The Present and the Future of Java

dr. sc. **Branko Mihaljević**

dr. sc. **Aleksander Radovan**

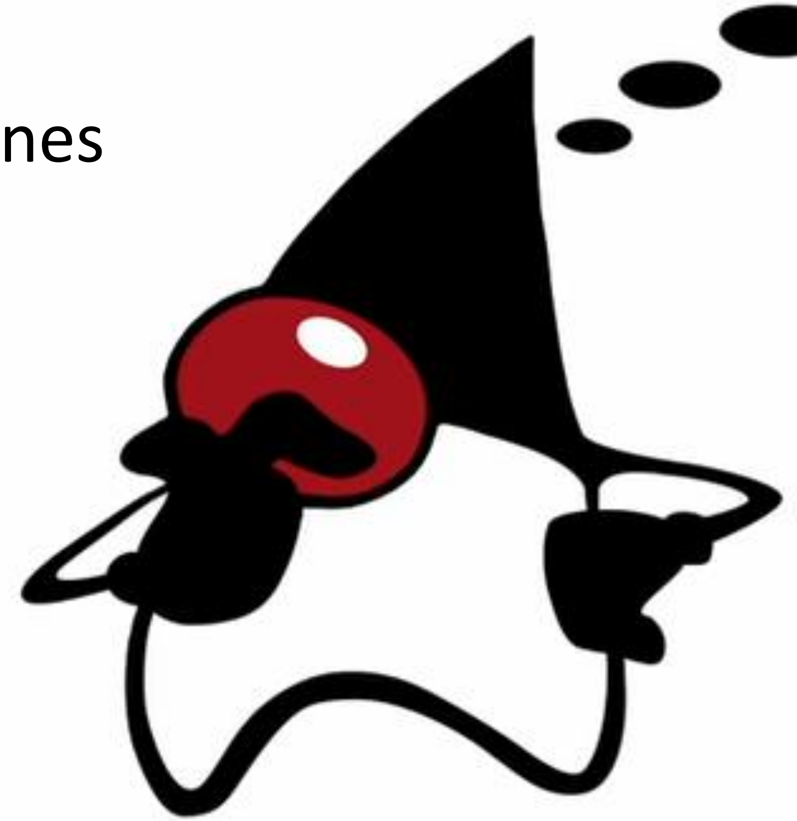
Stjepan Matijašević

dr. sc. **Martin Žagar**

HUJAK

Assessing the New Development Landscape

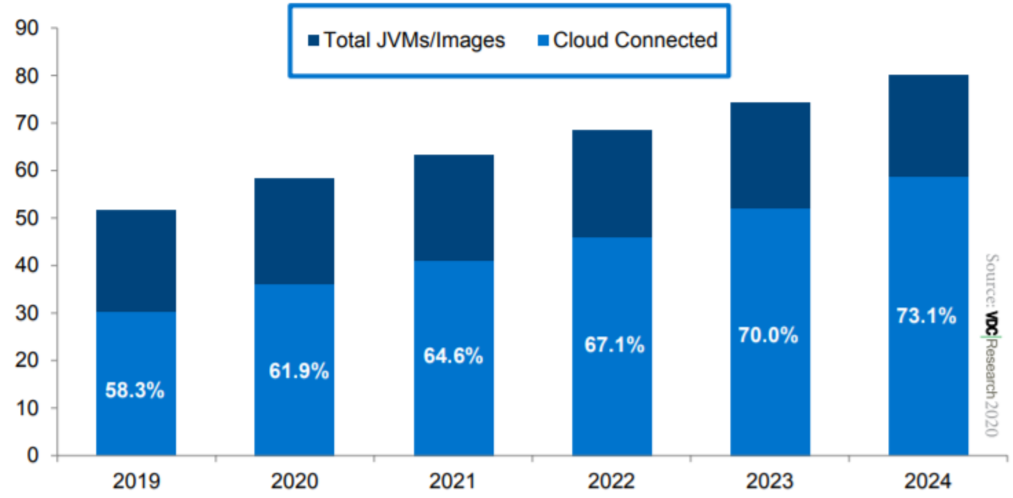
- New programming **languages**, programming **polyglotism & interoperability**
- New software development **paradigms**
- New **frameworks** or new (different) versions of old ones
- Changing hardware and software **architectures**
- Modern application **solutions**
- Variety of **deployment models**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



Java **Facts**

- **#1 Development Platform**
  - **Continued growth** for **27+ years**
- **#1 Programming Language**
- **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

Why do we (still) use Java and JVM?

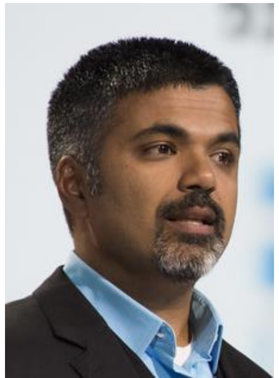
- **Openness** and **Platform Independence**
- Community **Acceptance** and **Familiarity**
- Variety of **Tools, Libraries** and **Frameworks**
- **Reliability** and **Trust**
- **Continuous Innovation** and **Predictability**
- **Contribution** of the Entire **Community**

…

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

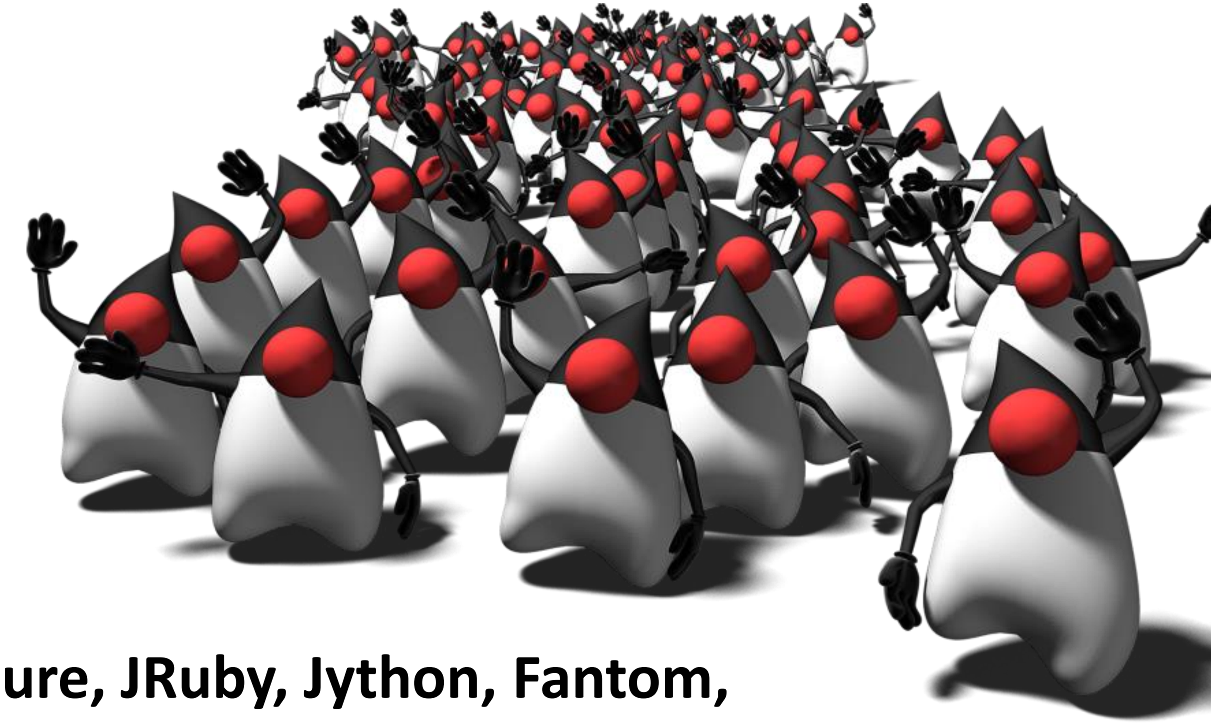
*According to an IDC report over **10 million developers**, representing **75% of full-time developers** worldwide, **use Java**, more than any other language.*

Sharat Chander, September 2021



More Java Facts

- **10 Million Java Developers**
  - With many **Java Certificates**
- **98% Fortune 100** companies hiring **Java Developers**
- **69%** of **Software Developers** run (some kind of…) Java

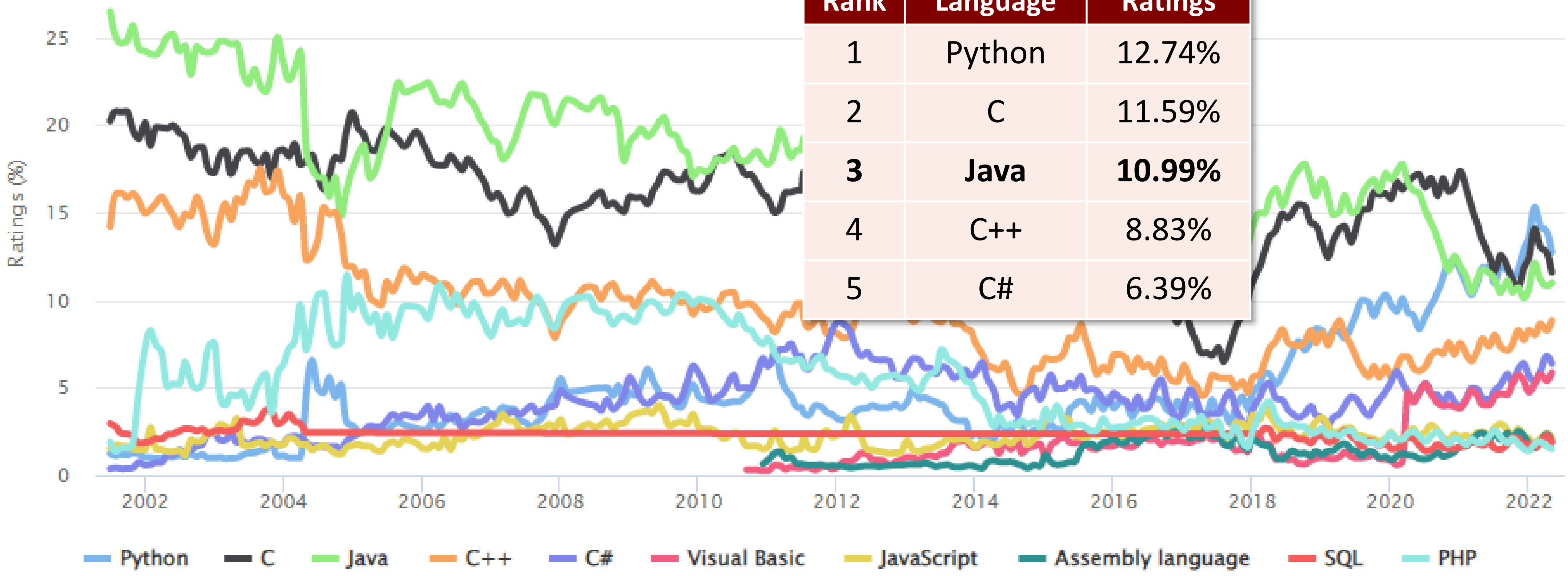


- **50+ JVM languages**
  - JVM languages : **Groovy, Kotlin, Scala, Clojure, JRuby, Jython, Fantom, Ceylon, Xtend, X10, LuaJ, Golo, Frege, Mirah, Eta**, **JavaScript…**
- And **all other languages** with **GraalVM (+ Truffle + Sulong)**

Is Java still popular?

- **TIOBE index** for May 2022

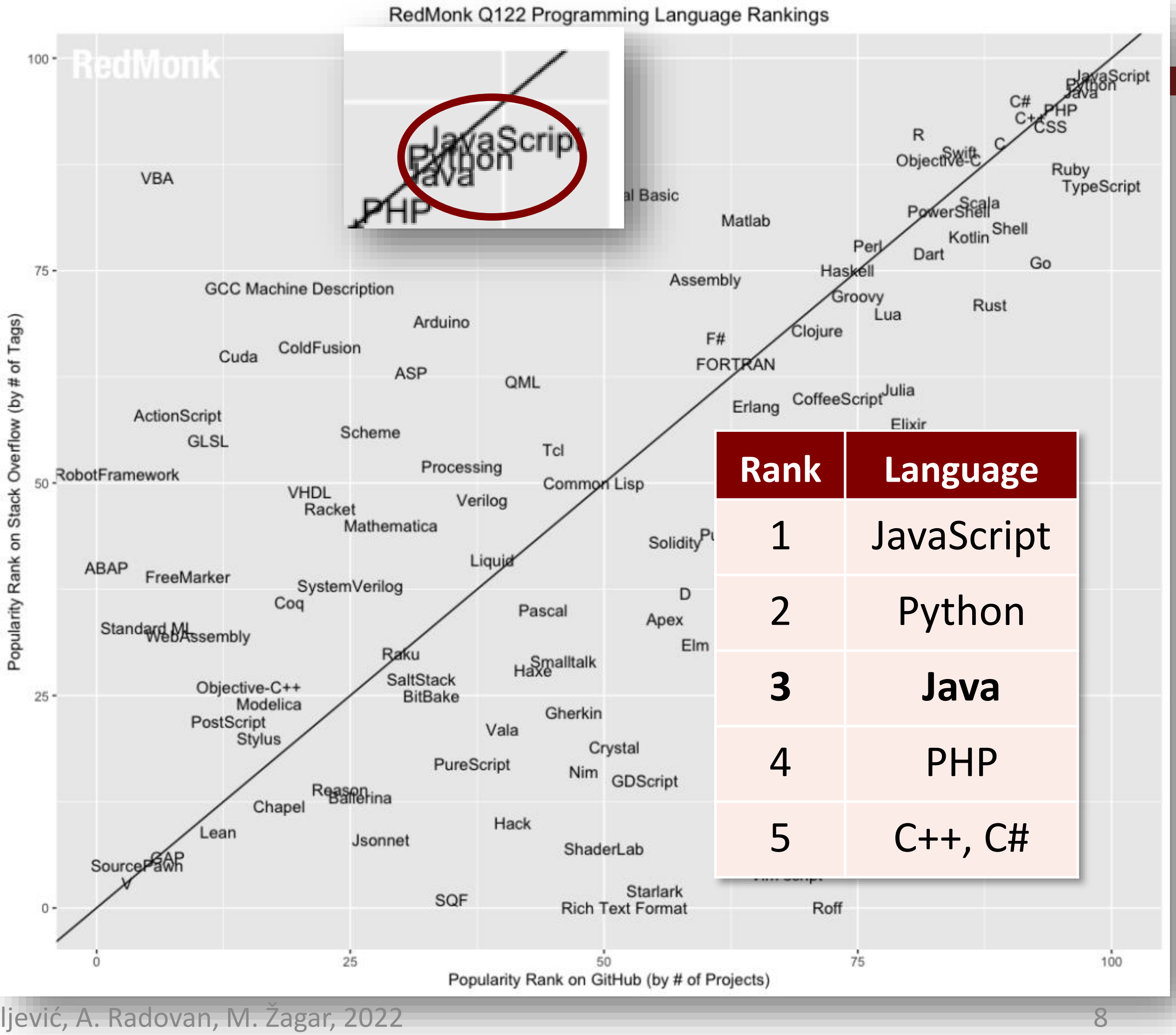
Rank	Language	Ratings
1	Python	12.74%
2	C	11.59%
3	**Java**	**10.99%**
4	C++	8.83%
5	C#	6.39%



