	Sessions	Tracks	Attendees	Countries
JavaCro'19	Umag	12.-14.5.2019.	46	5	300	15
Javantura v6	Zagreb	23.2.2019.	28	3	300	-
HrOUG 2018	Rovinj	16.-19.10.2018.	70	7	370	-
JavaCro'18	Rovinj	7.-9.5.2018.	47	5	300	15
						-
						15
						-
						11
						15
						-
						11
						-
JavaCro'14	Poreč	11.-13.5.2014.	50	5	220	11
Javantura v1	Zagreb	22.2.2014.	12	-	150	-
WebCamp 2013	Zagreb	26.10.2013.	24	-	600	-
HrOUG 2013	Rovinj	15.-19.10.2013.	11 (od 90)	1 (od 7)	370	12
JavaCro'13	Tuhelj	3.-5.6.2013.	50	5	200	-
HrOUG 2012	Rovinj	16.-20.10.2012.	11 (od 114)	1 (od 7)	370	13
WebCamp 2012	Zagreb	24.11.2012.	24	-		-
Java 2012	Tuhelj	29.-30.5.2012.	34	7	170	-
HrOUG 2011	Rovinj	18.-22.10.2011.	12 (od 96)	1 (od 9)	460	11

This is our 21st conference!!! ☺

#Javantura #JavaCro #HrOUG #proud

20 conferences in 8 years and we are still there ☺



Conferences HUJAK supports

droidcon

hroug
godišnja konferencija

DORS/CLUC

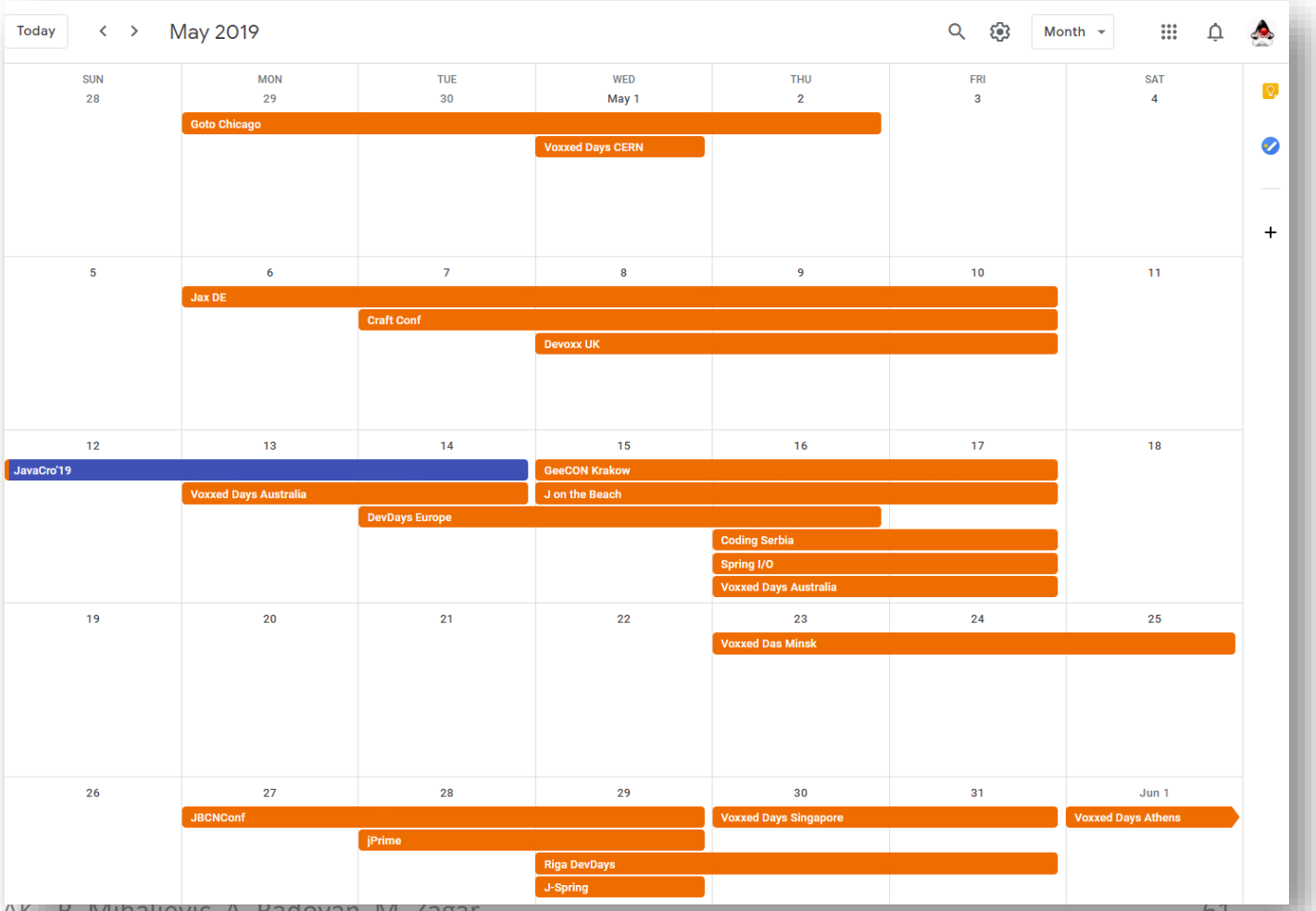
Shift
DEVELOPER CONFERENCE

change
powered by KING ICT

INFOBIP DEV DAYS

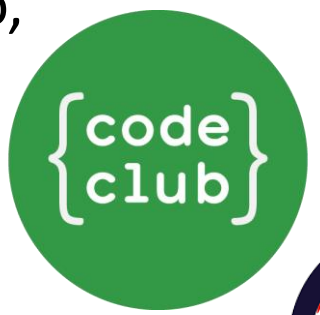
Calendar of Java-related Conferences in EU

