

Java

is Here to Stay

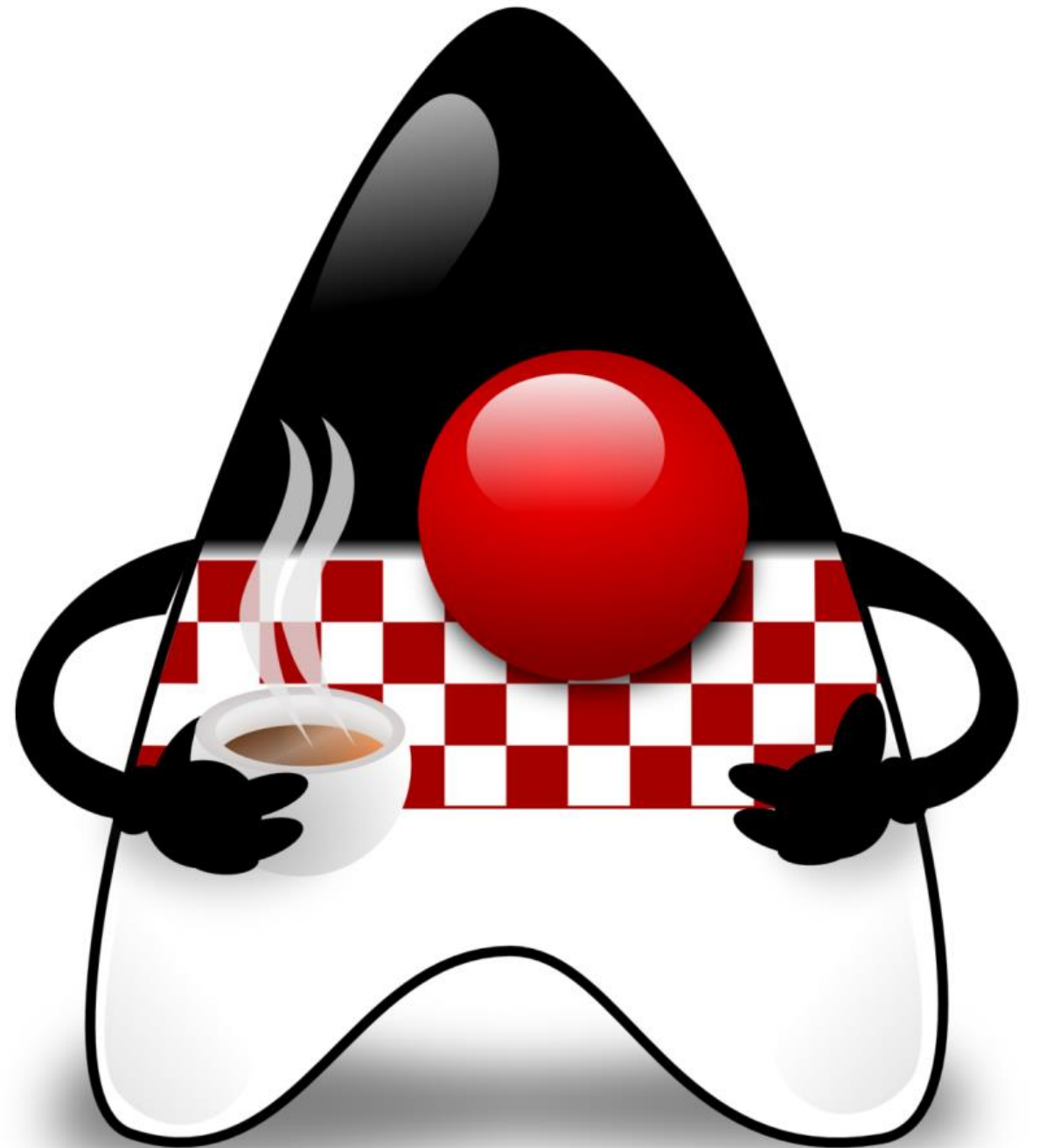
dr. sc. Branko Mihaljević

dr. sc. Aleksander Radovan

Stjepan Matijašević

izv. prof. dr. sc. Martin Žagar

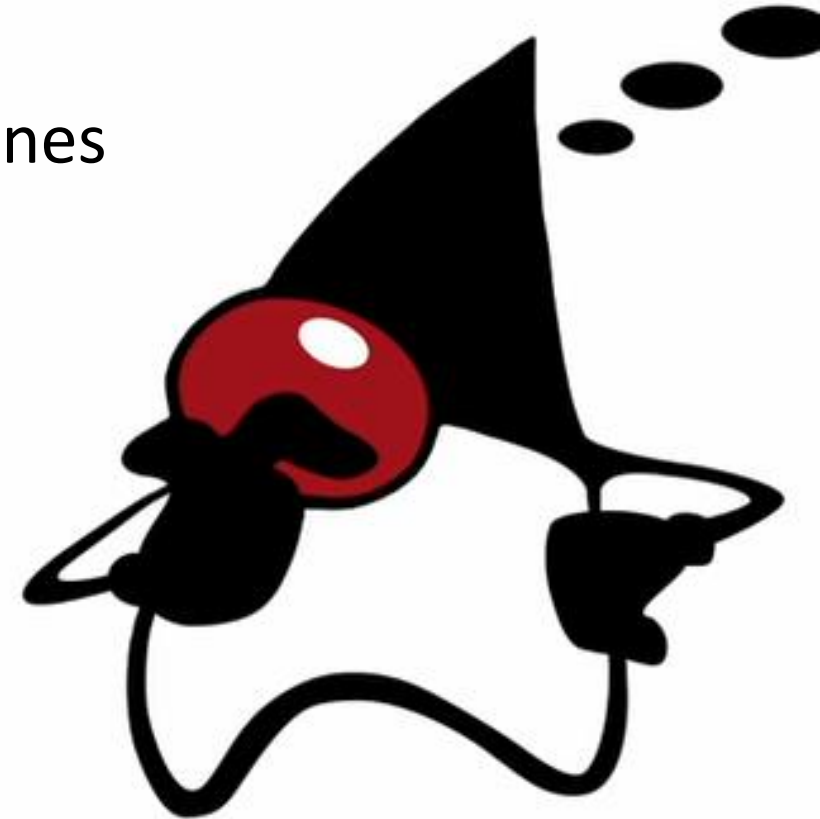
HUJAK





Assessing the **New Development** Landscape

- New programming **languages**, programming **polyglotism & interoperability**
- New software development **paradigms**
- New **frameworks** or new (different) versions of old ones
- Changing hardware and software **architectures**
- Modern application **solutions**
- Variety of **deployment models**
- **Cloud**-everywhere
- **Microservices**
- **Anything/everything-as-a-service**





Why we (still) use **Java** and JVM?



- **Openness and Platform Independence**
- **Community Acceptance and Familiarity**
- **Contribution** of the entire community
- **Variety of tools, libraries and frameworks**
- **Reliability and Trust**
- **Continuous Innovation and Predictability**

Need more proof?

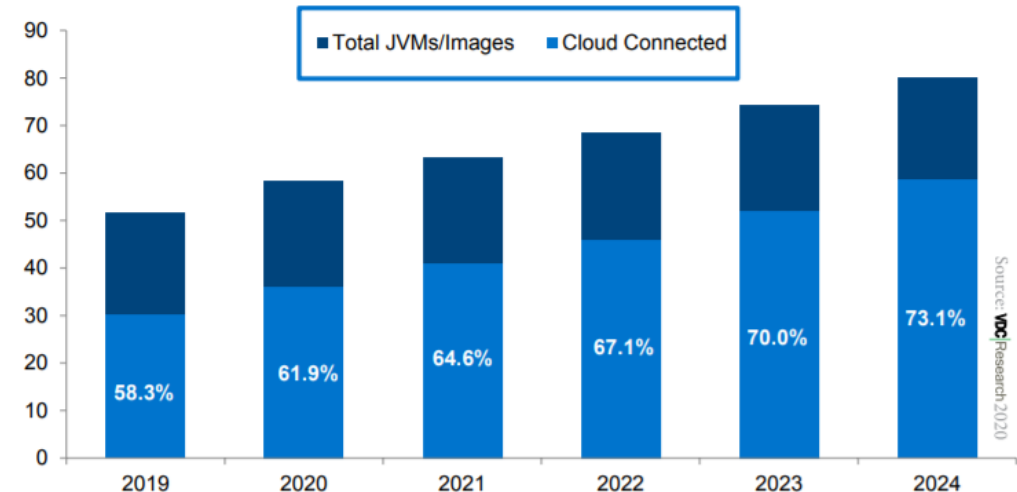


Some Java Facts



- **#1 Development Platform**
 - Continued **growth** for **25+** years
- **#1 Programming Language**
 - Big Data, Microservices, CI Dev Tools, DevOps, Data Management, Mobile ...
- **A Few Dozen Billion Devices** run Java
- **56 Billion Active JVMs**
 - **64%** are **Cloud-based** JVMs
 - Expected to grow at over **9% per year** over the next 5 years

#1 Analytics	#3 Artificial intelligence	#2 Augmented reality/virtual reality	#1 Big data	#2 Blockchain/distributed hyperledger	#1 Chatbots	#1 Continuous integration dev tools
#1 Data management	#1 DevOps	#2 Internet of Things	#1 Microservices	#1 Mobile	#2 Security	#1 Social



Source: Addressing Next-Generation Development with Java, Chris Rommel, VDC <https://www.oracle.com/a/ocom/docs/2020-oracle-wp-next-generation-development-vdc.pdf>



...

*Java's ability to boost **performance, stability, and security** continues to make it the **world's most popular programming language**.*

*According to an IDC report over **10 million developers**, representing **75% of full-time developers** worldwide, **use Java**, more than any other language.*



Sharat Chander, Oracle, September 2021



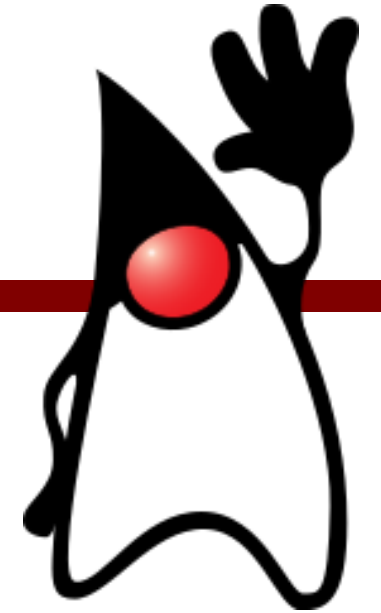
More Java Facts

- **10 Million Java Developers**
- **980k+ Java Certificates**
- **98% Fortune 100** companies hiring Java Developers
- **69%** of Software Developers run Java (some kind of...)
- **50+ JVM languages**
 - JVM languages include: **Groovy, Kotlin, Scala, Clojure, JRuby, Jython, Fantom, Ceylon, Xtend, X10, LuaJ, Golo, Frege, Mirah, Eta, JavaScript...**
- And even more with **GraalVM (+ Truffle + Sulong)**





Current State of Java?



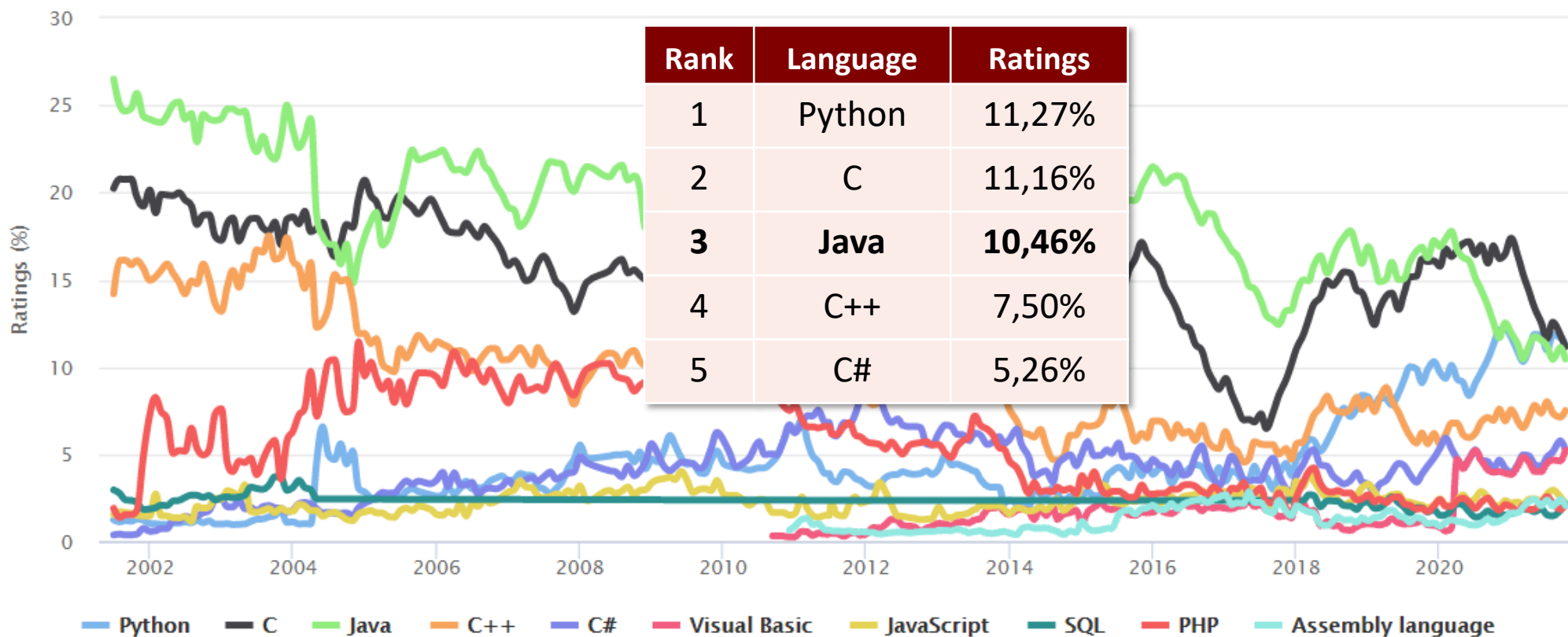
Some questions for all of us:

- Still using **Java 8** (from 2014)?
- Switched to the **LTS** version **Java 11** (from 2018)?
 - Upgraded to 6-month release of **Java 12-16**?
- Thinking about the **latest LTS** version of **Java 17**?
 - What about ~~Java~~ **Jakarta EE** 9 or 9.1?