Source: TIOBE Index for May 2022, www.tiobe.com/tiobe-index/

Is Java still popular? #2

- **RedMonk Programming Language Rankings**: January 2022
  - Extraction of language rankings from **GitHub** and **Stack Overflow**
  - Combining them to reflect both code (GitHub) and discussion (Stack Overflow) traction



Rank	Language
1	JavaScript
2	Python
3	**Java**
4	PHP
5	C++, C#

Source: The RedMonk Programming Language Rankings: January 2022, https://redmonk.com/sogrady/2022/03/28/language-rank

Available JDKs?

- Oracle JDK or one of many OpenJDKs

- 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 **18**.0.1.1, Java SE **17**.0.3.1 (LTS), Java SE **11**.0.15.1 (LTS) or Java SE **8**u333 (LTS)
  - License: NFTC (17 and after) or OTN (before 17) – to be discussed later
- **Oracle OpenJDK** jdk.java.net
  - Java SE **18**.0.1.1, Java SE **17** (LTS), or any other between **16** and **7**
  - License: GPLv2+CE
- **~~AdoptOpenJDK~~ Adoptium Eclipse Temurin** adoptium.net
  - Java SE **18**.0.1+10, Java SE **17**.0.3+7 (LTS), Java SE **16**.0.2+7, Java SE **11**.0.15+10 (LTS) or Java SE **8**u332 (LTS)
  - License: Apache Version 2.0 and GPLv2+CE
- **Azul's Zulu OpenJDK** www.azul.com/downloads/?package=jdk
  - Java SE **18**.0.1+10, Java SE **17**.0.3+7 (LTS), and all others back to Open JDK **6**

Is Java **Moving Forward**?

Predictability

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

Trust

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

Innovation

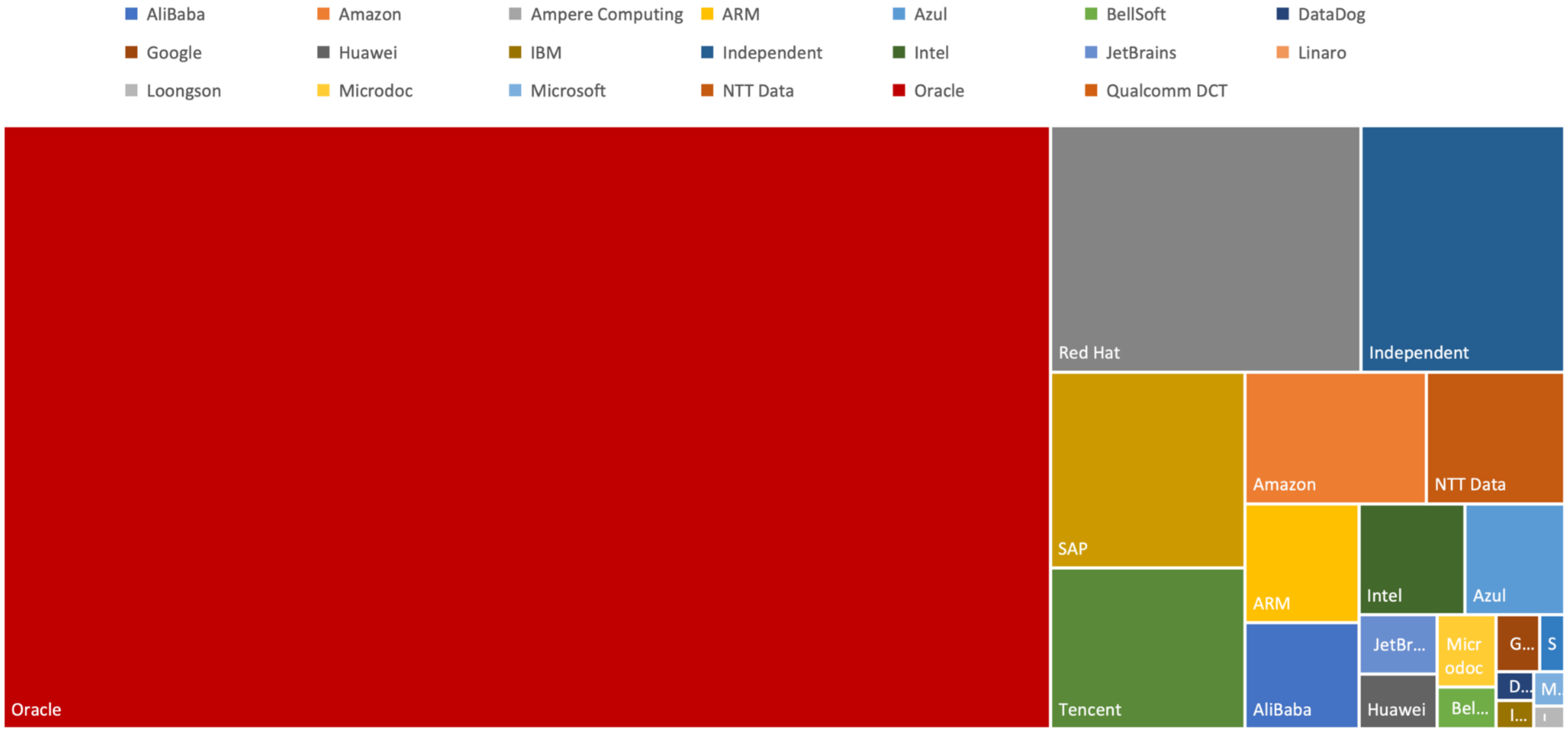
- Respecting contemporary software development – **innovative improvements**
- Gradually introducing language and platform enhancements – **cautious innovation**

Innovation with Incubators and Preview

- **Incubator Features**
  - New API and tools that, after stabilization, are most likely to be included in JDKs
- **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
- **JDK development projects**
  - Amber, Valhalla, Panama, Loom, Leyden, ZGC and many others

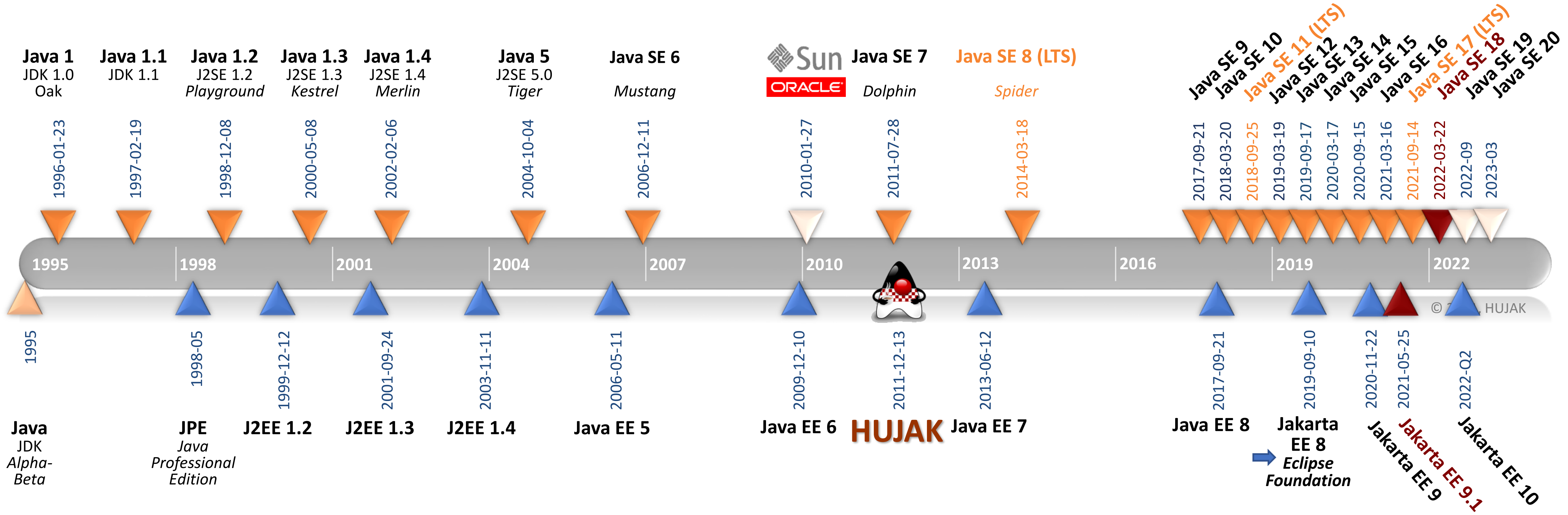
Creating Java Together

- Issues fixed in JDK 11-17 per **organization**
  - Led by **Oracle**, but significant contributions by many others: **Red Hat, SAP, Tencent, Amazon, NTT Data, ARM, Intel, Azul, Alibaba, JetBrains, Microdoc, Huawei** …



Java **Timeline**

- 27+ years of history…



Java 1
JDK 1.0
Oak
1996-01-23

Java 1.1
JDK 1.1
1997-02-19

Java 1.2
J2SE 1.2
Playground
1998-12-08

Java 1.3
J2SE 1.3
Kestrel
2000-05-08

Java 1.4
J2SE 1.4
Merlin
2002-02-06

Java 5
J2SE 5.0
Tiger
2004-10-04

Java SE 6
Mustang
2006-12-11

Sun
ORACLE
2010-01-27

Java SE 7
Dolphin
2011-07-28

Java SE 8 (LTS)
Spider
2014-03-18

Java SE 9 — 2017-09-21
Java SE 10 — 2018-03-20
Java SE 11 (LTS) — 2018-09-25
Java SE 12 — 2019-03-19
Java SE 13 — 2019-09-17
Java SE 14 — 2020-03-17
Java SE 15 — 2020-09-15
Java SE 16 — 2021-03-16
Java SE 17 (LTS) — 2021-09-14
Java SE 18 — 2022-03-22
Java SE 19 — 2022-09
Java SE 20 — 2023-03