- **HUJAK's conference calendar** is available at: hujak.hr/kalendar/
  - Take a look at **May 2019**
  - P.S. If we are missing some event please send email to info (at) hujak.hr



A few nice things happened in 2018/2019

- **Java Zagreb meetups** – many great meetups so far
- **Java in high schools initiative** with **Oracle Academy**
- **Croatian Makers league** (IRIM) continues
  - Micro:bit, Logo, mBot, Scratch, Arduino, Little Bits…
- **Digital Academy** (Algebra)
  - ScratchJr, RunMarco, Studio Code, Play Lab, Scratch, App studio, micro:bit, Arduino…
- **Code Club Croatia** (**Programerko** & **STEMI**)
- Udruga za darovitu djecu "**Dar**" and many many others
- Great **Javantura and JavaCro conferences** ☺



How can you (and your kids) start?

- **Scratch** (7-16 g.) i **ScratchJr** (5-7 g.)
  - scratch.mit.edu, MIT Media Lab
- **Alice** (11-18 g.)
  - www.alice.org, Carnegie Mellon University
- **Greenfoot** (13-20+ g.)
  - www.greenfoot.org, University of Kent
- **BlueJ** (15-20+ g.) and **jGRASP**
- **Eclipse**, **IntelliJ IDEA**, **NetBeans** …
- Other: **robotics, Minecraft, Raspberry Pi…**








However, the most interesting are… robots!