Is Java **still popular?**

- **TIOBE index for October 2021**

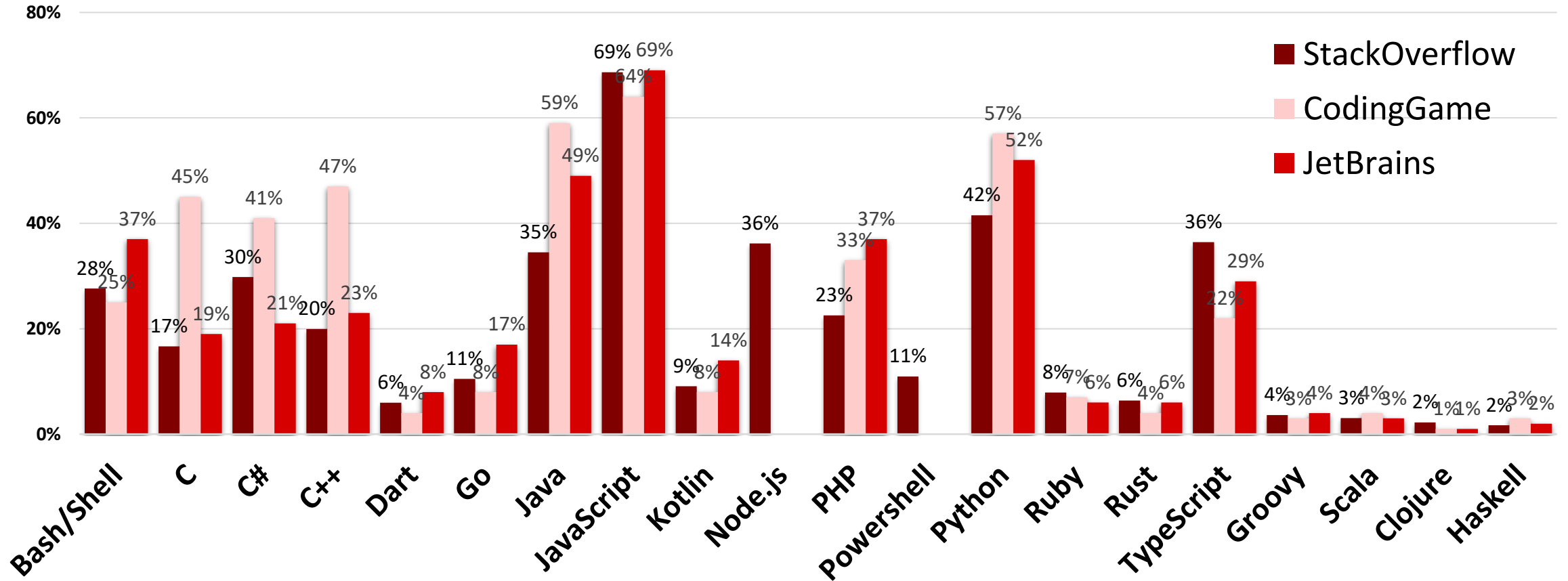


Source: TIOBE Index for October 2021, www.tiobe.com/tiobe-index/



Programming Languages

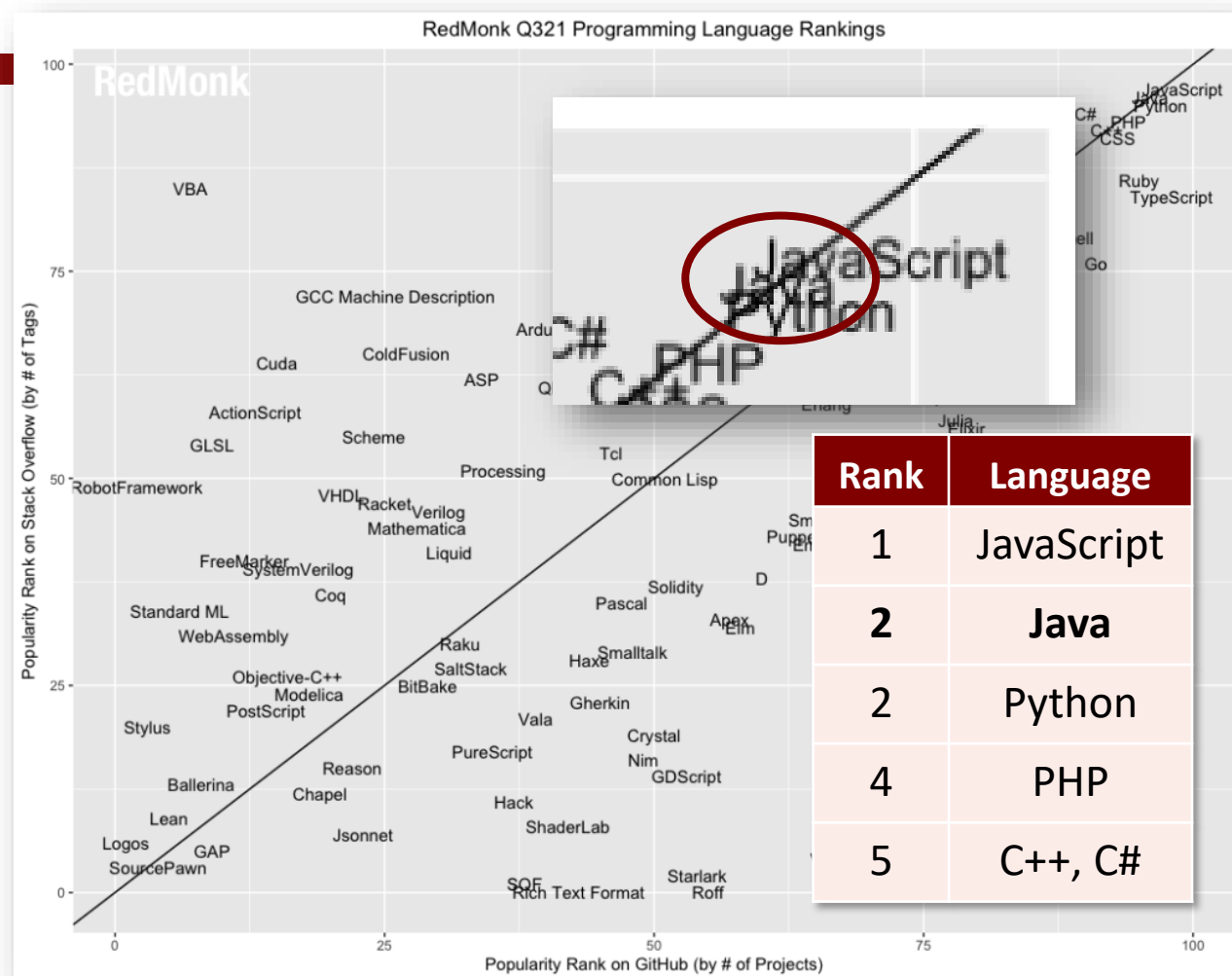
- Which programming languages do you use?





Is Java still popular? #2

- **RedMonk Programming Language Rankings: June 2021**
 - Extraction of language rankings from **GitHub** and **Stack Overflow**
 - Combining them to reflect both code (GitHub) and discussion (Stack Overflow) traction



Source: The RedMonk Programming Language Rankings: June 2021, redmonk.com/sogrady/2021/08/05/language-rankings-6-21/



...

*Java surged back into **a tie for second place** with Python. Java's performance on these rankings continues to **impress**, all these years later, and as it's shown a **remarkable ability to adapt** to a rapidly changing landscape it's a language that would be **difficult to bet against**.*

Stephen O'Grady, RedMonk, August 2021





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 301
from July 20, 2021
- OK, but what about the real **latest JDK download(s)**?

The screenshot shows the Java website's download page. At the top, there is a red header with the Java logo and navigation links for 'Download', 'Help', and 'Developers'. A search bar is located in the top right corner. Below the header, the page title is 'Java Downloads for All Operating Systems' with a recommended version of '8 Update 301' released on July 20, 2021. A prominent yellow warning box contains an 'Important Oracle Java License Update' notice, stating that the Oracle Java License has changed for releases starting April 16, 2019. The notice explains that the new Oracle Technology Network License Agreement for Oracle Java SE is substantially different from prior licenses and that certain uses may no longer be available. It also mentions that commercial licenses are available through a Java SE Subscription and that OpenJDK is available under the GPL license at jdk.java.net. Below the warning box, there is a section for selecting the download file based on the operating system, with links for 'All Java Downloads', 'Remove Older Versions', and 'What is Java?'. At the bottom, a disclaimer states that by downloading Java, the user acknowledges and accepts the terms of the Oracle Technology Network License Agreement for Oracle Java SE.



Available JDKs?

- Oracle **JDK** or one of many **OpenJDK**s
 - Oracle **OpenJDK**
 - AdoptOpenJDK's **OpenJDK**
 - Azul **Zulu OpenJDK**
 - Amazon's **Corretto OpenJDK**
 - **Linux** distribution's **OpenJDKs**
 - RedHat's **OpenJDK**
 - IBM **Java SDK**
 - Azul **Zing**
 - Alibaba **Dragonwell**
 - Bellsoft **Liberica OpenJDK**
 - Eclipse **Adoptium OpenJDK**
 - SAP **SapMachine**
 - Microsoft **OpenJDK** 😊
 - ...
- + Oracle **GraalVM CE or EE**



Available JDKs and Versions

- **Oracle JDK** www.oracle.com/java/technologies/downloads/
 - Java SE **17** (LTS), Java SE **16.0.2**, Java SE **11.0.12** (LTS) or Java SE **8u301** (LTS)
 - NFTC (17) or OTN (before 17) license – to be discussed later
- **Oracle OpenJDK** jdk.java.net
 - Java SE **17** (LTS), Java SE **16.0.2** or any other between **15** and **7**
 - GPLv2+CE license
- ~~AdoptOpenJDK~~ **Adoptium OpenJDK** adoptium.net
 - Java SE **17+35** (LTS), Java SE **16.0.2**, Java SE **11.0.12** (LTS) or Java SE **8u302** (LTS)
 - Apache License, Version 2.0 and GPLv2+CE
- **Azul's Zulu OpenJDK** www.azul.com/downloads/?package=jdk
 - From OpenJDK **17+35** back to Open JDK **6**



Is Java **Moving Forward?**

• **Predictability**

- Evolving Java incrementally and predictably – **stable** evolution
- Progress not alienating extremely large user base – **careful** backward compatibility

• **Trust**

- Open and transparent development model – **no** surprises
- Java community suggests and adopts new features – **community** involvement

• **Innovation**

- Respecting contemporary software development – **innovative** improvements
- Gradually introducing language and platform enhancements – **cautious** innovation



Predictability via JCP



Java
Community
Process

- **Incremental delivery** of new enhancements
 - 6-month releases work like clockwork 😊
- **Continual development** transparency
 - Java Community Process (**JCP**), based on Java Specification Requests (**JSRs**) and **JDK Enhancement Proposals (JEPs)**
- **Ongoing ecosystem participation**
 - Participate in JEPs creation and/or evaluation
 - Participate in committees
 - "Adopt" some JDK features and help
 - Everyone is invited!

Subject: [JCP] LAST DAY! Nominations for 2021 Executive Committee Election Close Tonight!

Dear JCP Member,
There are only a few hours remaining to yourself for an Elected or Associate Seat in the Java Community Process (JCP) program 2021 Executive Committee (EC) Election.