1995
Java
JDK
Alpha-Beta

1998-05
JPE
Java Professional Edition

1999-12-12
J2EE 1.2

2001-09-24
J2EE 1.3

2003-11-11
J2EE 1.4

2006-05-11
Java EE 5

2009-12-10
Java EE 6

2011-12-13
HUJAK

2013-06-12
Java EE 7

2017-09-21
Java EE 8

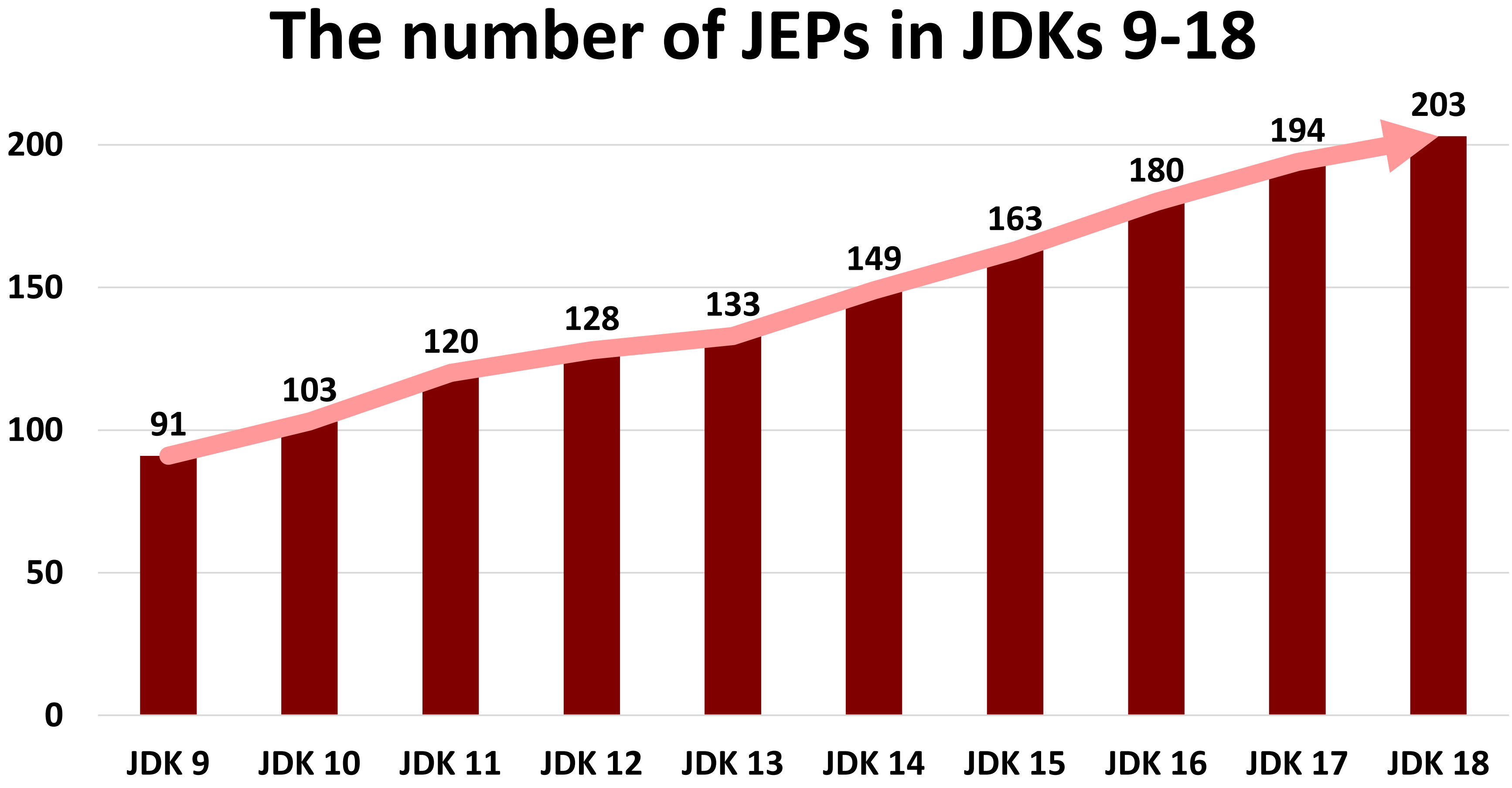
2019-09-10
Jakarta EE 8
➡ ***Eclipse Foundation***

2020-11-22
Jakarta EE 9

2021-05-25
Jakarta EE 9.1

2022-Q2
Jakarta EE 10

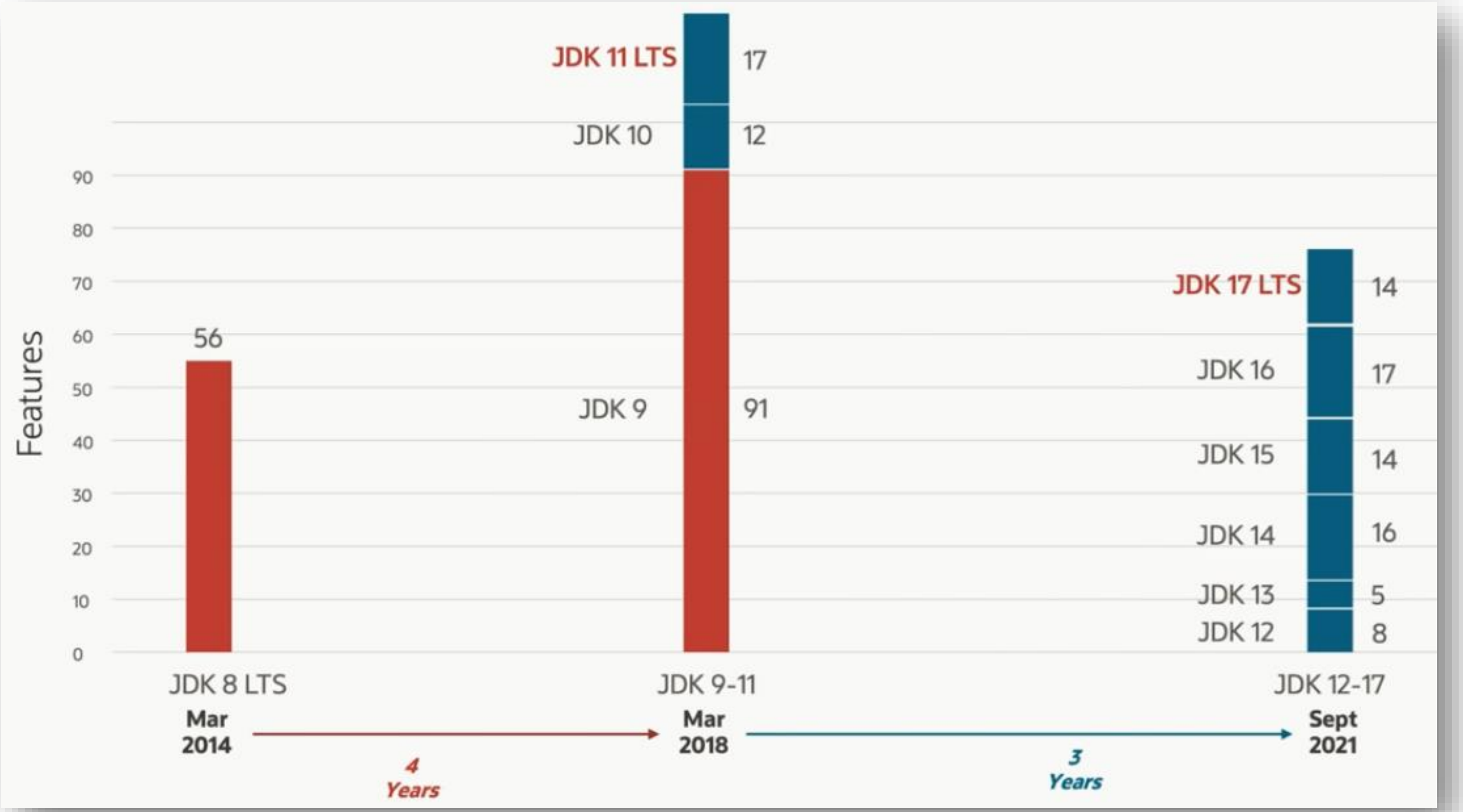
Constant Evolution through JEPs



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

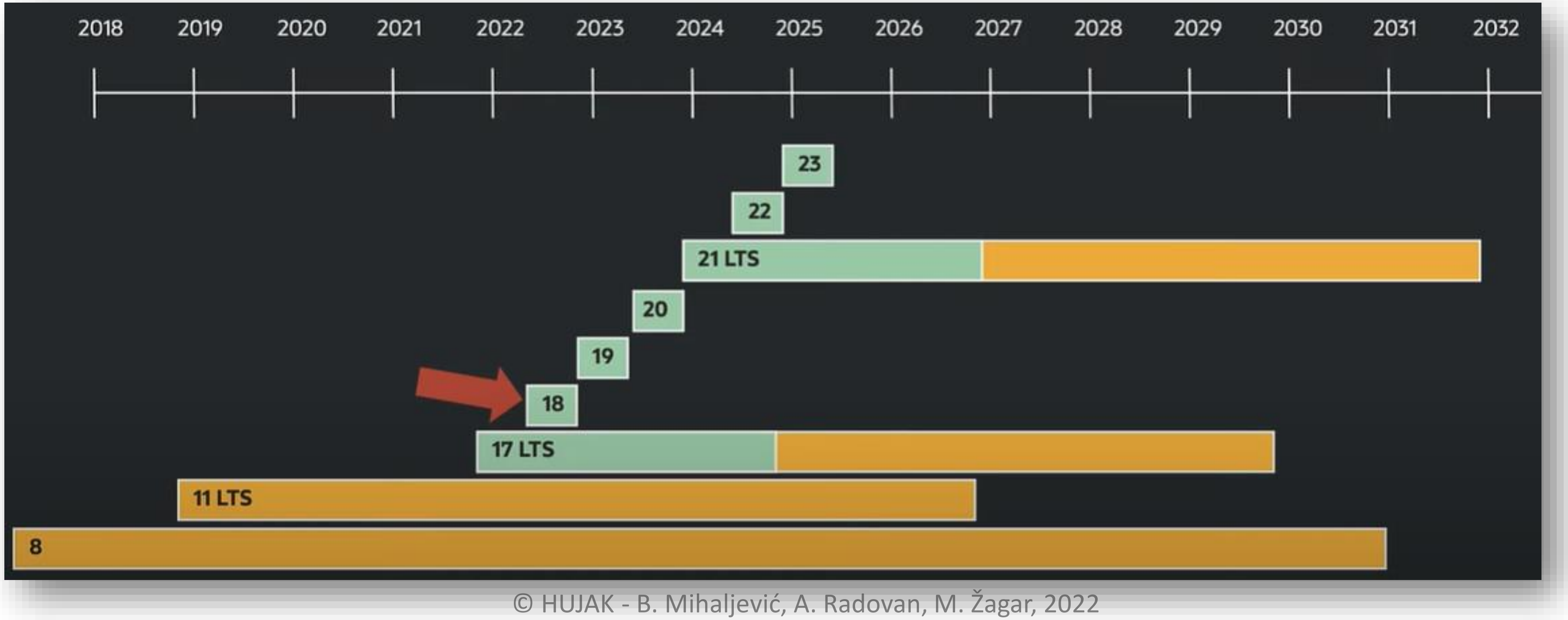
LTS Releases Include Many JEPs

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



LTS Release – Every 2 Years NEW!

- Demand for LTS (Long Term Support) has grown
  - Surveys show 6-month releases not being used often in production
- Proposing the **new LTS release** schedule –> **every 2 years** (instead of 3)



Is Java Really "Free"?

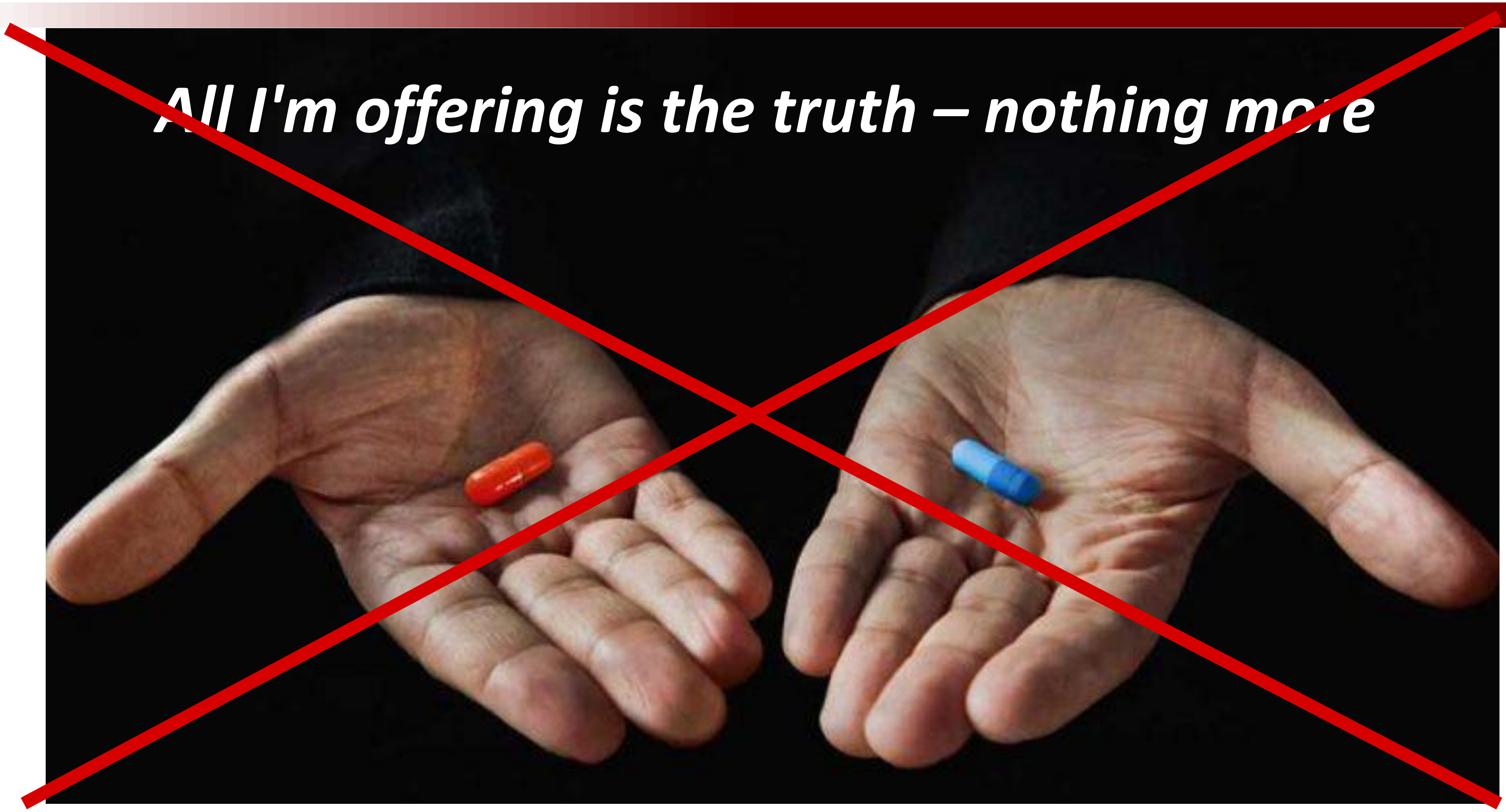
- Use of **OpenJDK** for **free** with **GPLv2+CE license**
- **Updates** (code patches) – typically **free of charge**
- **Support** (fixing bugs and answering questions) – it was **never free of charge**

What about **Oracle JDK?**

- **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 **Oracle JDK 17+?**

OpenJDK or Commercial JDK?