Photo from Javantura v3 conference, author Johan Janssen

HUJAK members ☺

Partners & Friends

hroug
hrvatska udruga oracle korisnika

HROPEN

ORACLE ACADEMY

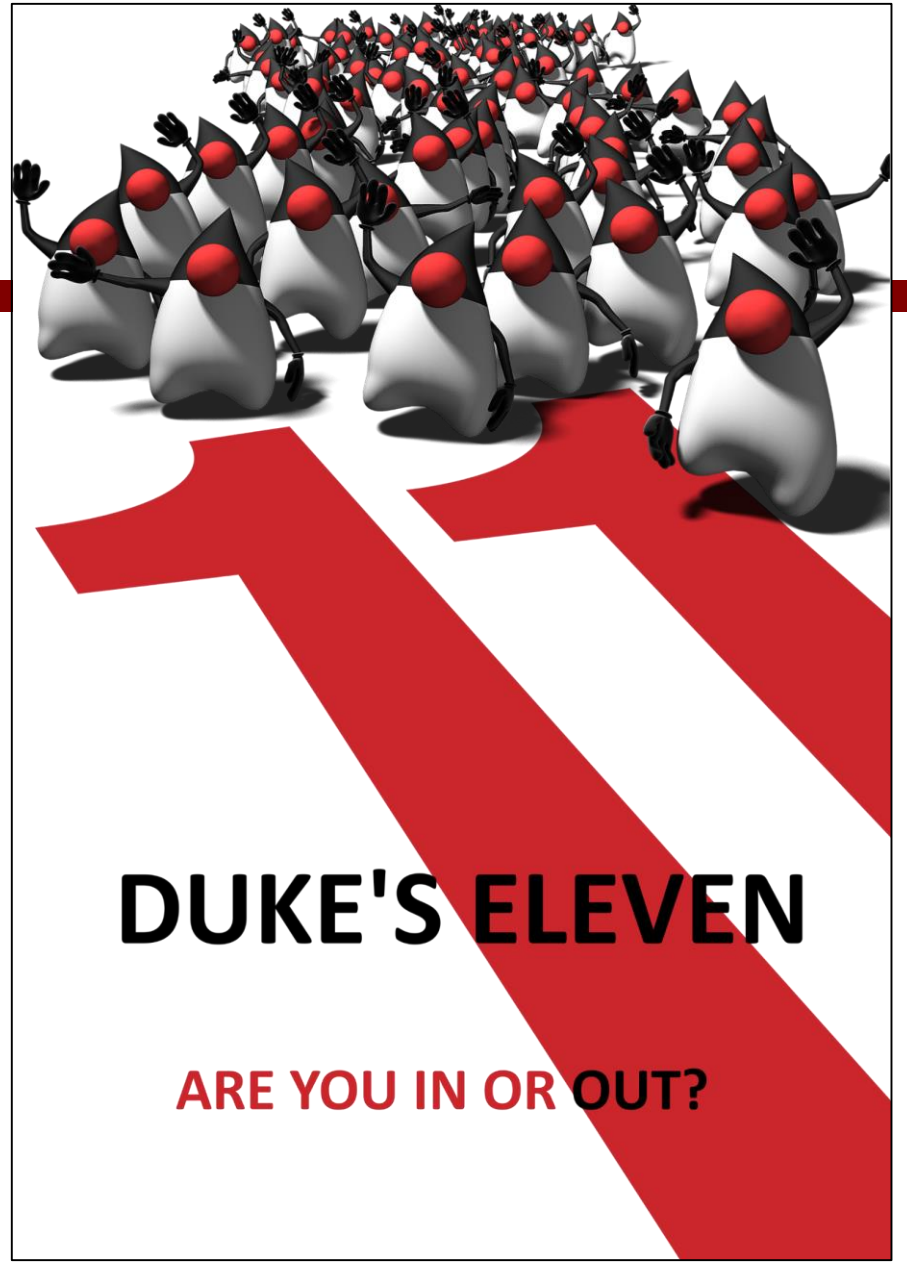
SIOUG

What is our advice?

- Obviously – **use Java 11 LTS**

  or use **Java 12+**

- **OpenJDK** (any) or **Oracle JDK** or any other – it's up to you ☺
- Try to **abandon older** versions (7 and 8)
- Check what is **@Deprecated**
- Migrate every **3 months or 3 years** with LTS
- Get involved more with **HUJAK**!



What is HUJAK?

HUJAK is...

YOU!!!

Thank you & greetings from HUJAK!

- Web page **hujak.hr**
  - www.hujak.hr
- LinkedIn group **HUJAK**
  - www.linkedin.com/groups?gid=4320174
- Facebook group page **HUJAK.hr**
  - www.facebook.com/HUJAK.hr
- Twitter profile **@HUJAK_hr**
  - twitter.com/HUJAK_hr