Trust via OpenJDK

- **Preserving Java Values**

Openness

Ease of Use

Independence

Reliability

- **Continual Delivery**

Innovation

Performance

Stability

Security

- **Trust in OpenJDK**

Open innovation

Open development

Ecosystem support



Innovation with Incubators and Preview

- **Incubator Features**

- New API and tools that, after improvements and stabilization, will most likely be included in the next JDK

- **Preview Features**

- Features believed to be implemented but subject to changes before becoming final

- **Experimental Features**

- Test-bed to gather feedback on nontrivial enhancements

- **Early Access Releases**

- Allowing developers to prepare for the next version of JDK in advance

- As a part of other **JDK development projects**

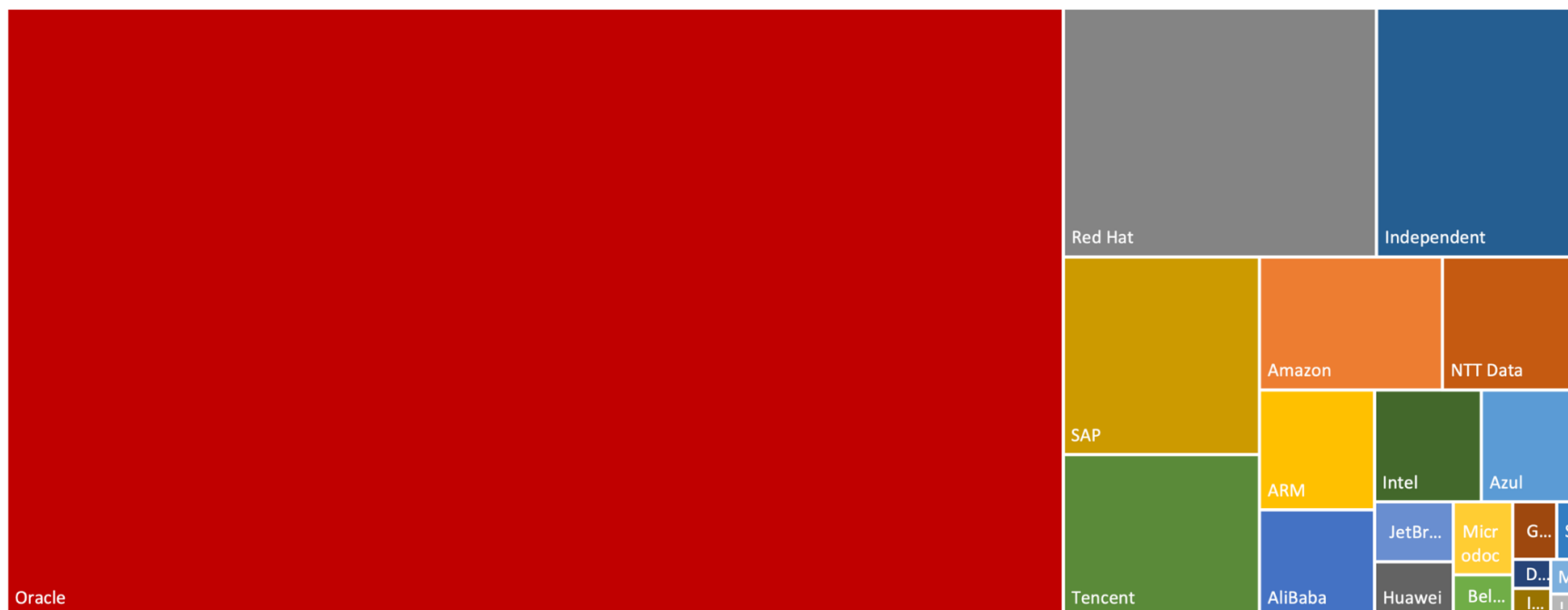
- Amber, Valhalla, Panama, Loom, Leyden, ZGC and many others



Creating Java Together

- Issues fixed in JDK 11-17 per **organization**

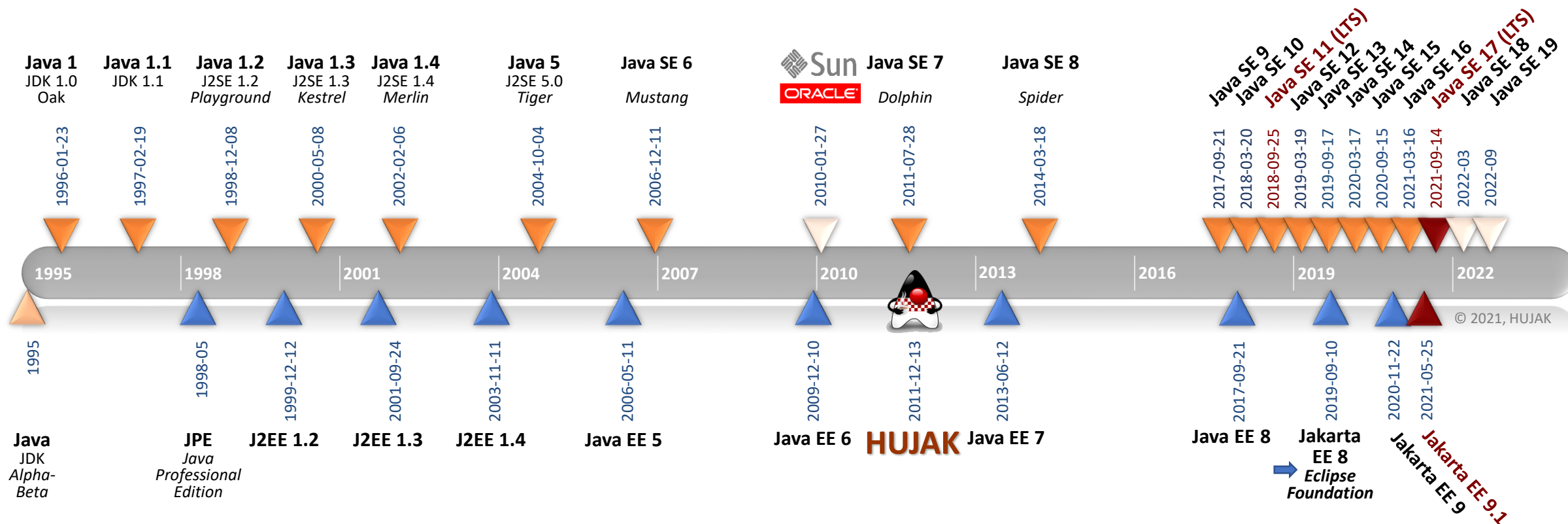
- Led by **Oracle**
- Significant contribution by many others (**Red Hat, SAP, Tencent, Amazon, NTT Data, ARM, Intel, Azul, Alibaba, JetBrains, Microdoc, Huawei ...**)





Java Timeline

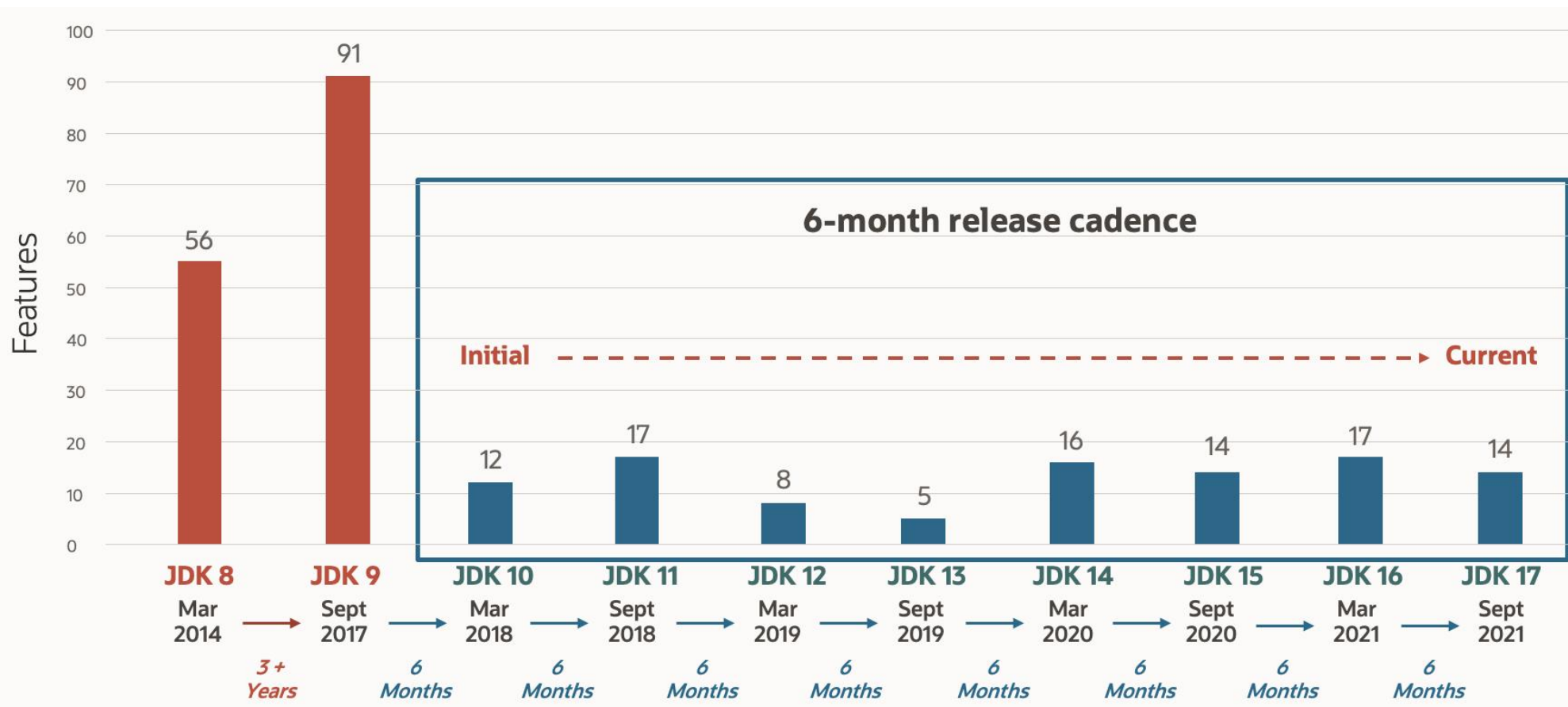
- A looong history ...