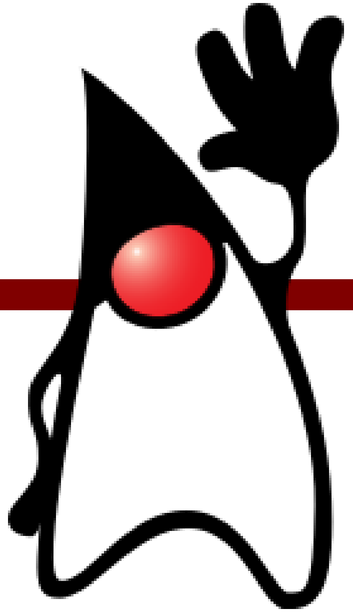


~~*All I'm offering is the truth – nothing more*~~

Java is (Finally Completely) Free!

- **Oracle JDK 17 and later** with **Oracle No-Fee Terms and Conditions (NFTC)** licensing
  - Oracle JDK permits **free use** for all users, even commercial and production use
  - www.oracle.com/downloads/licenses/no-fee-license.html
- Oracle JDK free releases and updates will be provided for at least **one full year** after the **next** LTS release
  - Prior versions are not affected by this change
- Oracle **OpenJDK** releases will continue to be provided under GPL
  - On the same releases and schedule as it has since Java 9

Current State of Java?



Some questions for all of us:

- Still using **Java 8** (2014)?
- Switched to the **old LTS** version **Java 11** (2018)?
- Upgraded to 6-month release of **Java 12-16** or **18**?
- Upgrading to the **latest LTS** version **Java 17** (2021)?
- What about ~~Java~~ **Jakarta EE** (9.1)?

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 (now 2) **years**
  - Updates available for at least 3 years

JDK 10 – was new, but 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

JDK 11 – the "older" 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 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

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)
- Others …

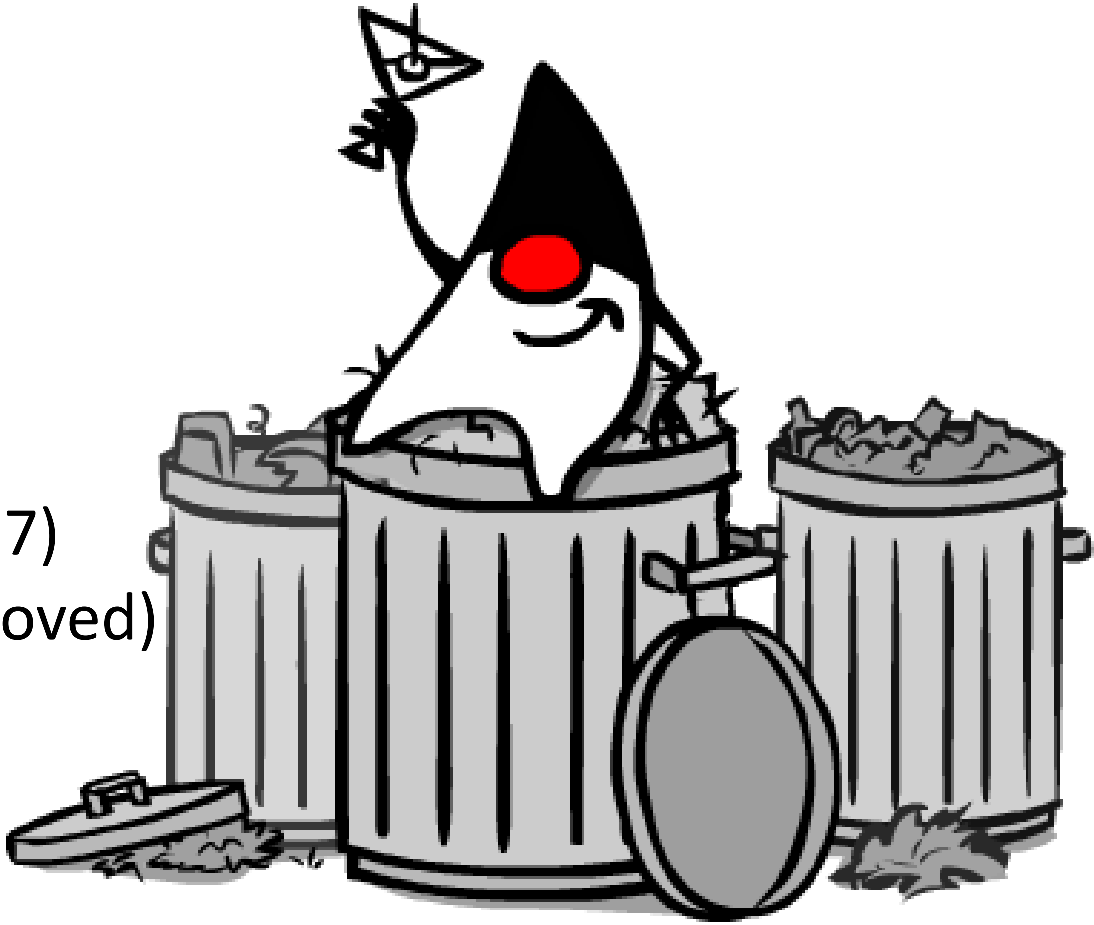


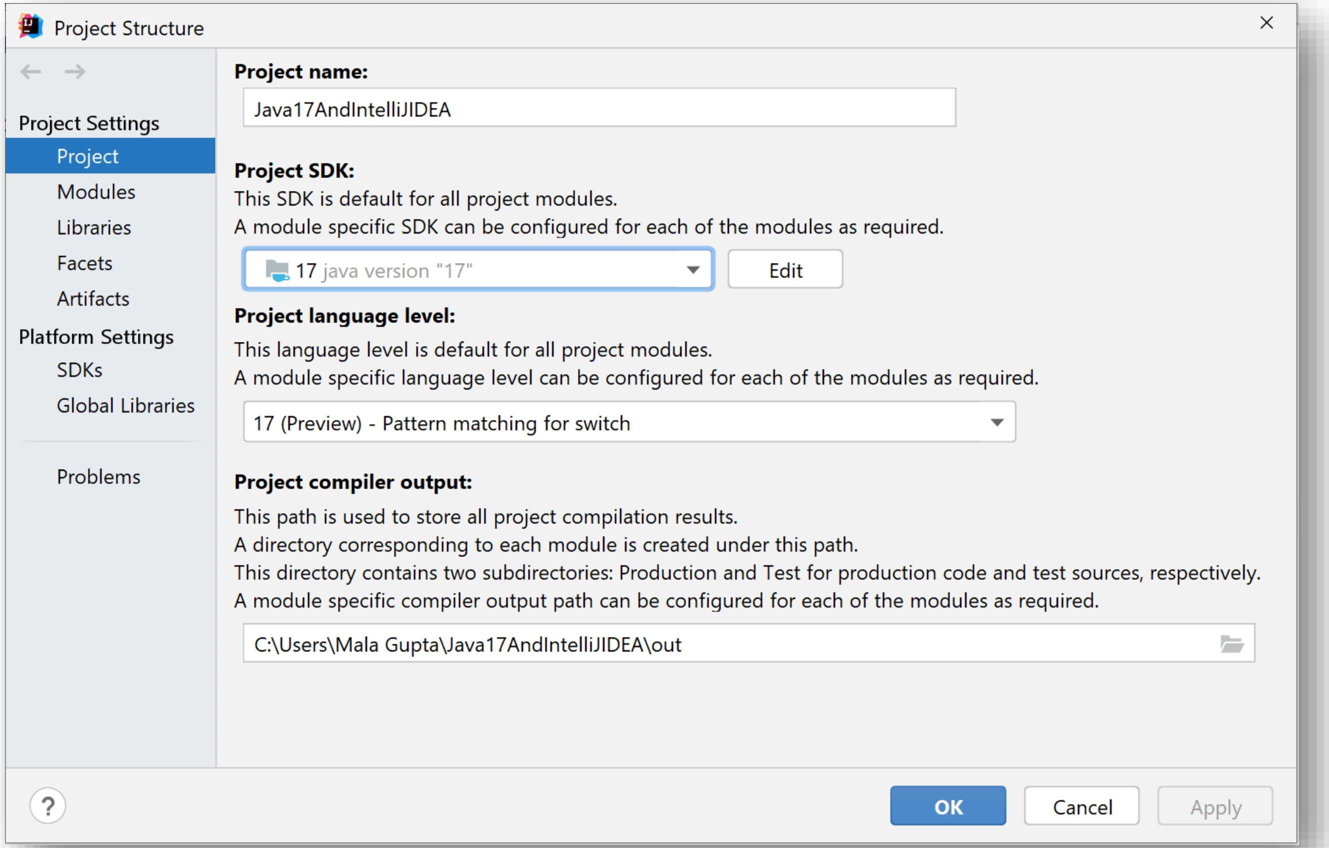
Image Source: 5 Coding Hacks to Reduce GC Overhead, Tall Weiss, OverOps, 2013

JDK 12 – with the first "Previews"

- **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`



Project name:

Java17AndIntelliJIDEA

Project SDK:

This SDK is default for all project modules.

A module specific SDK can be configured for each of the modules as required.

17 java version "17"

Project language level:

This language level is default for all project modules.

A module specific language level can be configured for each of the modules as required.

17 (Preview) - Pattern matching for switch

Project compiler output:

This path is used to store all project compilation results.

A directory corresponding to each module is created under this path.

This directory contains two subdirectories: Production and Test for production code and test sources, respectively.

A module specific compiler output path can be configured for each of the modules as required.

C:\Users\Mala Gupta\Java17AndIntelliJIDEA\out

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**: Uncommitted 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)

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 and Hidden Classes

- **Sealing** allows classes and interfaces to define their permitted subtypes
  - Enabling more fine-grained inheritance control in Java.
- *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
- **Hidden** classes cannot be used directly by the bytecode or other classes
  - A standard way to generate dynamic classes

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 { }
```

Sometimes data is just … data.

Mark Reinhold