The Pace of Evolution

- **8 Feature Releases** in 3 years presented many new features in **JEPs**

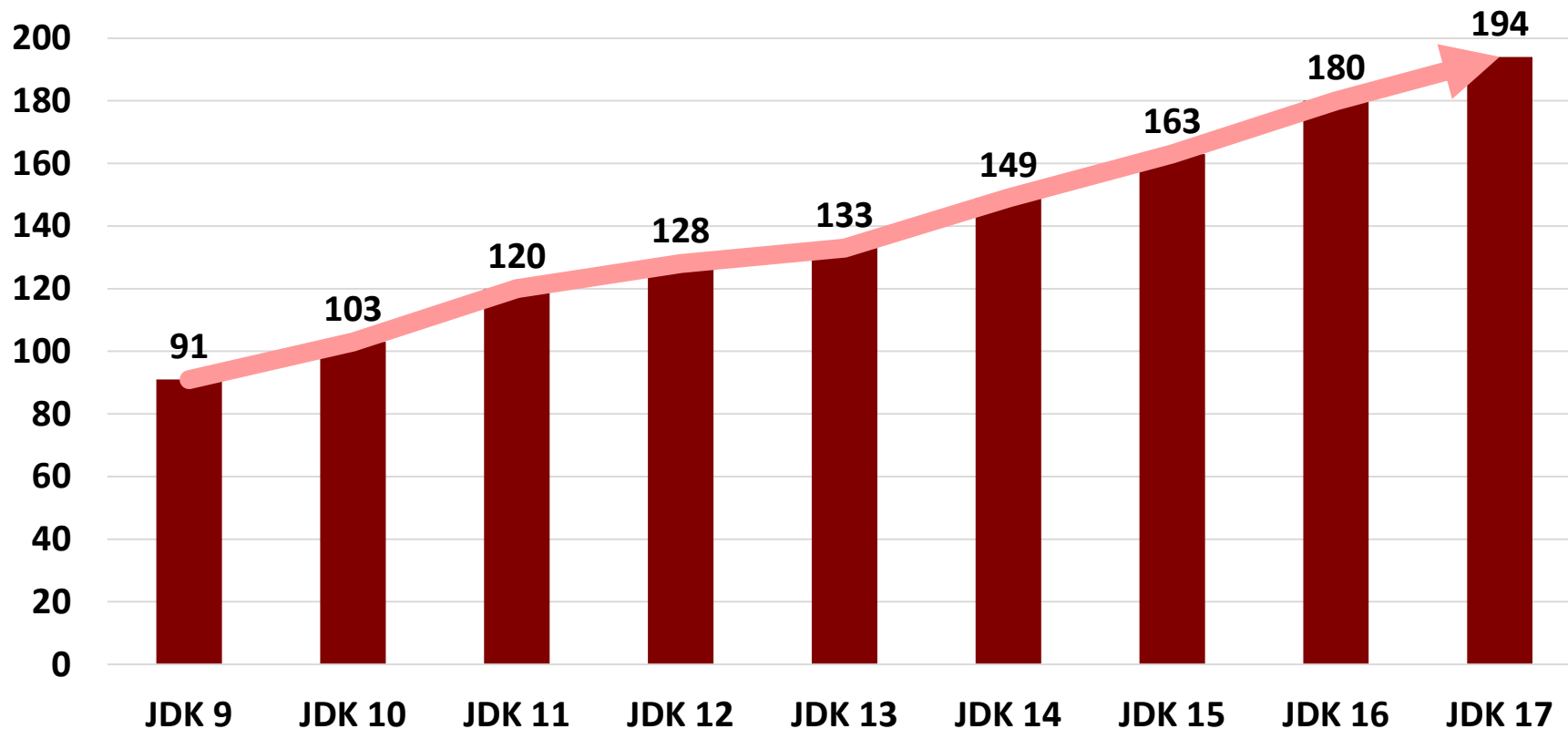


Source: The Arrival of Java 17, Sharat Chander, Sep 2021



Constant Evolution through JEPs

The number of JEPs since JDK 8

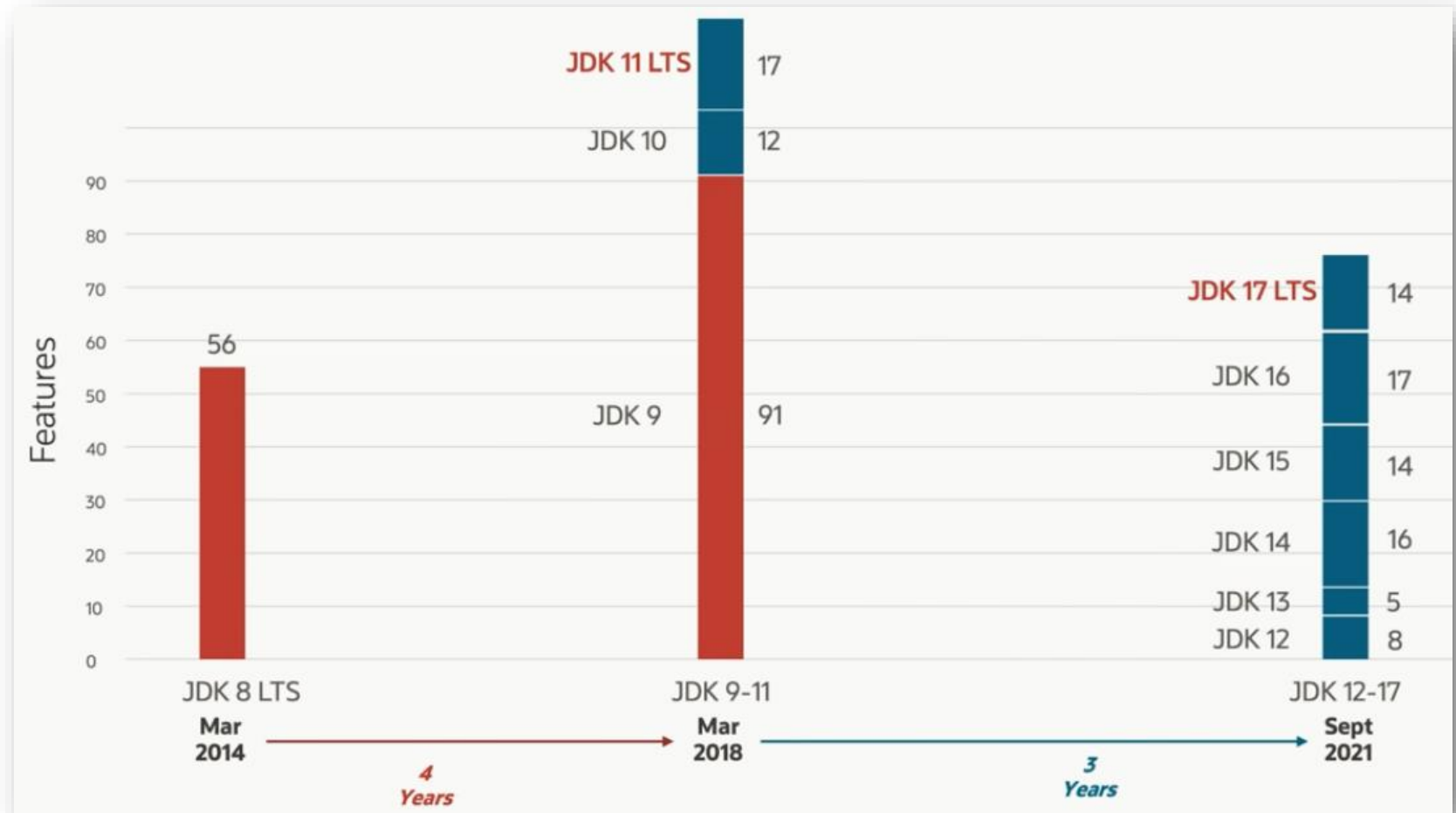


- The number of JEPs is on the **constant rise**
- Continuous flow of **new features**
- But what about Long Term Support?



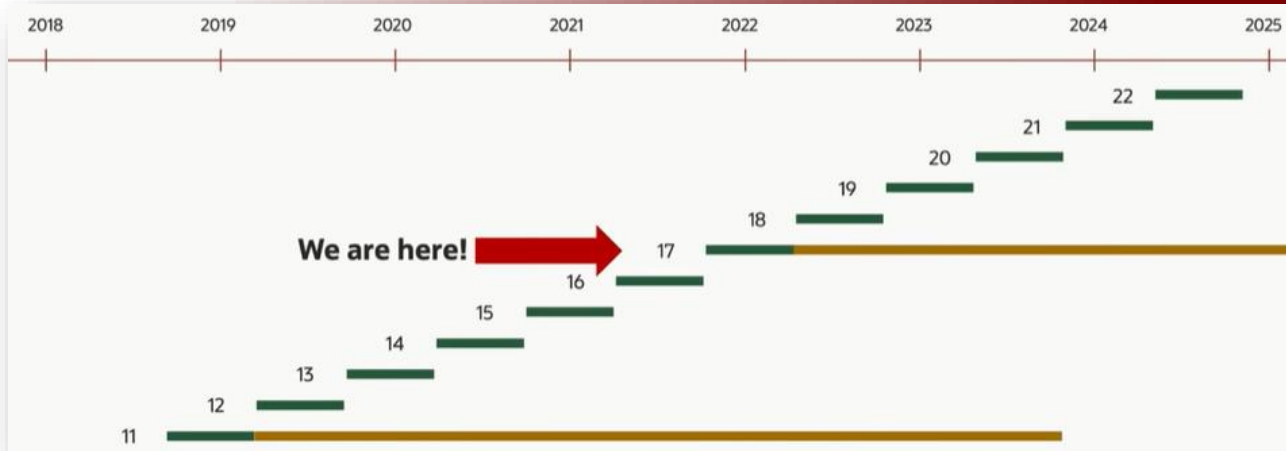
LTS Releases Include Many JEPs

- Accumulating improvements over 6-months feature releases every 3 years
- LTS (Long Term Support) Releases presented a number of JEPs
 - JDK 8 – 56 JEPs
 - JDK 9-11 – 120 JEPs
 - JDK 12-17 – 74 JEPs



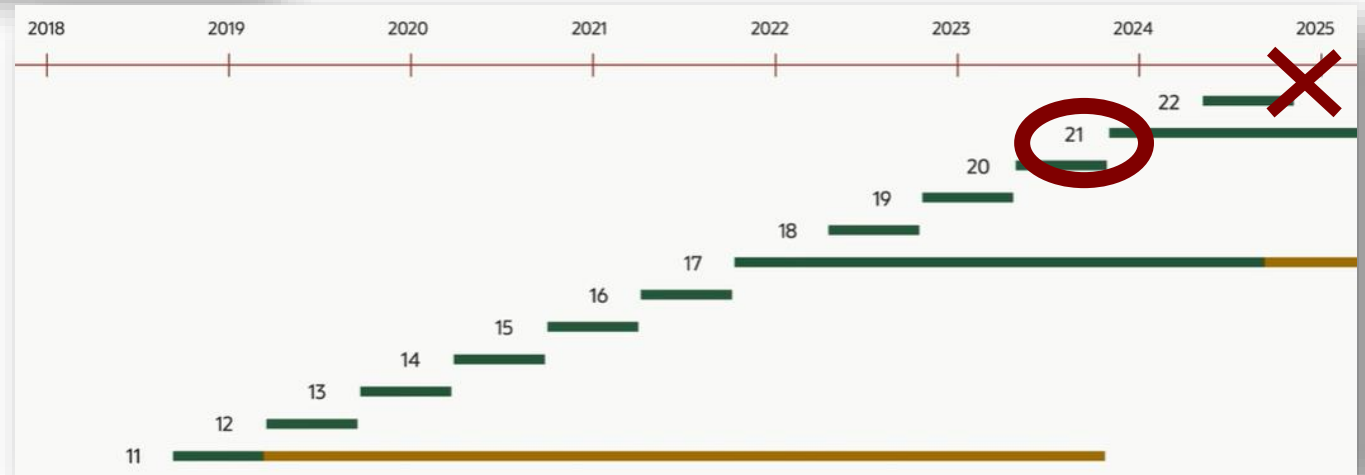


LTS Release – Every ? Years



- Demand for **LTS** (Long Term Support) has grown
- Recent surveys show 6-month (non-LTS) releases being used by between a 25-50% of developers
- Only about half of those note their use is in production

- Proposing the new LTS release schedule → **every 2 years**





LTS Every **2 Years** Proposal

- Oracle proposes to shift the cadence of JDK **LTS** releases from every **3 years** to **every 2 years**
 - Giving enterprises and developers more frequent opportunities to move an application onto an LTS release, knowing it will remain supported long-term with necessary stability, security and performance updates
- If accepted, next JDK LTS release will be **JDK 21** in **2023** rather than JDK 23 in 2024
 - It also makes the use of 6-month releases more appealing as organizations know the next available LTS will always be less than 2 years away
- *Source:* Moving Java Forward Even Faster by Mark Reinhold, Sep 14, 2021, mreinhold.org/blog/forward-even-faster
- *More:* Moving the JDK to a Two Year LTS Cadence by Donald Smith, Sep 14, 2021, blogs.oracle.com/java/post/moving-the-jdk-to-a-two-year-lts-cadence



LTS Support and Updates

- Duration of the **LTS support** should remain **unchanged**
 - For example, **Oracle** will support LTS releases at least of 5 or 8 years, so JDK 17 will receive updates with Premier support until 2026 and Extended support until 2029
 - Another vendor, **Azul** also offers LTS Production support until 2029
 - **Red Hat** usually offers support for 6 years
 - Support from other vendors could have different end dates and limitations
- **Critical Updates** available for LTS releases (JDK 8, JDK 11, and JDK 17)
- What about (non)commercial licenses (for production)?



Is Java "Free"?

- For **free** use of **OpenJDK** with **GPLv2+CE** license
- **Updates** refers to code patches – typically **free of charge**
- **Support** means fixing bugs and answering questions – was **never free of charge**
- **Oracle JDK 8** can be used **indefinitely for free**
 - Of course, without any further security patches and bug fixes
- **Oracle JDK 11-16** in **production** used with **commercial Java SE** subscription
 - Under **Oracle Technology Network (OTN)** license
 - Completely free JDK 11-16 are only OpenJDK binaries
- What about **JDK 17**?



OpenJDK or Commercial JDK?

All I'm offering is the truth – nothing more





Java is (Finally Completely) Free!



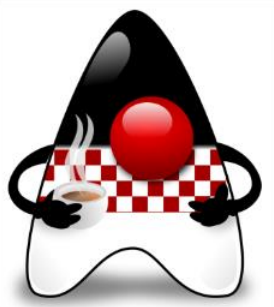
- **Oracle JDK 17** comes with **Oracle No-Fee Terms and Conditions (NFTC)** licensing
 - Oracle JDK permits **free use** for all users, even commercial and production use
 - Redistribution is permitted as long as it is not for a fee
 - www.oracle.com/downloads/licenses/no-fee-license.html
- Oracle will provide free releases and updates (starting with Oracle JDK 17) and continue for at least **one full year** after the **next** LTS release
 - Developers and organizations can now easily download, use, share and redistribute the Oracle JDK without needing a click-through
 - Prior versions are not affected by this change
- Additionally, Oracle will continue to provide Oracle **OpenJDK** releases under GPL
 - On the same releases and schedule as it has since Java 9
- *Source:* Introducing the Free Java License by Donald Smith, Sep 14, 2021, blogs.oracle.com/java/post/free-java-license



Java Subscriptions from Oracle

- **Oracle Java SE Desktop Subscription** – approx. 120-200 kn
 - Combines Java SE Licensing and Support for use on Desktop deployments
 - 1 Year Term Subscription; *Metric*: Named User Plus
- **Oracle Java SE Subscription** – approx. 1200-2000 kn
 - Combines Java SE Licensing and Support for use on Desktops, Servers or Cloud deployments
 - 1 Year Term Subscription; *Metric*: Processor
- **Oracle Java Development Tools Support** – approx. 8000 kn
 - Support for NetBeans, Oracle JDeveloper, and Oracle Enterprise Pack for Eclipse
 - 1 Year Term Subscription; *Metric*: Named User Plus





Let's see what is **inside** Java?



JDK 8 – ancient history



- JDK 8 in **March 2014**
- We hope that we will **not** talk about JDK 8 anymore 😊
- Ooops, are you **still using it?**





JDK 9 – very, very old news?



- **JDK 9 in September 2017**
 - **Many** new features and APIs (after 3.5 years)
- **90 JEPs included**
- The most important – **Java Platform Module System (JPMS)**
 - All core Java libraries are now modules (JEP 220) – 97 modules
- **"New" 6-months OpenJDK release model**
 - New features included (only) **when ready**
 - Feature release versions every **6 months** (in March & September)
 - Update releases **quarterly** (in January, April, July, and October)
- **Long-term support (LTS) feature release every 3 (or now 2) years**
 - Updates available for at least 3 years

OpenJDK



JDK 10 – new, and already old?



- **JDK 10** in **March 2018**

- **109** new features and **APIs** (after 6 months)

- **12 JEPs** included (only?)

- 286: **Local-Variable Type Inference** ← *vars*
- 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



JDK 11 – the "old" LTS version



- **JDK 11** in **September 2018**

- **90** new features and **APIs!**

- **LTS version** (first in a long time, after JDK 8 in 2014)

- **17 JEPs** included:

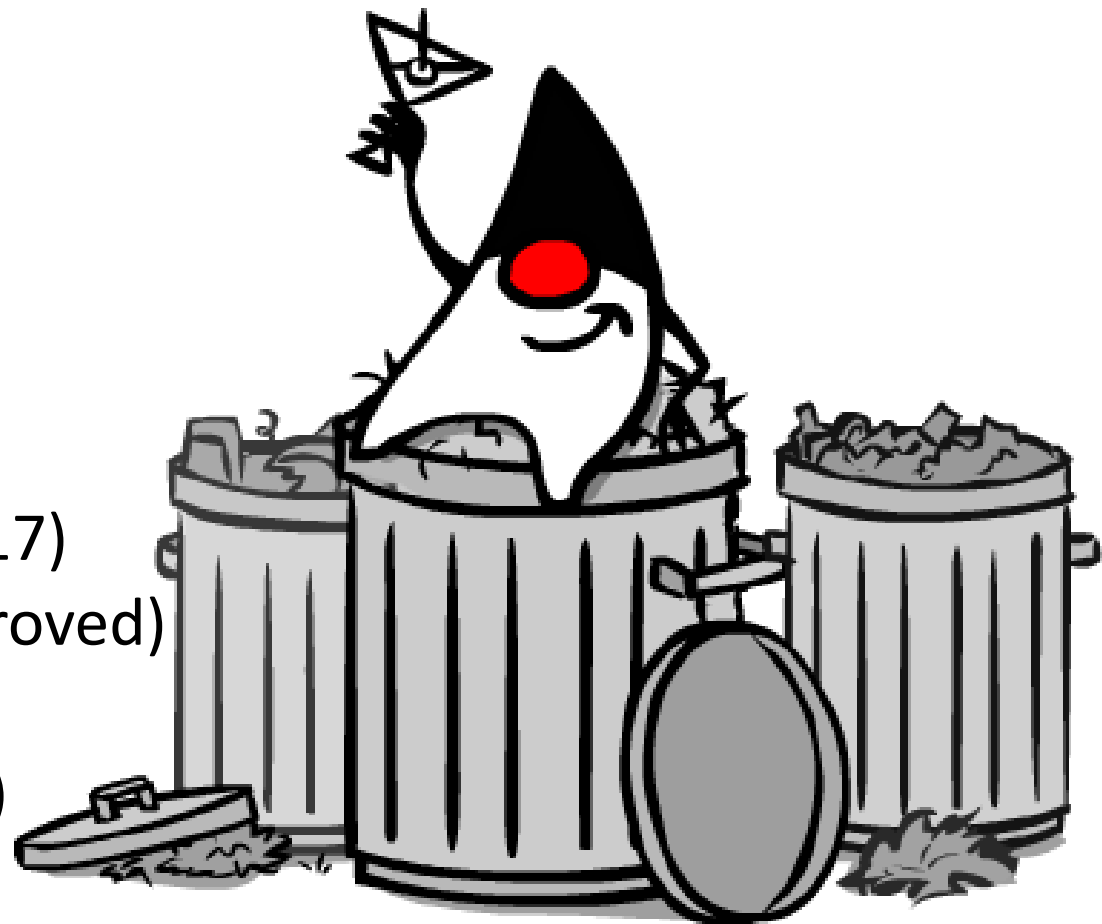
- 181: Nest-Based Access Control
- 309: Dynamic Class-File Constants
- 315: Improve Aarch64 Intrinsic
- 318: Epsilon: A No-Op Garbage Collector
- 320: Remove the Java EE and CORBA Modules
- 321: **HTTP Client** (Standard)
- 323: **Local-Variable Syntax for Lambda Parameters**
- 324: Key Agreement with 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