```
    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 …
}
```

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 (LTS) 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

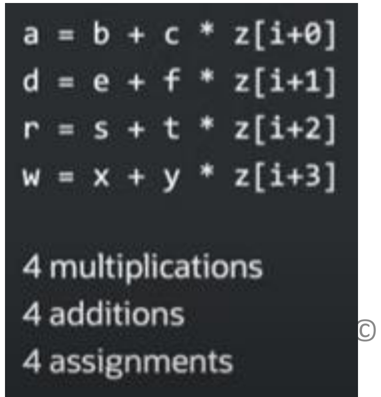
- JEP 412: **Foreign Function & Memory API** (Incubator) openjdk.java.net/jeps/412
- Introducing API to of statically-typed, pure-Java access to **native code**
- Simplifying error-prone process of **binding** to a native library
- Use any native library
- Java Native Interface (JNI) was a bit hard an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

JDK 18 – Foreseeable Future

- **JDK 18 in March 2022**
  - **New features** and **APIs** at https://openjdk.java.net/projects/jdk/18/
- **9 JEPs** targeted:
  - 400: **UTF-8** by Default
  - 408: **Simple Web Server**
  - 413: Code Snippets in Java API Documentation
  - 416: Reimplement Core Reflection with Method Handles
  - 417: **Vector API** (Third Incubator)
  - 418: Internet-Address Resolution SPI
  - 419: Foreign Function & Memory API (Second Incubator)
  - 420: Pattern Matching for switch (Second Preview)
  - 421: Deprecate Finalization for Removal

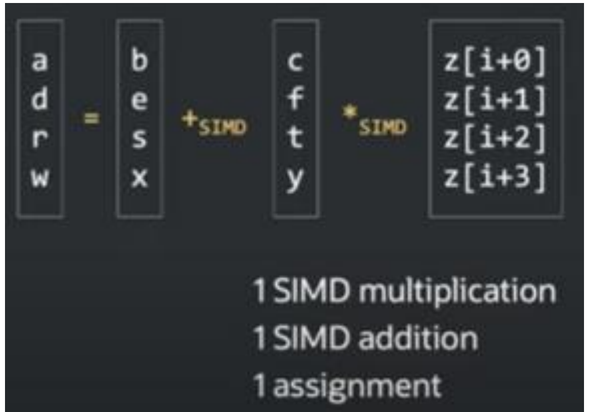
Vector API

- 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



1 SIMD multiplication
1 SIMD addition
1 assignment

JDK 19 – Foreseeable Future

- **JDK 19 in September 2022**
  - **New features** and **APIs** at https://openjdk.java.net/projects/jdk/19/
- **JEPs** targeted (so far):
  - 422: Linux/RISC-V Port
  - 424: **Foreign Function & Memory API** (Preview)
  - 425: **Virtual Threads** (Preview)
  - 426: **Vector API** (Fourth Incubator)
  - 427: **Pattern Matching for switch** (Third Preview)