New Garbage Collectors

Many GCs to choose from:

- **Serial GC**
- **Parallel GC** (and **Parallel Old GC**)
- **CMS GC** (deprecated in Java 9)
- **G1** (Garbage-First) **GC** (default since Java 9)
 - **Parallel Full GC** for G1 (updated in Java 10)
 - Improved in Java 12+
- **ZGC** (experimental in Java 11, improved in 12-17)
- **Shenandoah GC** (experimental in Java 12, improved)
- **Azul's C4 GC**
- **Epsilon GC** (no-op GC, experimental in Java 11)
- ...





JDK 12 – with the first "Preview"

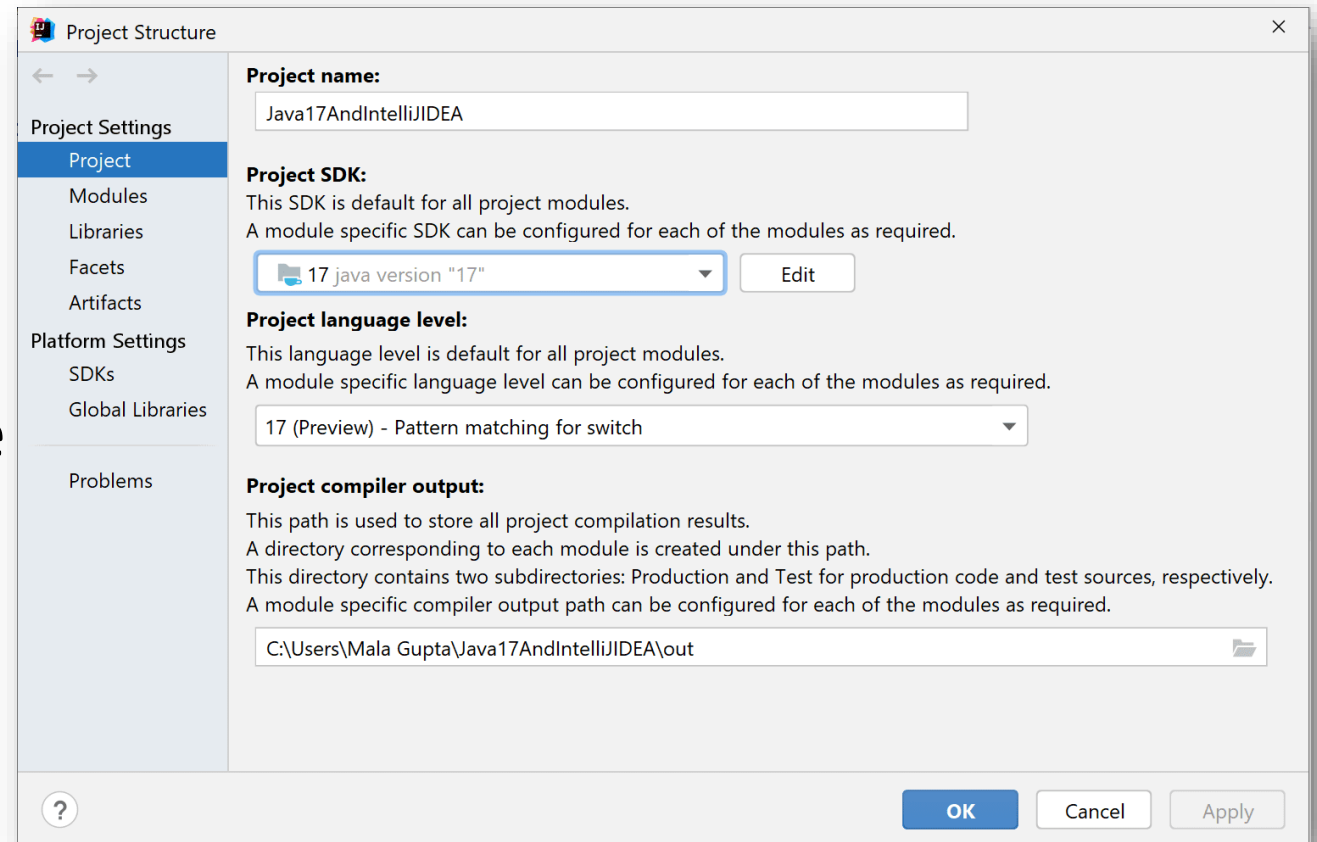


- **JDK 12 in March 2019**
 - **39 new features and APIs** – relatively small number
- **8 JEPs included:**
 - 189: **Shenandoah: A Low-Pause-Time Garbage Collector (Experimental)**
 - 230: Microbenchmark Suite
 - 325: **Switch Expressions (Preview)**
 - 334: JVM Constants API
 - 340: One AArch64 Port,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



What are **Preview** Features?

- **Preview language** (or VM) **features** are **fully implemented** and **fully specified**, yet **impermanent**
 - Made available in a release to get real world use feedback from developers
- To try out a preview feature it has to be **enabled** at compile time and at runtime
- Use **--enable-preview**






Switch Expressions (Preview)

```
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);
};
```

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



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

Used as an expression
No fall through
No default needed



JDK 13 – "only 5" JEPs



- **JDK 13 in September 2019**
 - **81** new features and **APIs** – relatively small n
- **5 JEPs** included (only):
 - 350: Dynamic CDS Archives
 - 351: **ZGC**: Uncommit Unused Memory
 - 353: Reimplement the Legacy Socket API
 - 354: **Switch Expressions (Second Preview)**
 - 355: **Text Blocks (Preview)**





Switch Expressions (2nd Preview)

- When working switch expression, if a full block is needed, a new **yield statement** is introduced
- It **yields a value** that becomes the value of the enclosing switch expression

```
int j = switch (day) {  
    case MONDAY    -> 0;  
    case TUESDAY  -> 1;  
    default        -> {  
        int k = day.toString().length();  
        int result = f(k);  
        yield result;  
    }  
}
```



JDK 14 – a lot of new features



- JDK 14 on **March 17, 2020**

- **Many** new features and **APIs** at openjdk.java.net/projects/jdk/14/

- **16 JEPs** targeted:

- 305: **Pattern Matching for instanceof (Preview)**
- 343: **Packaging Tool (Incubator)**
- 345: **NUMA-Aware Memory Allocation for G1**
- 349: **JFR Event Streaming**
- 352: **Non-Volatile Mapped Byte Buffers**
- 358: **Helpful NullPointerExceptions**
- 359: **Records (Preview)**
- 361: **Switch Expressions (Standard)**
- 362: **Deprecate the Solaris and SPARC Ports**
- 363: **Remove the ~~Concurrent Mark Sweep (CMS)~~ Garbage Collector**
- 364: **ZGC on macOS**
- 365: **ZGC on Windows**
- 366: **Deprecate the ~~ParallelScavenge + SerialOld~~ GC Combination**
- 367: **Remove the Pack200 Tools and API**
- 368: **Text Blocks (Second Preview)**
- 370: **Foreign-Memory Access API (Incubator)**



Helpful NullPointerExceptions (JEP 358)

- Improve the usability of **NullPointerExceptions** (NPEs) generated by the JVM by **describing precisely which variable was *null***
 - Offers helpful information to developers about the premature termination
 - Improves program understanding by clearly associating a dynamic exception with static program code
- Computed at runtime through bytecode access path analysis
- Reduces the confusion about NullPointerExceptions
- *Example: a.b.c.i = 1;* could generate the message
Exception in thread "main" java.lang.NullPointerException:
Cannot read field 'c' because 'a.b' is null.
- *Example: a[i][j][k] = 99;* could generate the message
Exception in thread "main" java.lang.NullPointerException:
Cannot load from object array because 'a[i][j]' is null.



Text Blocks

- **SQL** example using a "two-dimensional" block of text

```
String query = ""
```

```
    SELECT "EMP_ID", "LAST_NAME" FROM "EMPLOYEE_TB"  
    WHERE "CITY" = 'INDIANAPOLIS'  
    ORDER BY "EMP_ID", "LAST_NAME";  
"";
```

- **Polyglot language** example using a "two-dimensional" block of text

```
ScriptEngine engine = new ScriptEngineManager().getEngineByName("js");
```

```
Object obj = engine.eval("""
```

```
    function hello() {  
        print('"Hello, world"');  
    }  
    hello();  
""");
```



JDK 15 –



- **JDK 15 on September 15, 2020**

- New features and APIs at <https://openjdk.java.net/projects/jdk/15/>

- **14 JEPs targeted:**

- 339: Edwards-Curve Digital Signature Algorithm (EdDSA)
- 360: **Sealed Classes (Preview)**
- 371: **Hidden Classes**
- 372: Remove the ~~Nashorn~~ JavaScript Engine
- 373: Reimplement the Legacy DatagramSocket API
- 374: Disable and Deprecate Biased Locking
- 375: **Pattern Matching for instanceof (Second Preview)**

- 377: **ZGC: A Scalable Low-Latency Garbage Collector**
- 378: **Text Blocks**
- 379: **Shenandoah: A Low-Pause-Time Garbage Collector**
- 381: Remove the Solaris and SPARC Ports
- 383: **Foreign-Memory Access API (Second Incubator)**
- 384: **Records (Second Preview)**
- 385: Deprecate RMI Activation for Removal



Sealed Classes

- *Example:* Sealed class may omit permit if subclasses are defined in same file

```
abstract sealed class Shape { ...  
    final class Circle extends Shape { ... }  
    final class Rectangle extends Shape { ... }  
    final class Square extends Shape { ... }  
}
```

- Anonymous classes and local classes cannot be permitted subtypes of a sealed class



JDK 16 –



- **JDK 16 on March 16, 2021**

- New features and APIs at <https://openjdk.java.net/projects/jdk/16/>

- **17 JEPs targeted:**

- 338: **Vector API (Incubator)**
- 347: Enable C++14 Language Features
- 357: Migrate from Mercurial to Git
- 369: Migrate to GitHub
- 376: **ZGC: Concurrent Thread-Stack Processing**
- 380: Unix-Domain Socket Channels
- 386: Alpine Linux Port
- 387: Elastic Metaspace

- 388: Windows/AArch64 Port
- 389: **Foreign Linker API (Incubator)**
- 390: Warnings for Value-Based Classes
- 392: Packaging Tool
- 393: **Foreign-Memory Access API (Third Incubator)**
- 394: **Pattern Matching for instanceof**
- 395: **Records**
- 396: Strongly Encapsulate JDK Internals by Default
- 397: **Sealed Classes (Second Preview)**



Records

- *Example: A point*

```
class Point {  
  
    final double x;  
    final double y;  
  
    public Point (double x, double y) {  
        this.x = x;  
        this.y = y;  
    }  
  
    public double x() { return x; }  
    public double y() { return y; }
```

```
@Override  
public double equals (Object o) {  
    if (...)  
        ...  
    return ...  
}  
  
@Override  
Public double hashCode () {  
    return ...  
}  
  
@Override  
Public double toString() {  
    return ...  
}
```



Records

- *Example: A point*

```
record Point { }
```

```
final double x;  
final double y;
```

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

```
public double x() { return x; }
```

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

Sometimes data is just ... data.

```
@Override  
public double equals (Object o) {  
    if (...  
        ..  
        return ...  
}
```

Mark Reinhold

```
@Override  
Public double hashCode () {  
    return ...  
}
```

```
@Override  
Public double toString() {  
    return ...  
}
```



Pattern Matching for **instanceof**

- First preview as JEP 305 in JDK 14, Standard as JEP 375 in JDK 16

- *Example:*

```
if (obj instanceof String s && s.length() > 5) {..  
    s.contains(..) ..}
```

- Another example:

```
@Override public boolean equals(Object o) {  
    return (o instanceof CaseInsensitiveString cis) &&  
        cis.s.equalsIgnoreCase(s);  
}
```



JDK 17 – the "new" LTS



- **JDK 17 on September 14, 2021**

- New features and APIs at <https://openjdk.java.net/projects/jdk/17/>

- **14 JEPs targeted:**

- 306: Restore Always-Strict Floating-Point Semantics
- 356: **Enhanced Pseudo-Random Number Generators**
- 382: New macOS Rendering Pipeline
- 391: macOS/AArch64 Port
- 398: **Deprecate the Applet API for Removal**
- 403: Strongly Encapsulate JDK Internals
- 406: **Pattern Matching for switch** (Preview)
- 407: Remove RMI Activation
- 409: **Sealed Classes**
- 410: **Remove the Experimental AOT and JIT Compiler**
- 411: Deprecate the Security Manager for Removal
- 412: **Foreign Function & Memory API** (Incubator)
- 414: **Vector API** (Second Incubator)
- 415: Context-Specific Deserialization Filters



Pattern Matching for switch

- *Example:*

```
static String formatterPatternSwitch(Object o) {  
    return switch (o) {  
        case null          -> "null";  
        case Integer i     -> String.format("int %d", i);  
        case Long l        -> String.format("long %d", l);  
        case Double d      -> String.format("double %f", d);  
        case String s      -> String.format("String %s", s);  
        default            -> o.toString();  
    };  
}
```



Foreign Function & Memory API

- JEP 412: **Foreign Function & Memory API** (Incubator)
<https://openjdk.java.net/jeps/412>
- Introducing API to of statically-typed, pure-Java access to native code
- Simplifying error-prone process of binding to a native library
- Java Native Interface (JNI) was a bit hard an brittle
- Use any native library
- Foreign Linker API supports foreign function support
- Foreign Memory Access API allows access to memory outside of heap



Tooling Support for JDK 17

- Timely support for new features by tools and libraries helps drive developer productivity
- The efforts of leading IDE vendors whose most timely updates offer developers support for current Java versions
- Developers can already take advantage of Java 17 support today within:
 - **JetBrains IntelliJ IDEA 2021.2.1**
 - **Eclipse IDE 2021-09 (4.21)** via a separate marketplace solution
 - **NetBeans 12.5** with experimental support for JDK 17
 - **Visual Studio Code**
 - ...