- **JEPs** proposed to target (so far):
  - 405: **Record Patterns** (Preview)

Virtual Threads and Record Patterns

- **Virtual Threads** (Preview) – **lightweight threads** that dramatically reduce the effort of writing, maintaining, and observing high-throughput concurrent applications
- **Record Patterns** (Preview) – enhance Java with *record patterns* to deconstruct record values
  - Record patterns and type patterns can be nested to enable a powerful, declarative, and composable form of data navigation and processing
  - Lifts the declaration of local variables for extracted components into the pattern
  - Initializes those variables by invoking the accessor methods when a value is matched against the pattern
  - In effect, a record pattern disaggregates an instance of a record into its components

Tooling Support for JDK 17/18

- 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 17/18 support today within:
  - **JetBrains IntellliJ IDEA 2022.1**
  - **Eclipse IDE 2022-03 (4.23)** via a marketplace solution
  - **NetBeans 12.8** with support for JDK 17
  - **Visual Studio Code**
  - …

AI and Code

- **AI-generated Source Code**
  - Predicts your next block of code delivering accurate code completions
  - Accelerates development by providing code guidance with patterns learned from millions of projects
  - Automates repetitive work and reduces the need for expensive and distracting code search
  - Improves code quality and consistency across your project
- **Github Copilot** (copilot.github.com)
  - Technical Preview – based on a natural language processing model called “Codex”
  - Works best with Java, Python, JavaScript, Ruby, and Go but supports 50+ other languages