JDK 18 – Foreseeable Future



- JDK 18 in **March 2022**
 - New features and APIs at <https://openjdk.java.net/projects/jdk/18/>
- JEPs targeted so far:
 - 400: UTF-8 by Default
 - 413: Code Snippets in Java API Documentation
 - 417: **Vector API** (Third Incubator)



Vector API

- JEP 414: **Vector API** (Second Incubator) <https://openjdk.java.net/jeps/414>
- API to express vector computations that reliably compile at runtime to optimal vector hardware instructions
 - Achieve superior performance to equivalent scalar computations
- Taking advantage of the Single Instruction Multiple Data (SIMD) instructions on most modern CPUs
- Allows developers to write complex vector algorithms in Java

```
a = b + c * z[i+0]
d = e + f * z[i+1]
r = s + t * z[i+2]
w = x + y * z[i+3]
```

4 multiplications
4 additions
4 assignments

```
a d r w = b e s x +SIMD c f t y *SIMD z[i+0]
z[i+1]
z[i+2]
z[i+3]
```

1 SIMD multiplication
1 SIMD addition
1 assignment



Projects – Long-term Java Future

OpenDJK Projects – <https://openjdk.java.net/projects/>

- Project **Amber** – incubator for smaller, productivity-oriented **language features** and **simplifying syntax**
- Project **Valhalla** – incubator project for **advanced JVM and language feature** candidates
- Project **Panama** – to interconnect JVM and native code
- Project **Loom** – to reduce complexity in writing concurrent applications
- Project **Metropolis** – JVM re-written in Java, i.e. "**Java on Java**"
- Project **Skara** – alternative SCM & code review for JDK (Git)



Project Amber

- **Right-sizing** language ceremony
 - Explore and incubate smaller, productivity-oriented Java language features
 - openjdk.java.net/projects/amber/
- Have been accepted as candidate JEPs under the OpenJDK
- Most features go through at least one round of *Preview* before becoming an official part of Java SE

OpenJDK

OpenJDK FAQ
Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds
Mailing lists
Wiki - IRC
Bylaws - Census
Legal

JEP Process

Source code
Mercurial
GitHub

Tools

Mercurial
Git
jreg harness

Groups

(overview)
Adoption
Build
Client Libraries
Compatibility & Specification
Review
Compiler
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects

(overview)
Amber
Annotations Pipeline
2.0
Audio Engine
Build Infrastructure
CRaC
Caciocavallo
Closures
Code Tools
Coin
Common VM
Interface
Compiler Grammar
Detroit
Developers' Guide
DevOps

Project Amber

The goal of Project Amber is to explore and incubate smaller, productivity-oriented Java language features that have been accepted as candidate JEPs under the OpenJDK JEP process. This Project is sponsored by the Compiler Group.

Most Project Amber features go through at least one round of *Preview* before becoming an official part of Java SE. See JEP 12 for an explanation of the Preview process. For a given feature, there are separate JEPs for each round of preview and for final standardization. This page links only to the most recent JEP for a feature. Such JEPs will generally have links to earlier JEPs for the feature, as appropriate.

Status of JEPs

Currently in progress:

- JEP 405 Record Patterns and Array Patterns (Preview)

Delivered:

- JEP 409 Sealed Classes
- JEP 406 Pattern Matching for switch (Preview)
- JEP 397 Sealed Classes (Second Preview)
- JEP 395 Records
- JEP 394 Pattern Matching for instanceof
- JEP 384 Records (Second Preview)
- JEP 378 Text Blocks
- JEP 375 Pattern Matching for instanceof (Second Preview)
- JEP 368 Text Blocks (Second Preview)
- JEP 361 Switch Expressions
- JEP 360 Sealed Classes (Preview)
- JEP 359 Records (Preview)
- JEP 355 Text Blocks (Preview)
- JEP 354 Switch Expressions (Second Preview)
- JEP 325 Switch Expressions (Preview)
- JEP 323 Local-Variable Syntax for Lambda Parameters
- JEP 305 Pattern Matching for instanceof (Preview)
- JEP 286 Local-Variable Type Inference (var)

On hold:

- JEP 301 Enhanced Enums. (See [here](#) for an explanation)
- JEP 302 Lambda Leftovers
- JEP 348 Java Compiler Intrinsic for JDK APIs

Withdrawn:

- JEP 326 Raw String Literals (Preview). (Dropped in favor of Text Blocks; see [here](#) for an explanation.)



Project Amber – **Timeline** Example

- Improving the programming language continuously

	Java 10	Java 11	Java 12	Java 13	Java 14	Java 15	Java 16	Java 17
Local-Variable Type Inference - var	Standard	→						
Local-Variable Syntax for Lambda Parameters		Standard	→					
Switch Expressions			Preview	2 nd Preview	Standard	→		
Text Blocks				Preview	2 nd Preview	Standard	→	
Records					Preview	2 nd Preview	Standard	→
Pattern Matching for instanceof					Preview	2 nd Preview	Standard	→
Pattern Matching for switch								Preview
Sealed Classes						Preview	2 nd Preview	Standard



Project Valhalla

- Incubator project for more **advanced Java VM and language feature** candidates
- Problems to solve:
 - **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) in Draft – "*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) as Candidate – 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** and **native data access** from the JVM
- **Foreign function interface (FFI)** as a simple, safe and performant replacement for JNI, includes:
 - Native function calling from JVM (C, C++)
 - Native data access from JVM or inside JVM heap
 - Native library management APIs and native-oriented JIT optimizations
- Access to low-level hardware functionality from Java (vector instructions, special memory types)?
- Big Data, Machine Learning?



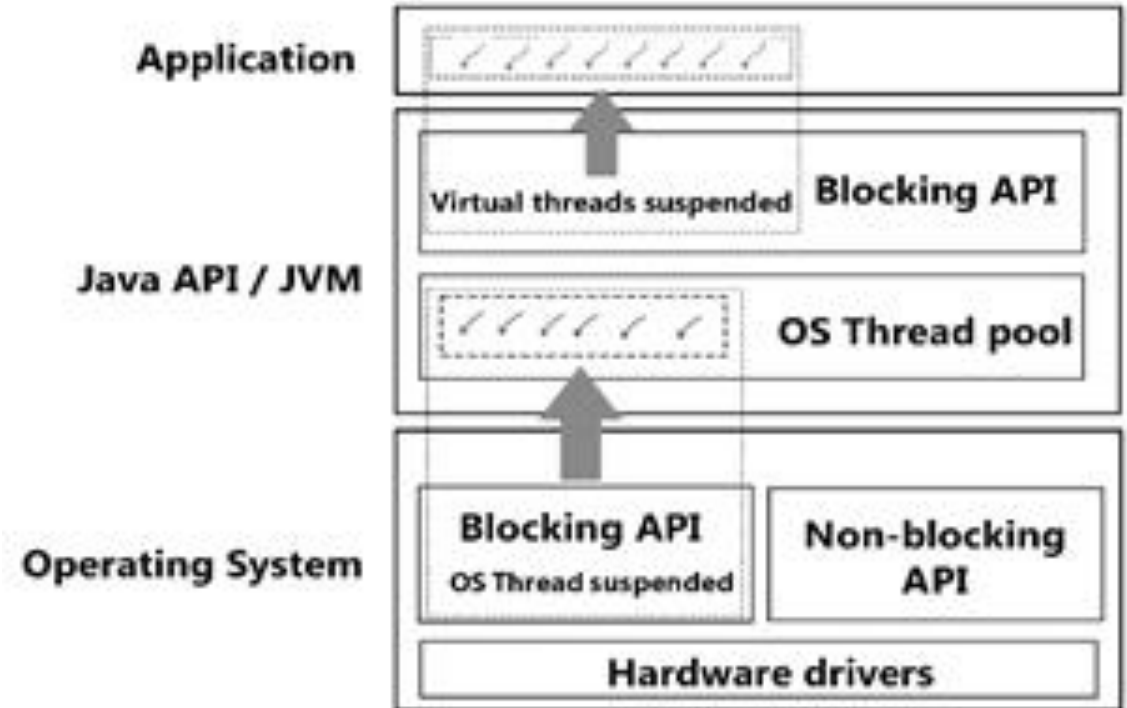
Project **Loom**

- **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
 - An expensive context switch between OS threads
- Project **Loom** – reducing complexity in writing concurrent applications via alternative, **user-mode thread implementations**
- Proposal for lightweight JVM-level threads called **Virtual Threads** as alternative implementation of threads
- Ordinary Java threads preserved, performance improved, and footprint reduced
 - Less memory and almost zero overhead when task switching



Virtual Threads in the JVM

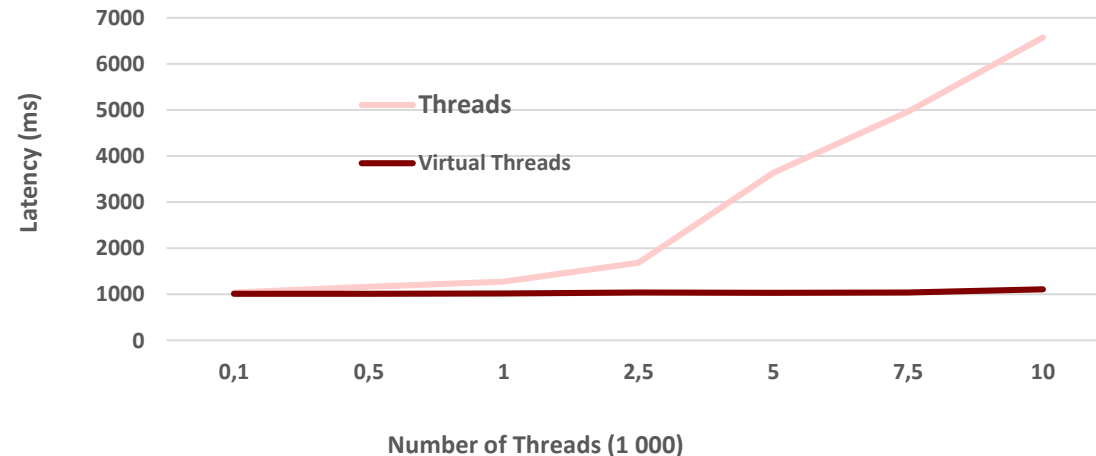
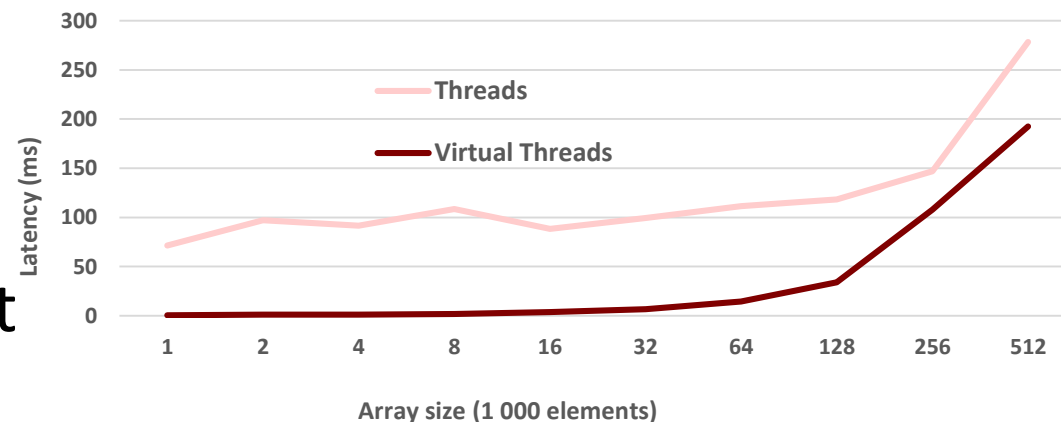
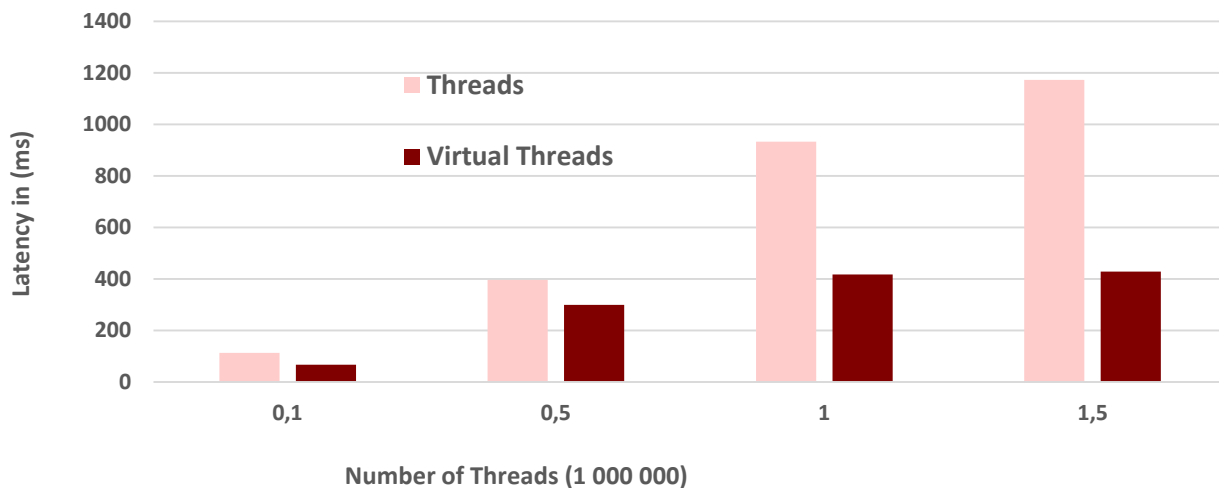
- **Virtual Threads (VT)** are a light-weight concurrency construct
- Created and managed by the JVM
- Flexible mapping to the OS Threads
 - A single VT can run on different OS Threads while performing a single task
- Consist of **continuation** and **scheduler**
- JVM controls VT mapping to OS Threads
 - while performing blocking operations which results in the more optimal use of resources compared to Threads