  - Extensions for many modern IDEs
- **Tabnine** (former Codota) (www.tabnine.com)
  - 25 supported languages including Java, Python, Go, Dart, Julia, HCL, Ruby, Rust, C++
  - 21 supported IDEs/editors
- **Kite** (www.kite.com)
  - Supports Python and 15 other languages
  - 16 supported IDEs/editors

The Most Important Java Features (11-18)

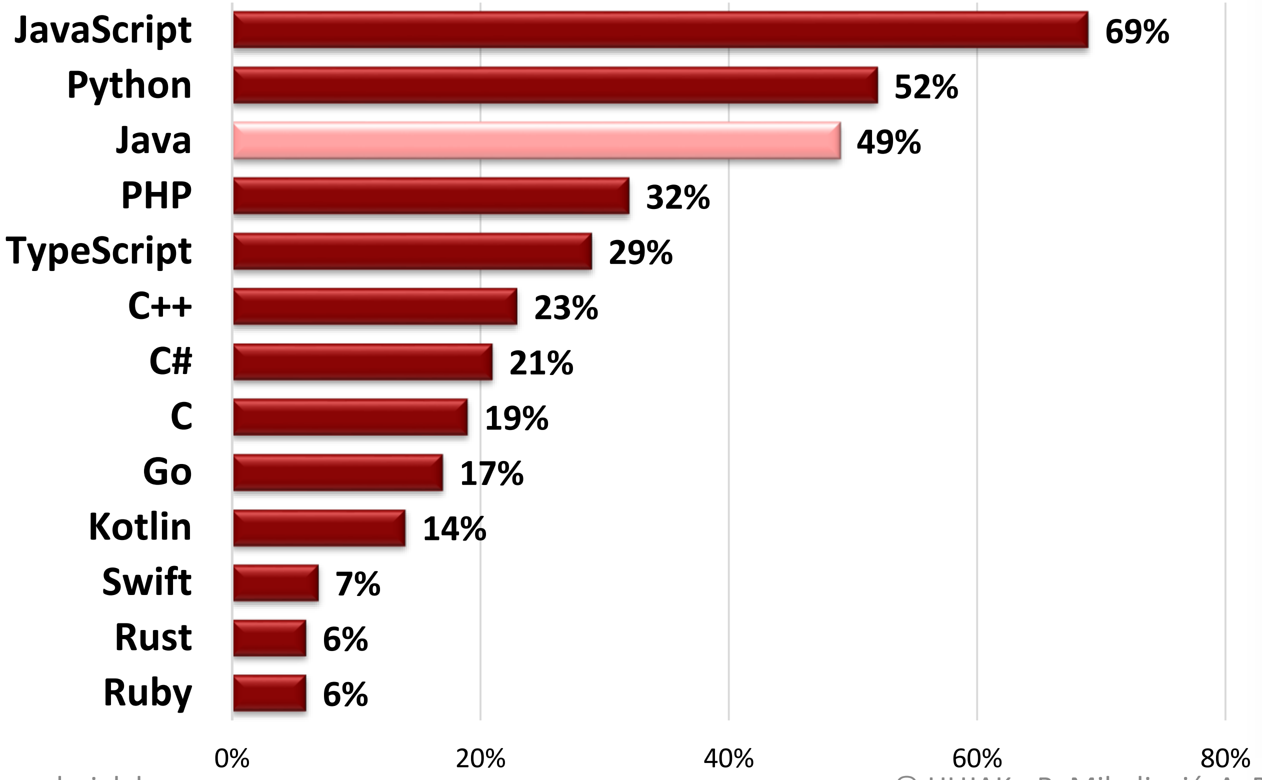
- **Languages Features** – Records, Switch Expressions, Text Blocks, vars, Pattern Matching for instanceof, Sealed Classes, Pattern Matching for switch…
- **Memory Management** – G1 GC enhancements (Full Parallel), ZGC, Shenandoah GC, Elastic Metaspace…
- **Library Enhancements** – Pseudo-Random Generator, Deserealization Filters, Vector API, Foreign Fuction & Memory API…
- **Future Proofing** – Module System (JPMS), Strong Encapsulation for JDK Internals…
- **Easier Debugging** – Flight Recorder, JFT Event Streaming, NullPointerExceptions…
- **Modernizing Infrastructure** – ~~Mercurial~~→Git, GitHub, AArch64 port…
- **Deprecations & Removals** – ~~CMS GC, Nashorn, Biased Locking, RMI Activation, Applet API, Security Manager~~…

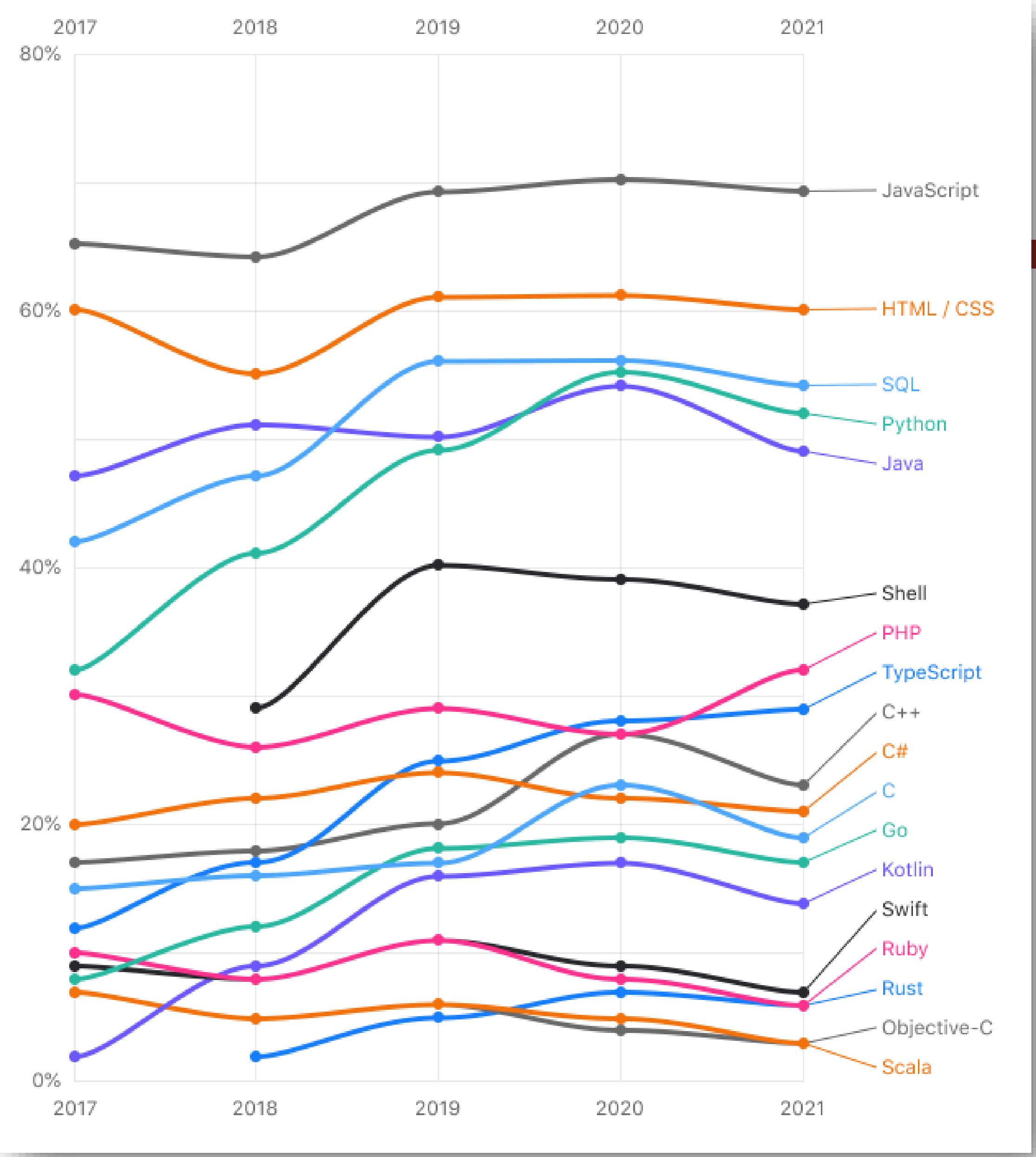
Some surveys

- **The State of Developer Ecosystem 2021** by **JetBrains**
  - 31743 developers from 183 countries in 2021
- **2021 Developer Survey** by **Stack Overflow**
  - 83000+ developers in 2021
- **2022 Java Developer Productivity Report** by **JRebel**
  - 876 developers in 2022
- **JVM Ecosystem Report 2021** by **Snyk**
  - 2000 Java developers in 2021
- **2021 Jakarta EE Developer Survey** by **Eclipse Foundation**
  - 940 developers in 2021
- **Java InfoQ Trends Report – December 2021**