Our Test Cases of Virtual Threads

- **Faster creation?** Sequential run test
- **Better synchronization?** Multi-threaded merge sort
- **Higher concurrency level?** Parallel run test





Programming **Polyglotism** & other trends

- **Polyglot programming problems:**

- Cross-language interoperability
- General-purpose programming languages not-so-good performance
- Language tools – configuration, debugging...

- OpenJDK's Project **Metropolis**

- **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

- + **Quarkus**: Kubernetes Native Java stack tailored for **GraalVM** & OpenJDK **HotSpot**



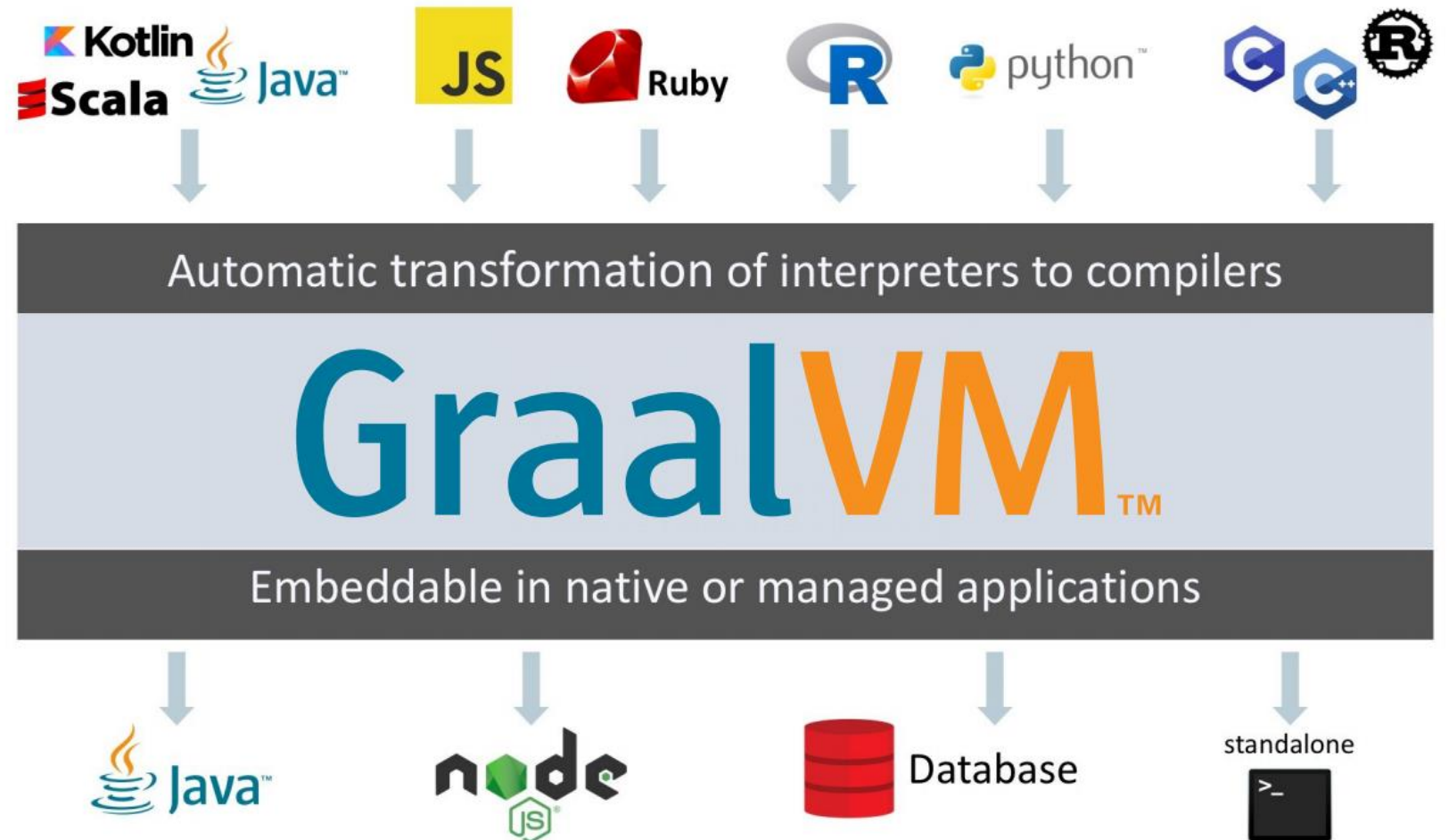
QUARKUS

GraalVM™



GraalVM – architecture

- **JVM** – HotSpot VM or other
- **JVM Compiler Interface (JVMCI)** – optimizing compiler
- **Graal Compiler** – new optimized compiler for JVM languages
- **Truffle** framework – any other language
- **Sulong** (LLVM) – high-performance Low-Level Virtual Machine bitcode interpreter
- + **Native images** with **Substrate VM**





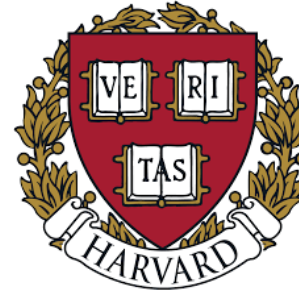
Where can you **learn** Java?

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



STANFORD

Caltech



ETH zürich

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



coursera

ORACLE
UNIVERSITY

Udemy

treehouse™

in LEARNING
WITH Lynda.com CONTENT



PLURALSIGHT

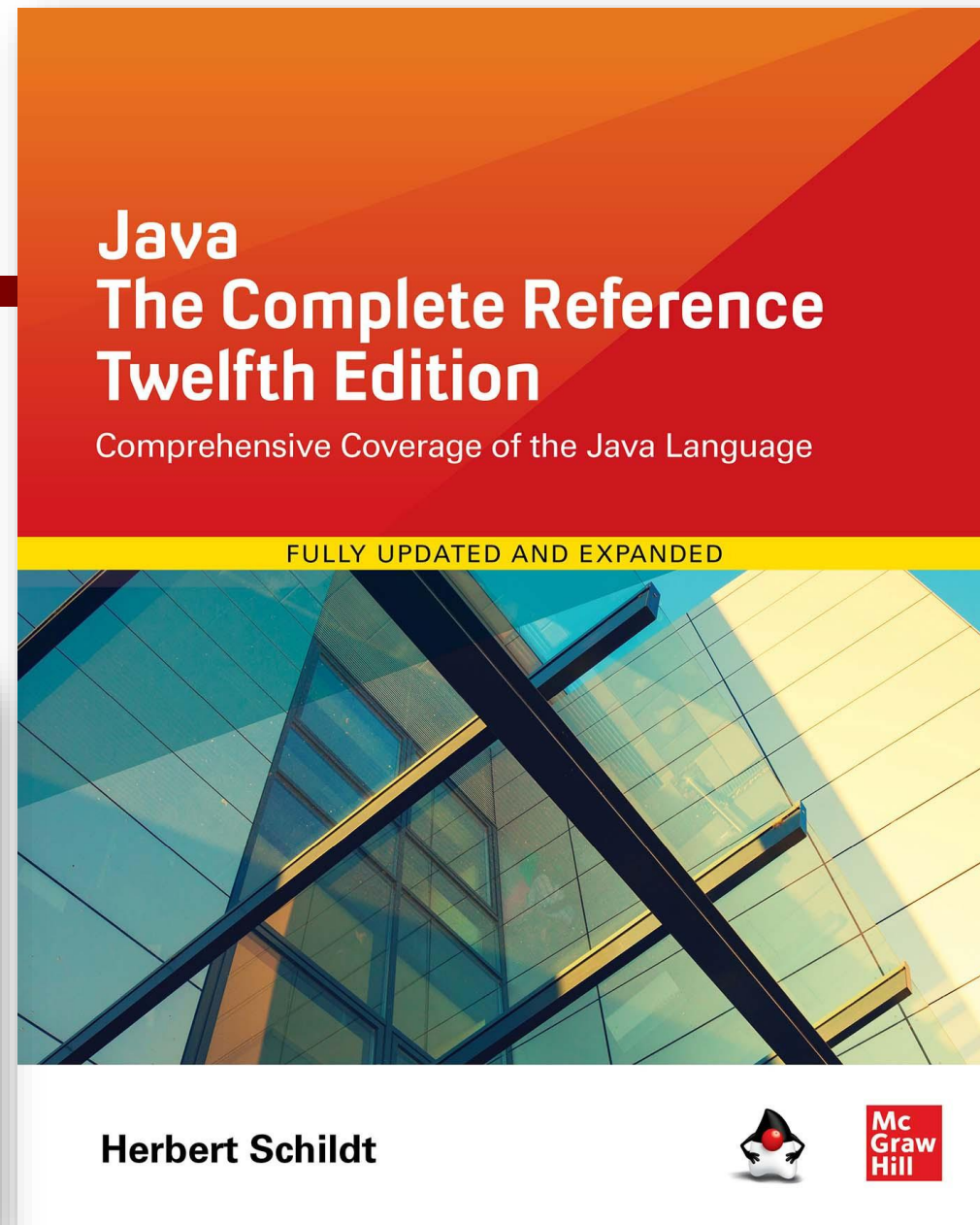
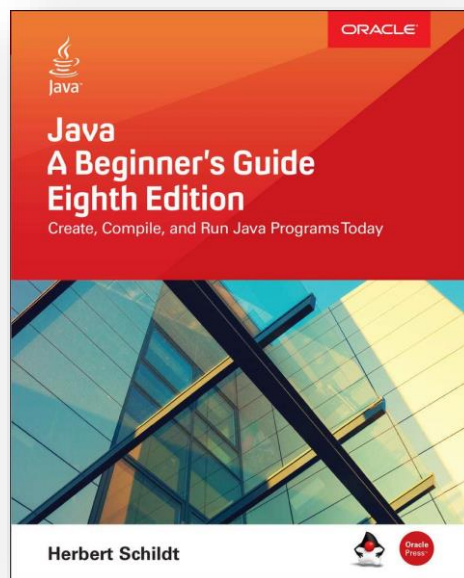


UDACITY



New Books

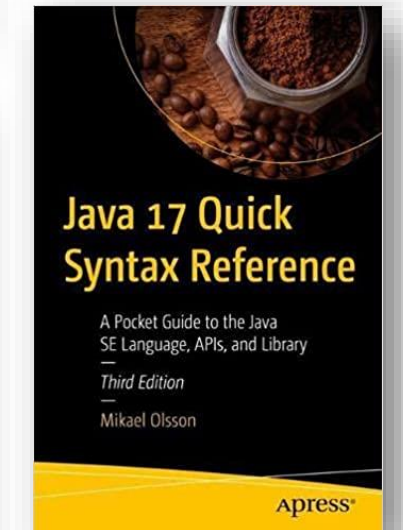
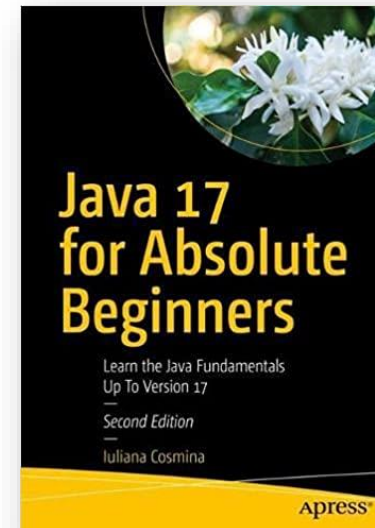
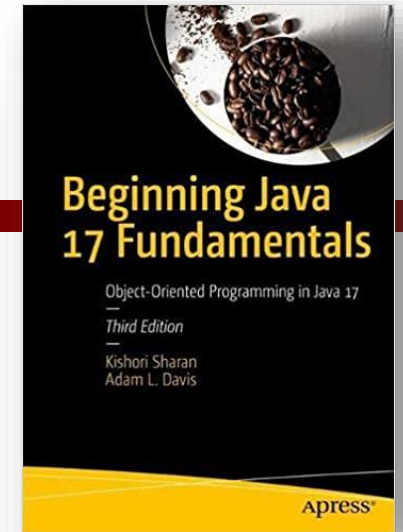
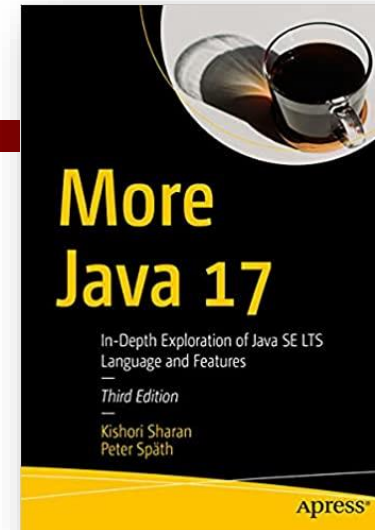
- **Java: The Complete Reference, 12th ed.,** Herbert Schildt, MGH, 2021
 - Preorder October 22nd, 2021
 - ISBN: 9781260463415
 - 1280 pages
 - Approx. 60 €
- **Java: A Beginner's Guide, 8th ed.,** Herbert Schildt, MGH, Nov 2018
 - ISBN: 9781260440218
 - 684 pages
 - Approx. 40 €





More New Books

- **Beginning Java 17 Fundamentals, 3rd ed.** by Kishori Sharan, Adam L. Davis, Apress, **Dec 2021**
 - 9781484273067, 800 pages, approx. 52 €
- **More Java 17, 3rd ed.** by Kishori Sharan, Peter Späth, Apress, **Jan 2022**
 - 9781484271346, approx. 56 €
- **Java 17 Quick Syntax Reference** by Mikael Olsson, Apress, **Nov 2021**
 - 9781484273708, 218 pages, approx. 32 €
- **Java 17 for Absolute Beginners** by Iuliana Cosmina, Apress, **Jan 2022**
 - 9781484270790, 600 pages, approx. 42 €





Learning Java

- **Java Learning Subscription by Oracle**
 - education.oracle.com/java-programming-learning-subscription/ls_40805
- Learning paths, Courses, Live sessions, User Forum
 - learn.oracle.com/ols/live-events/java-learning-subscription/40805
- Oracle University Free Resources
- Free basic Java training and accreditation
 - education.oracle.com/learning-explorer





Java Learning Paths

 Learning Path 7h 27m **Free**

Java Explorer

Learn Java programming basics such as variables, classes, objects, loops, arrays, and decision constructs. Get introduced to Java's

 [View Outline](#)

 Learning Path 33h 21m **Certification**

Java SE 11 Developer

NEW! Get a complete view of Java SE 11 technology and prepare for the certification exam.

 [View Outline](#)

 Learning Path 2h 55m

Java SE on OCI

Current version: Java 11

NEW! Craft custom Java solutions that leverage OCI DevOps, cloud-native services like functions, APIs and streaming, perform

 [View Outline](#)

 Learning Path 40m

Java SE 17 Developer

New! Welcome to this brand new Learning Path on everything Java SE 17. Take a look at our new Java SE 17: New Features course, and

 [View Outline](#)

 Learning Path 1h 6m

Beyond Java SE 11

Beyond Java SE 11 with JDKs 12, 13, 14, 15, and 16

NEW! This learning path discusses the JDK release cycle and significant new features introduced in the latest short term support

 [View Outline](#)

 Learning Path 10h 15m **Certification**

Java SE 11 Upgrade

Current Version: Java 11

This path is for experienced Java SE 8 programmers who need to update their skills to Java SE 11, including features introduced in


 [View Outline](#)

- **Java Explorer**
- 7.5 hours of free training
- **Topics:**
 - An Overview of Java;
 - Text and Numbers in Java;
 - Arrays, Conditions, and Loops;
 - Classes and Objects;
 - Exception Handling;
 - Inheritance and Interfaces;
 - Java on OCI
- Free assessment



Java Courses


- Java Courses
- Java Exam Preparation

 Course 18h 42m

Java SE: Programming II

This course is relevant for Java SE 11 and all applicable earlier versions.

This is a second-level course for programmers learning the Java language. It rounds-out the topics that were taught in the

 0% [View Outline](#)


 Course 19h 5m

Java SE: Programming I

This course is relevant for Java SE 11 and all applicable earlier versions.


This entry-level course is aimed at programmers who are new to Java and who need to learn its concepts, language

 0% [View Outline](#)

 Course 2h 53m **Exam Prep**

Prepare for Java SE Certification

Prepare for the Java SE certification by attempting and analyzing questions written in the style of what you'll likely encounter on the

 0% [View Outline](#)

 Course 52m **Exam Prep**

Prepare for Java SE 11 Upgrade Certification

120-817

NEW! Prepare for the Java SE 11 Upgrade Certification by attempting and analyzing questions written in the style of what you'll


 0% [View Outline](#)

 Course 1h 58m

Java on OCI: Develop a REST-based Microservice

Java on OCI: Develop a REST-based Microservice


 0% [View Outline](#)


 Course 8h 32m

Java SE: Exploiting Modularity and Other New Features

Course from Java SE Programming 2018 - Upgrade learning path


This course introduces the Java module system and other new features originally introduced in Java SE 9, and ultimately


 0% [View Outline](#)

 Course 40m

Java SE 17: New Features


Welcome to this short course which covers the new features introduced as part of the newly released JDK 17.


 0% [View Outline](#)

 Course 53m **Free**

Java SE 11 New Features


This comprises a single module where Oracle's Java product managers overview the features incorporated into Java SE 11. In


 0% [View Outline](#)

 Course 1h 6m

JDKs Beyond Java SE 11, 12, 13, 14, 15, and 16


NEW! This short course discusses the JDK release cycle and significant new features introduced in the latest short term support

 0% [View Outline](#)

 Course 58m

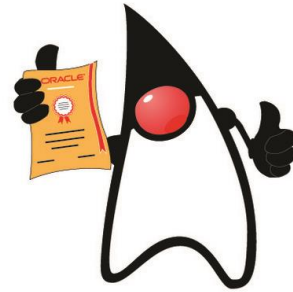
Develop, Test and Deploy Java Applications on OCI

NEW! Provision and configure an OCI compute instance and ADB to support basic Java SE functions like connecting to a

 0% [View Outline](#)



Java Certification



- **Oracle Java Certification Paths** (still for Java 11 only)
- **Oracle Certified Professional (OCP): Java SE 11 Developer**
 - Exam: Java SE 11 Developer 1Z0-819
 - Price: kn 1634 | Duration: 90 Minutes | Passing Score: 68%
- **Oracle Certified Professional (OCP): Java SE 11 Developer (upgrade from OCP Java 6, 7 & 8)**
 - **Retiring** on 2021-11-30
 - Exam: Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer 1Z0-817
 - Price: kn 1634 | Duration: 180 Minutes | Passing Score: 61% |





Is Java **really** "Moving Forward Faster"?

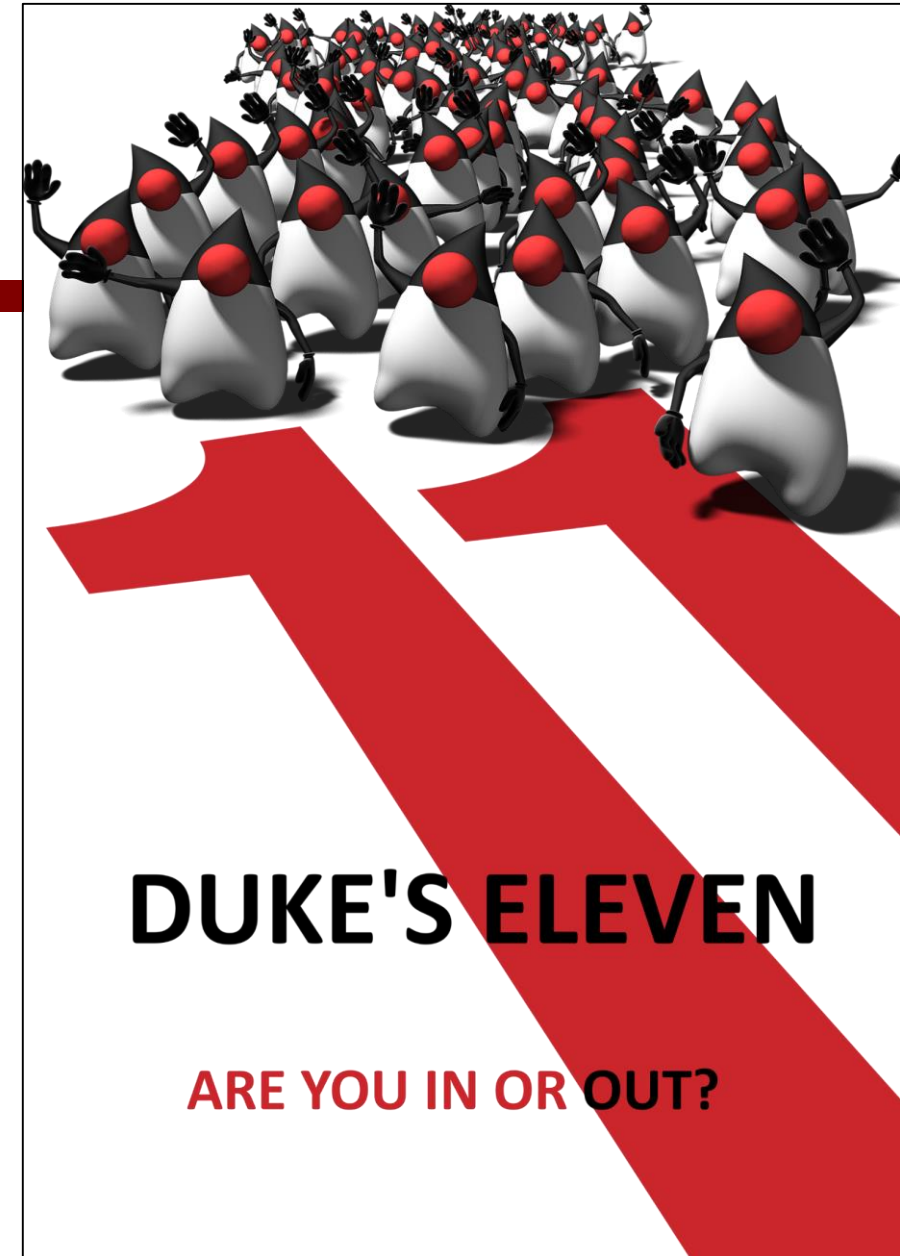
- Community opinion: well... **yeah!** 😊
- Much **more frequent** Java releases
- **Faster** access to **new** features
- **Many new** improvement ideas
- A lot of **maintenance** and **housekeeping**
- Java is **free**

Looking forward to **new things!**



What is our **advice**?

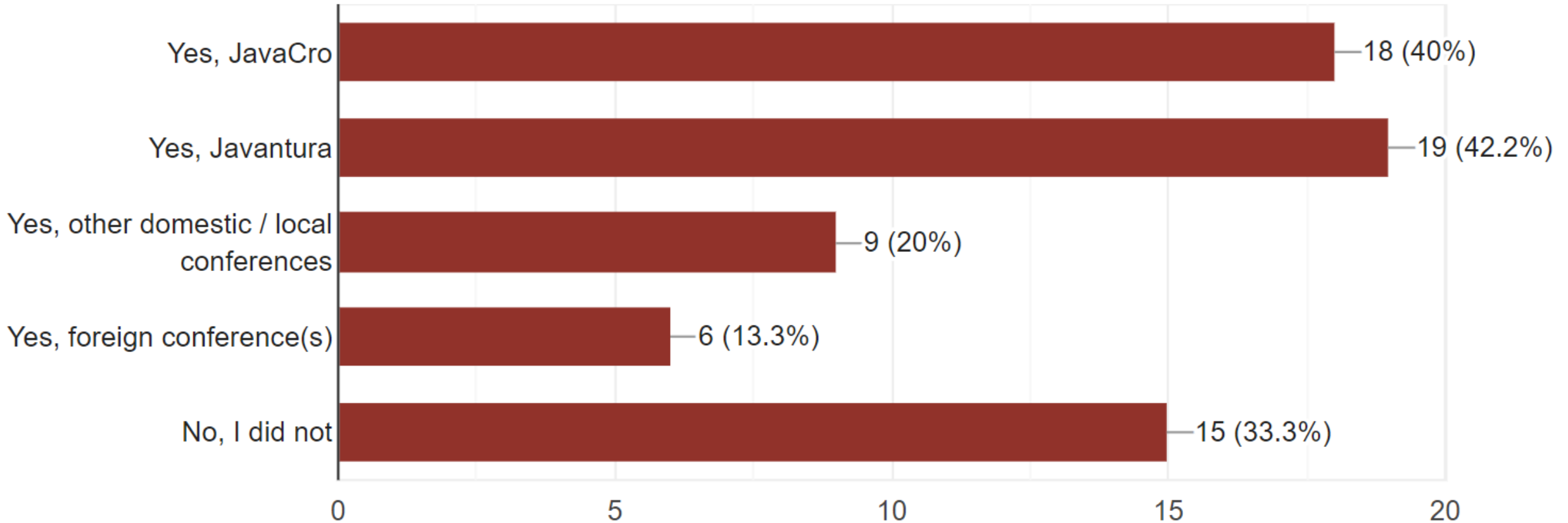
- Obviously – use Java **17 LTS** 😊
or at least use Java **11 LTS**
or something in between
- OpenJDK (any) or Oracle JDK or any other –
it's up to you 😊
- Try to **abandon** older versions (Java 8)
- Check what is **@Deprecated**
- Migrate every **3-6 months** or **2-3 years** (with LTS)
- Get involved more with **HUJAK!**
- Go to **conferences!**





Conferences

- Have you ever attended a **Java-related conference**?





Instead of **conclusion...**

Let's do another great **#Java** adventure

Javantura v8

2022, **Zagreb**

Warm welcome
from **HUJAK** and **CroDuke!**



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