- **2022 State of the Java Ecosystem** by **New Relic**

Top Languages

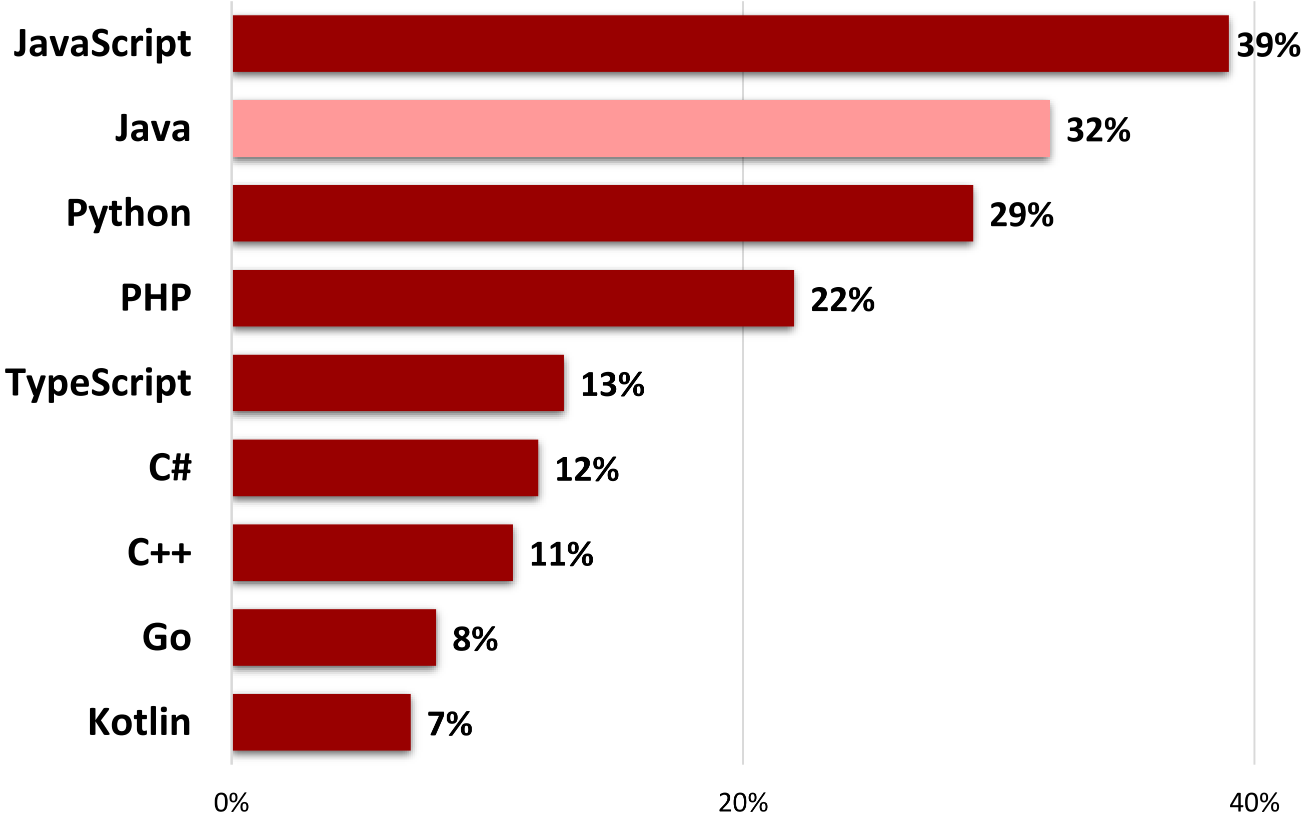
- What programming languages have you used (JetBrains)?



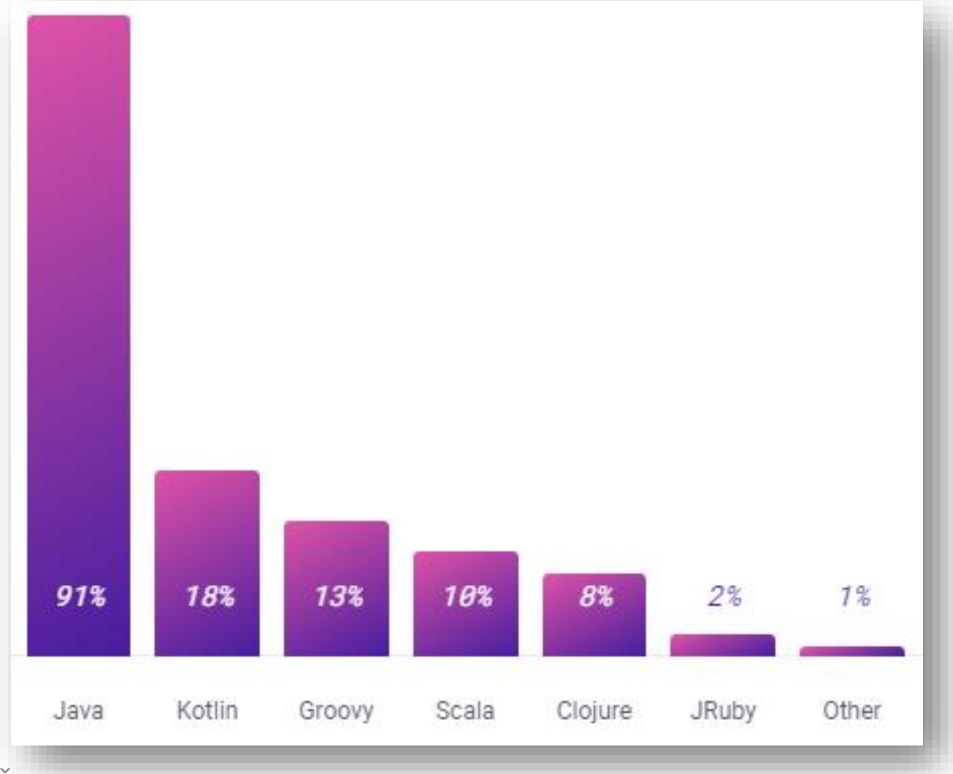


Main Language

- What is your **primary programming language** (JetBrains)?

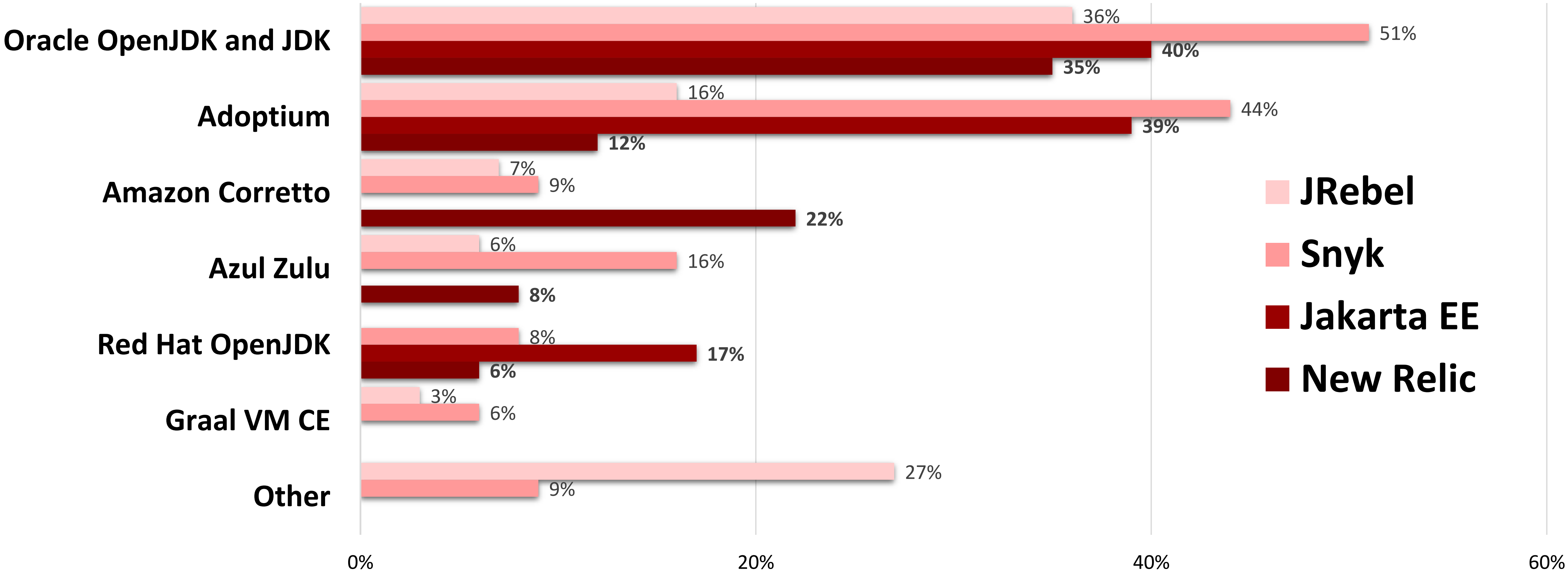


- **JVM languages**? (Snyk)



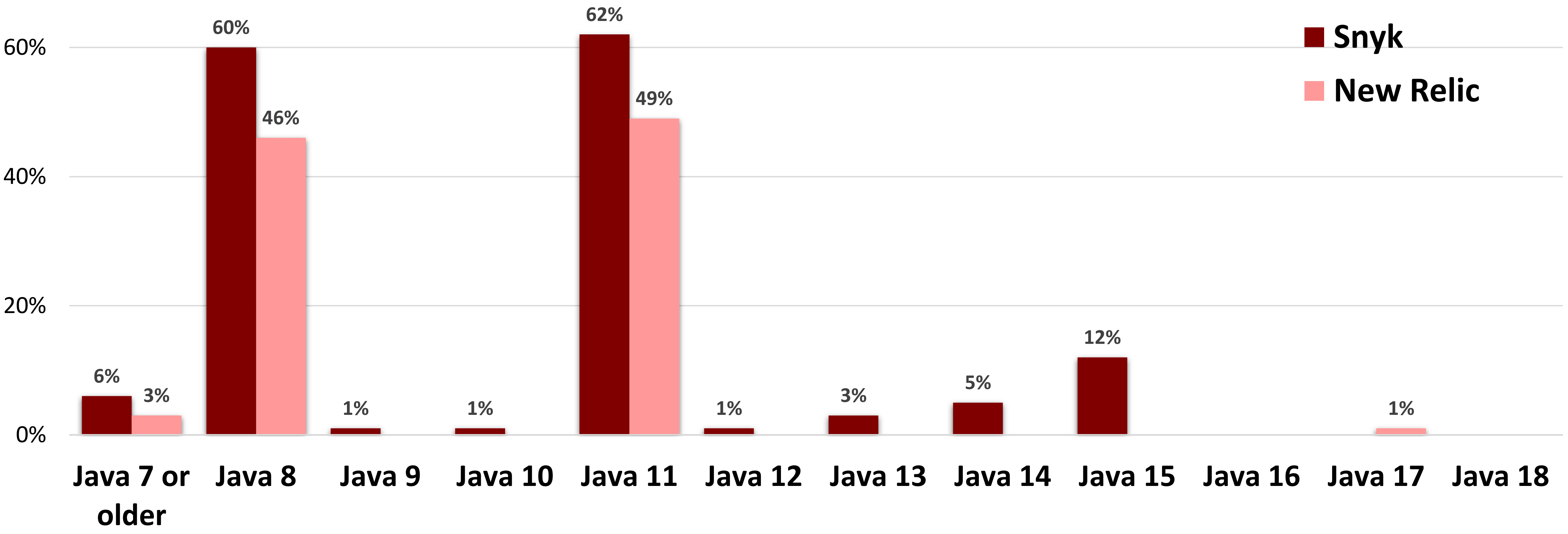
JDK Distributions

- Which **JRE/JDK distribution** do you use?



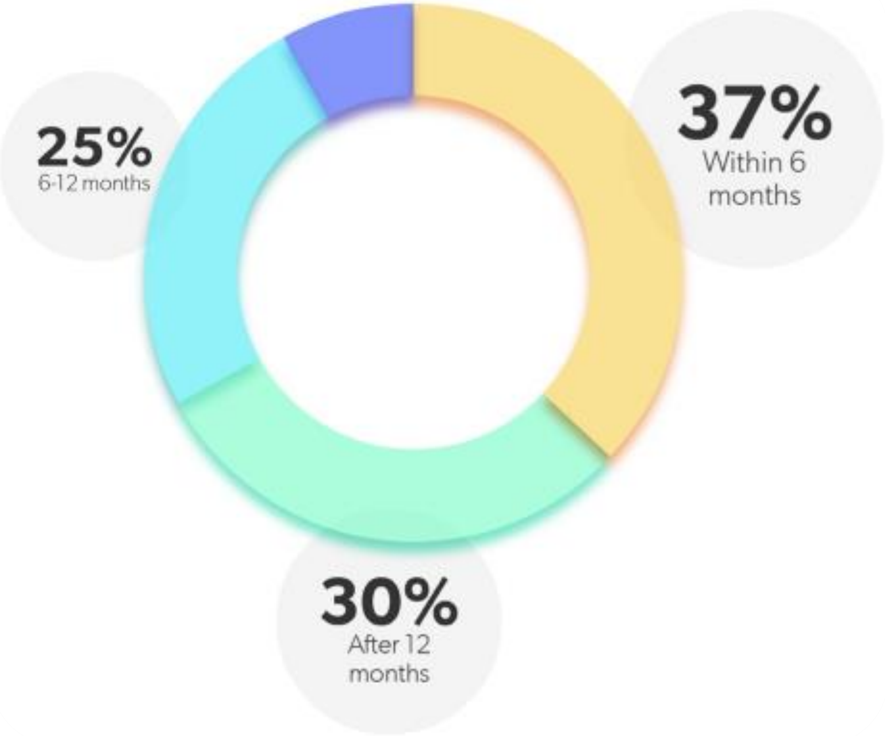
Versions of Java

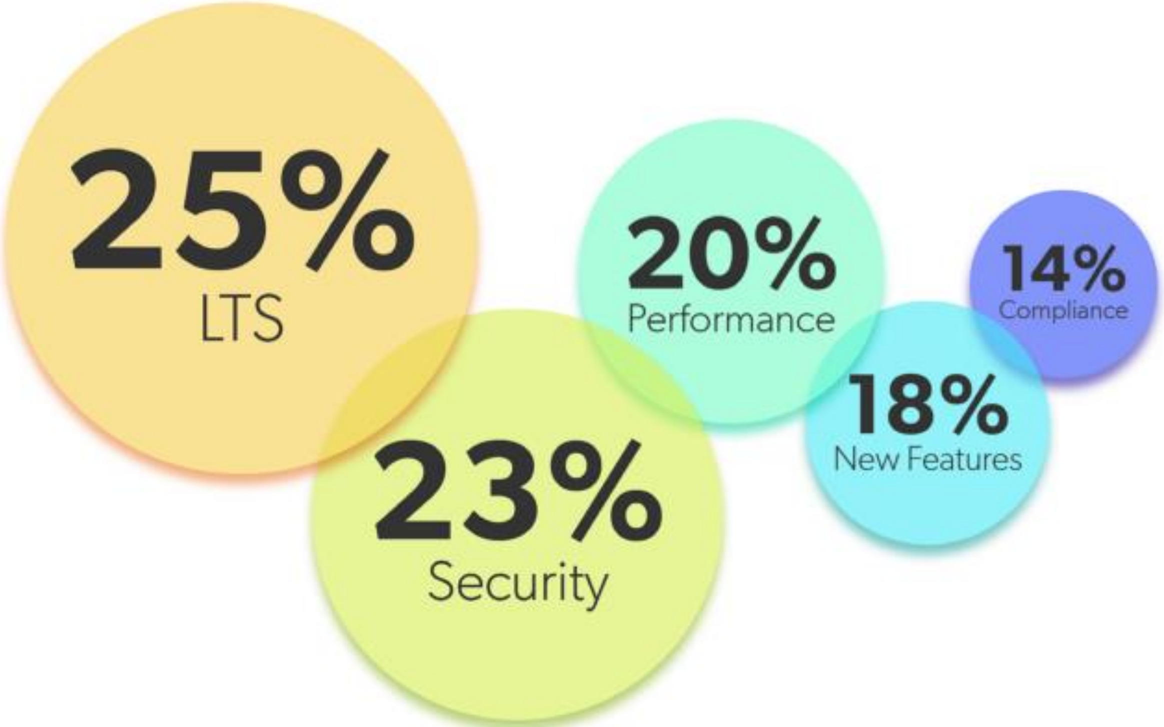
- **Java platform versions** used in projects (in production)



Upgrade to JDK 17

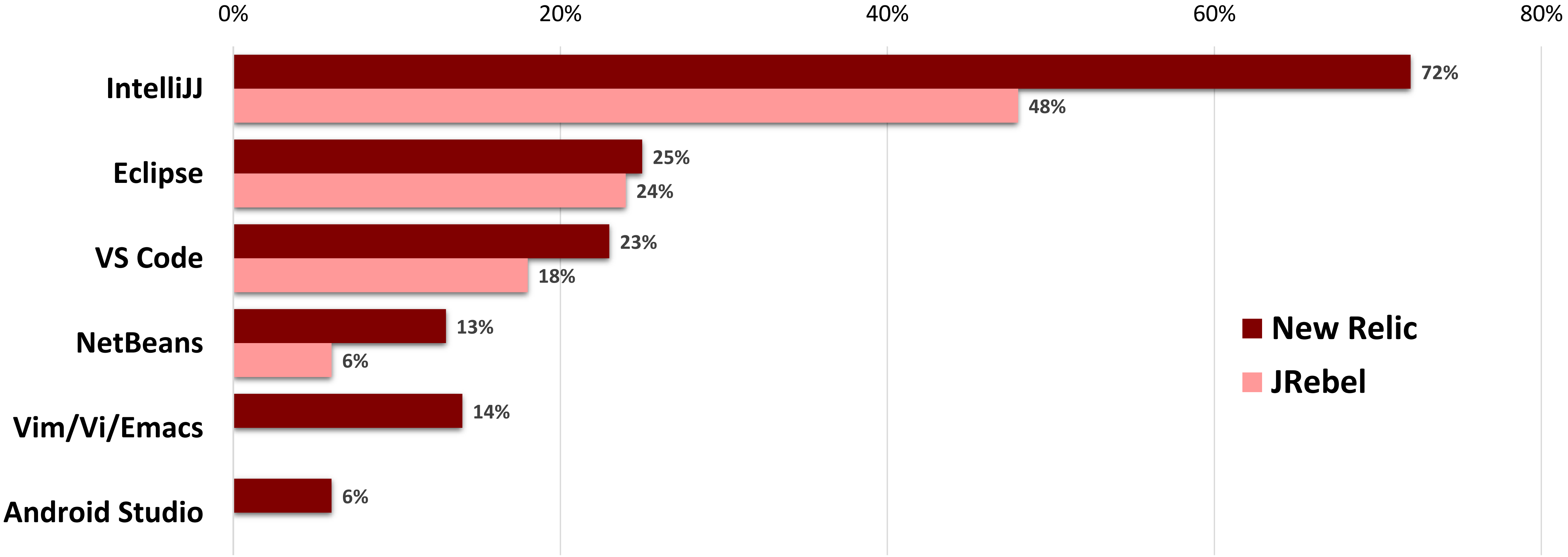
- Which **factors** influence your decision to upgrade JDK Versions? (JRebel)
- When will you upgrade to **JDK 17**? (JRebel)





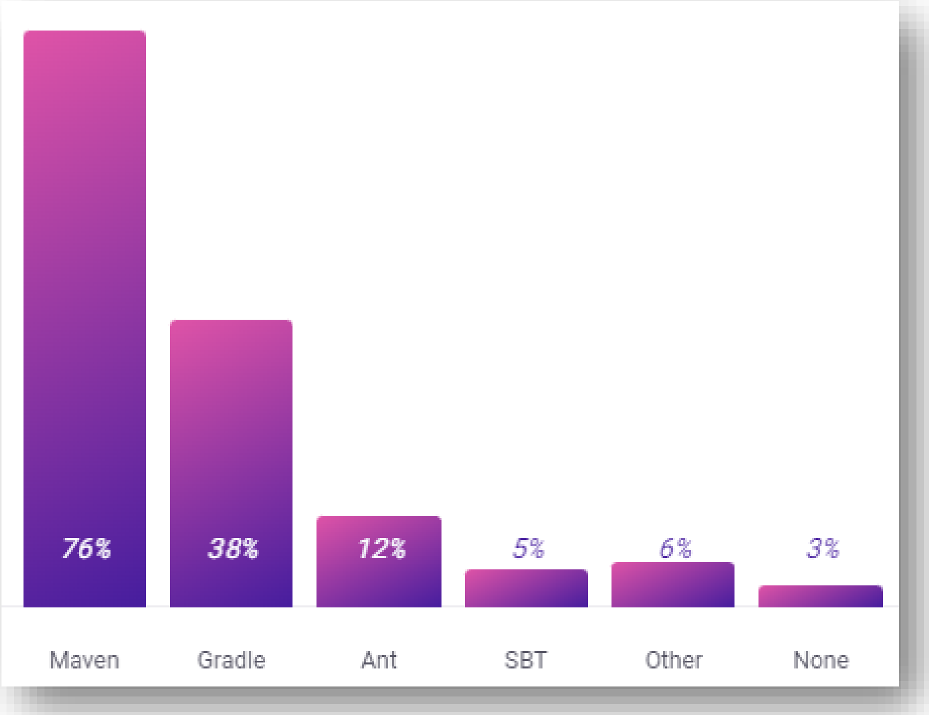
Most popular IDEs

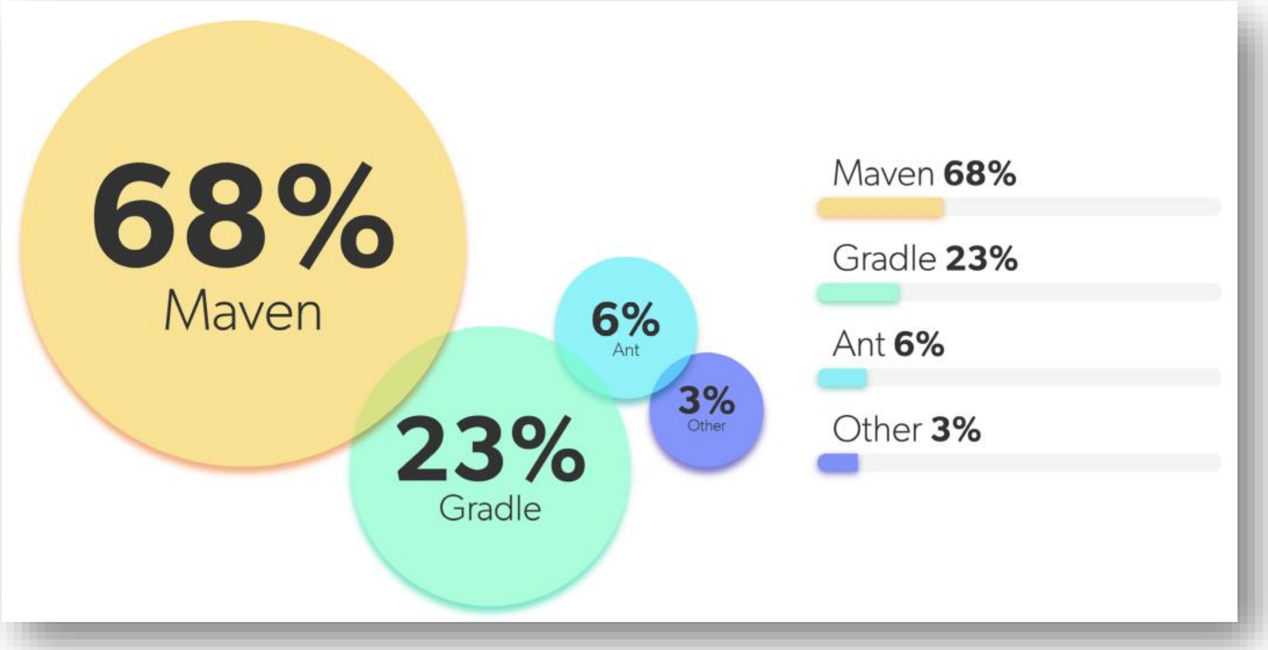
- The most **popular IDEs** used?



Build Tools

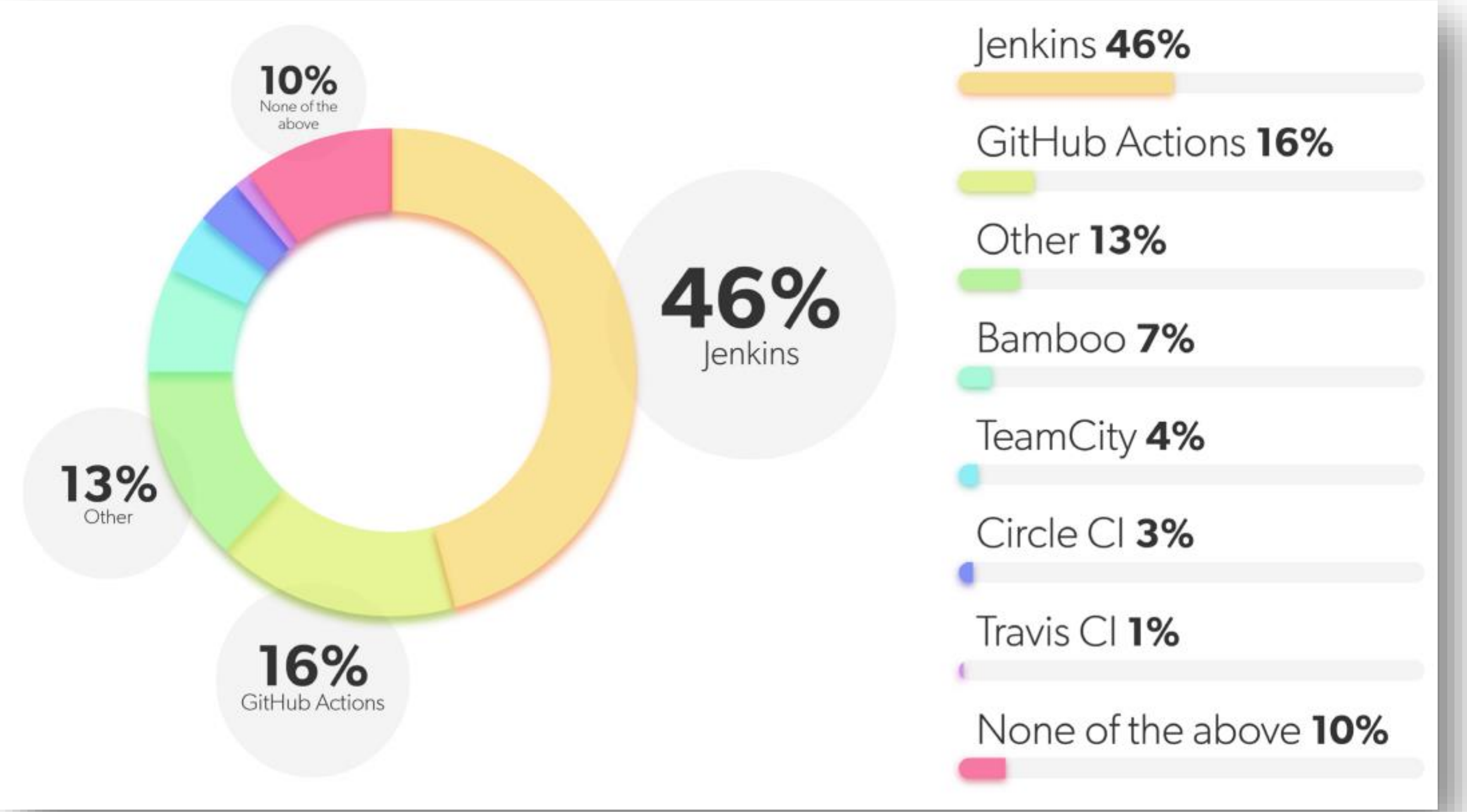
- **Tools for building** applications? (Snyk)
  - What **build tool** do you use in your main application? (JRebel)





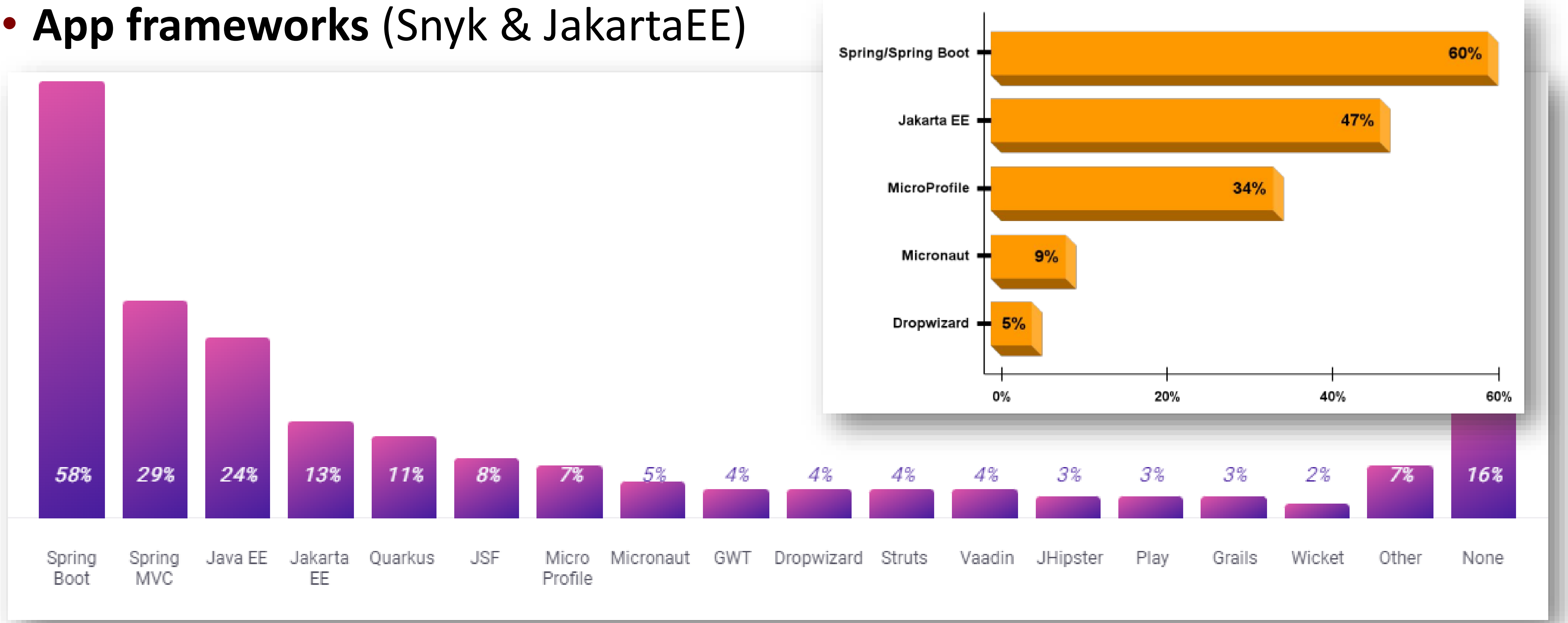
CI/CD

- Which **CI/CD technologies** are you using? (JRebel)



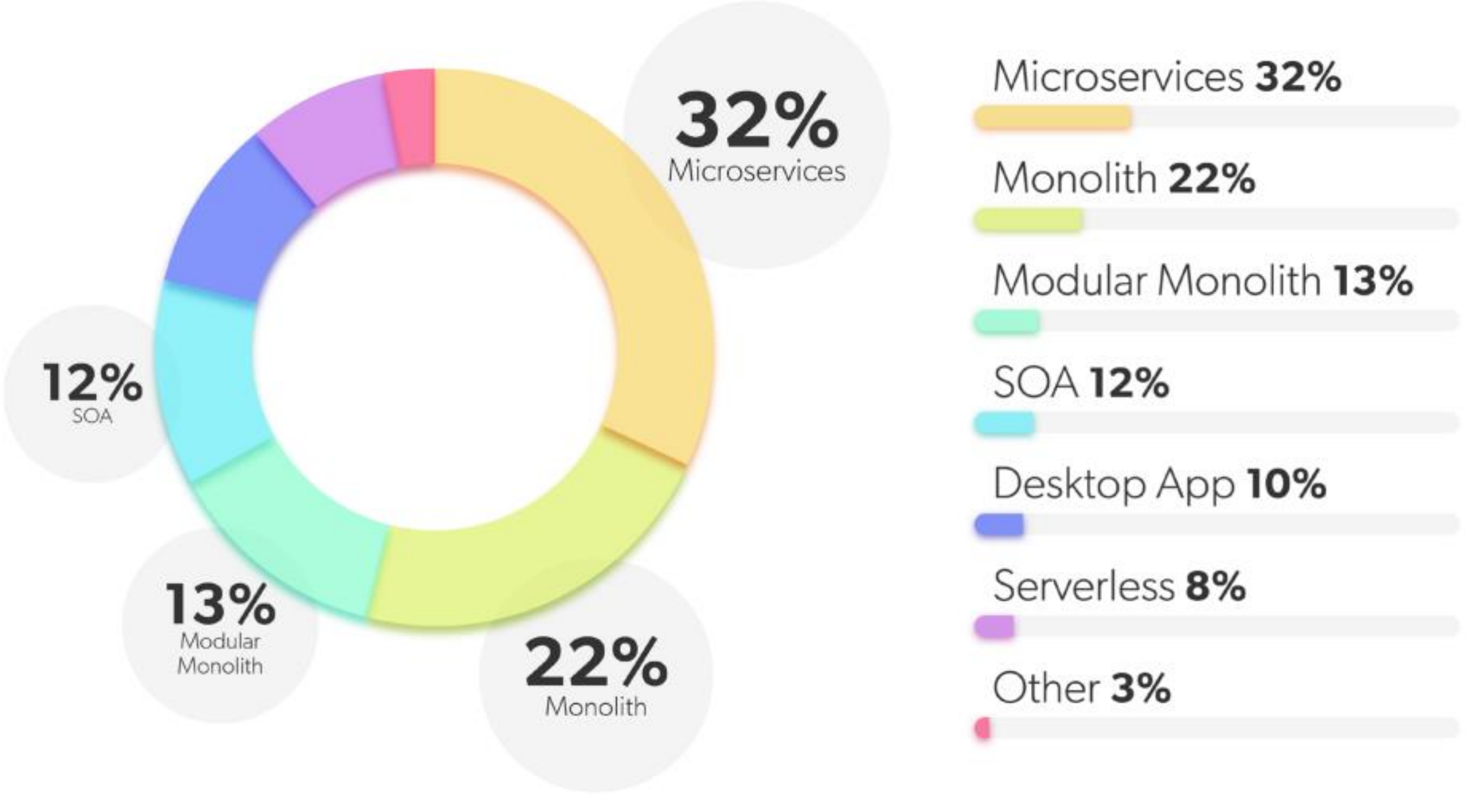
Application frameworks

- **App frameworks** (Snyk & JakartaEE)



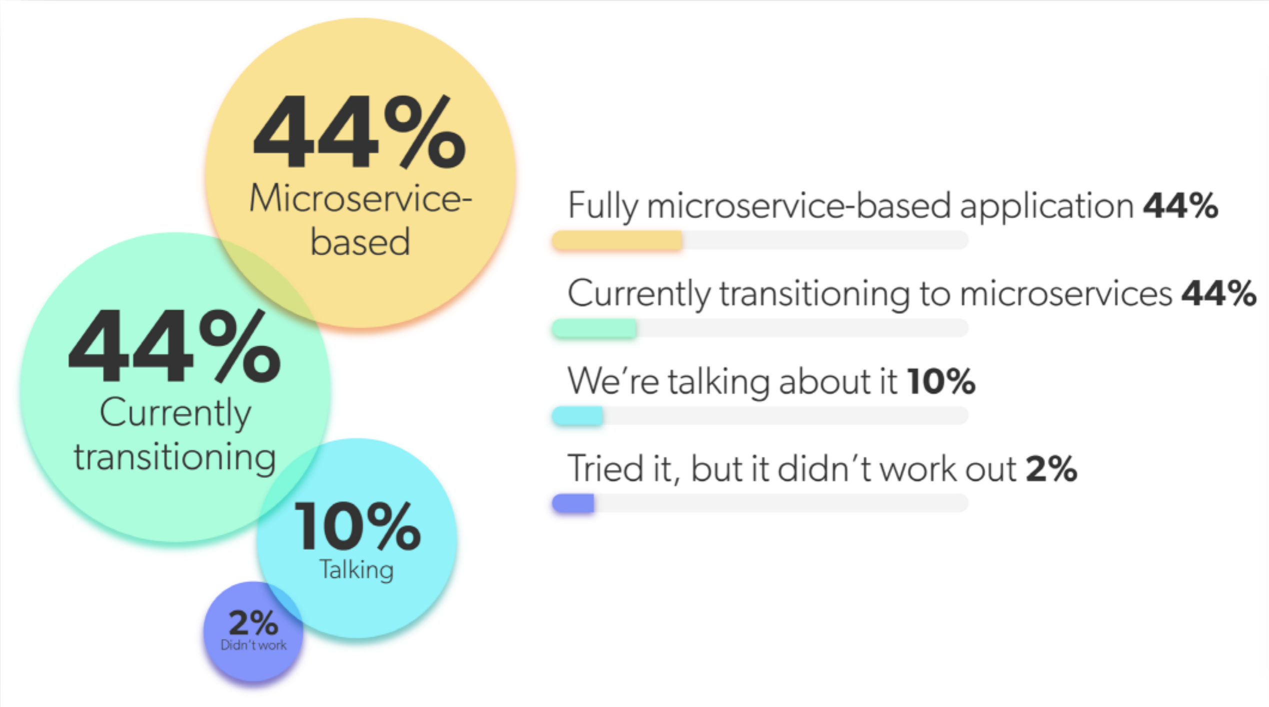
Architecture Trends

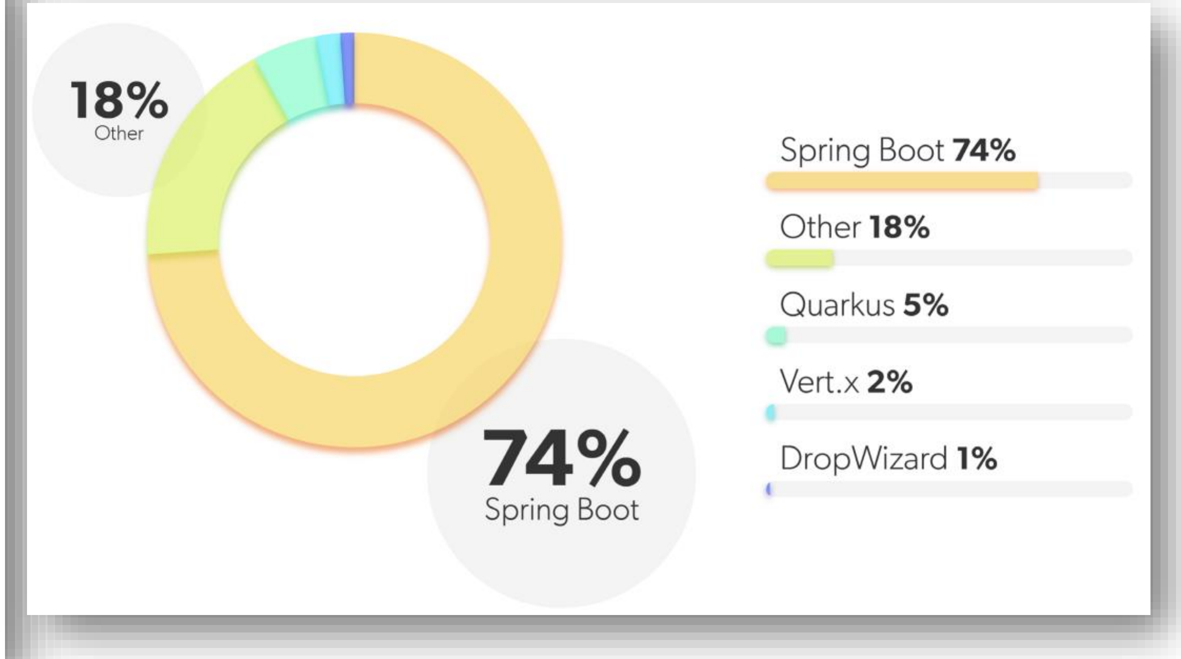
- What is the most used **architecture**?



Microservices

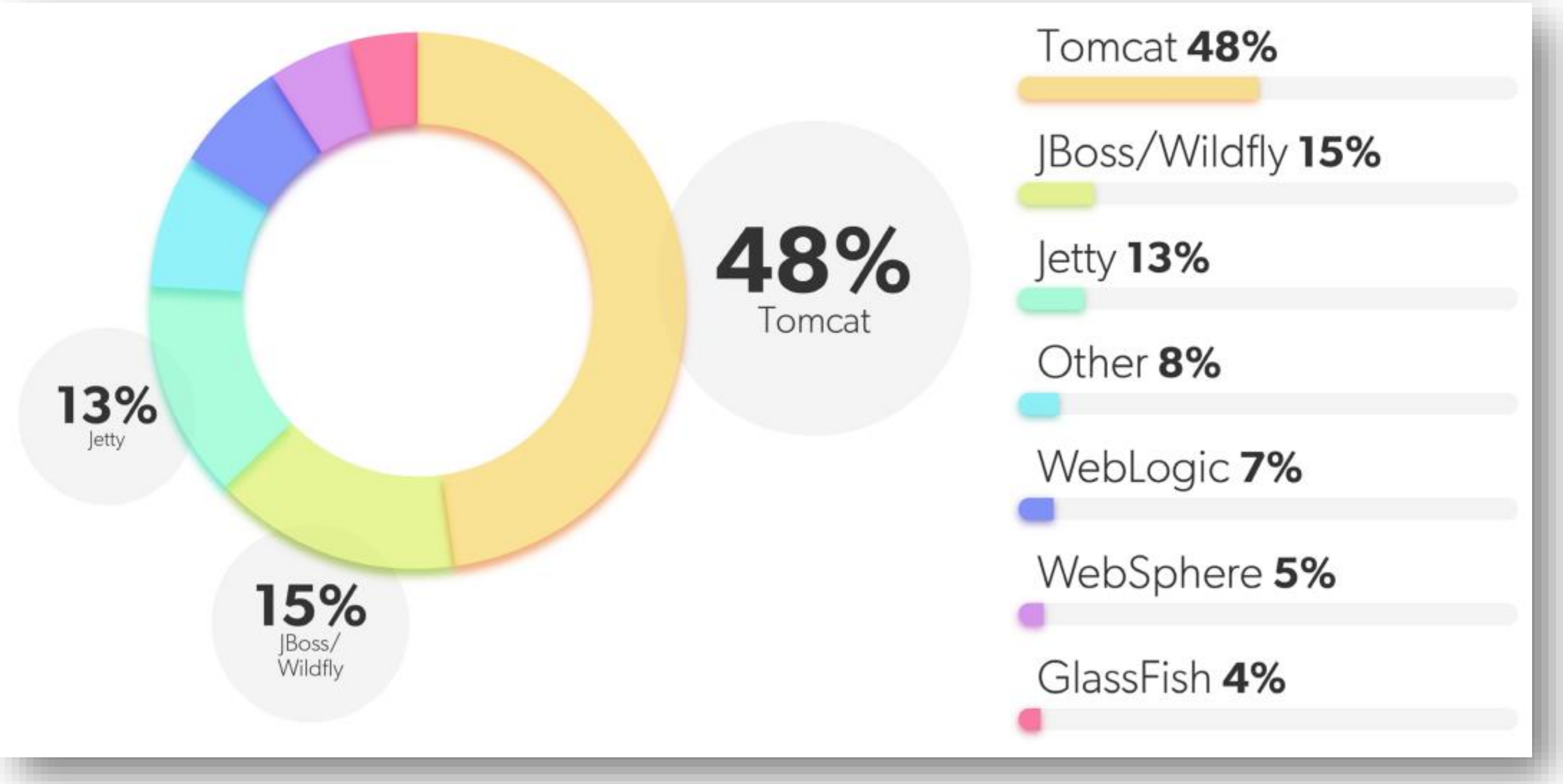
- What is your status for **Microservice** adoption? (JRebel)
- What **Microservice Application Framework** on your main project? (JRebel)





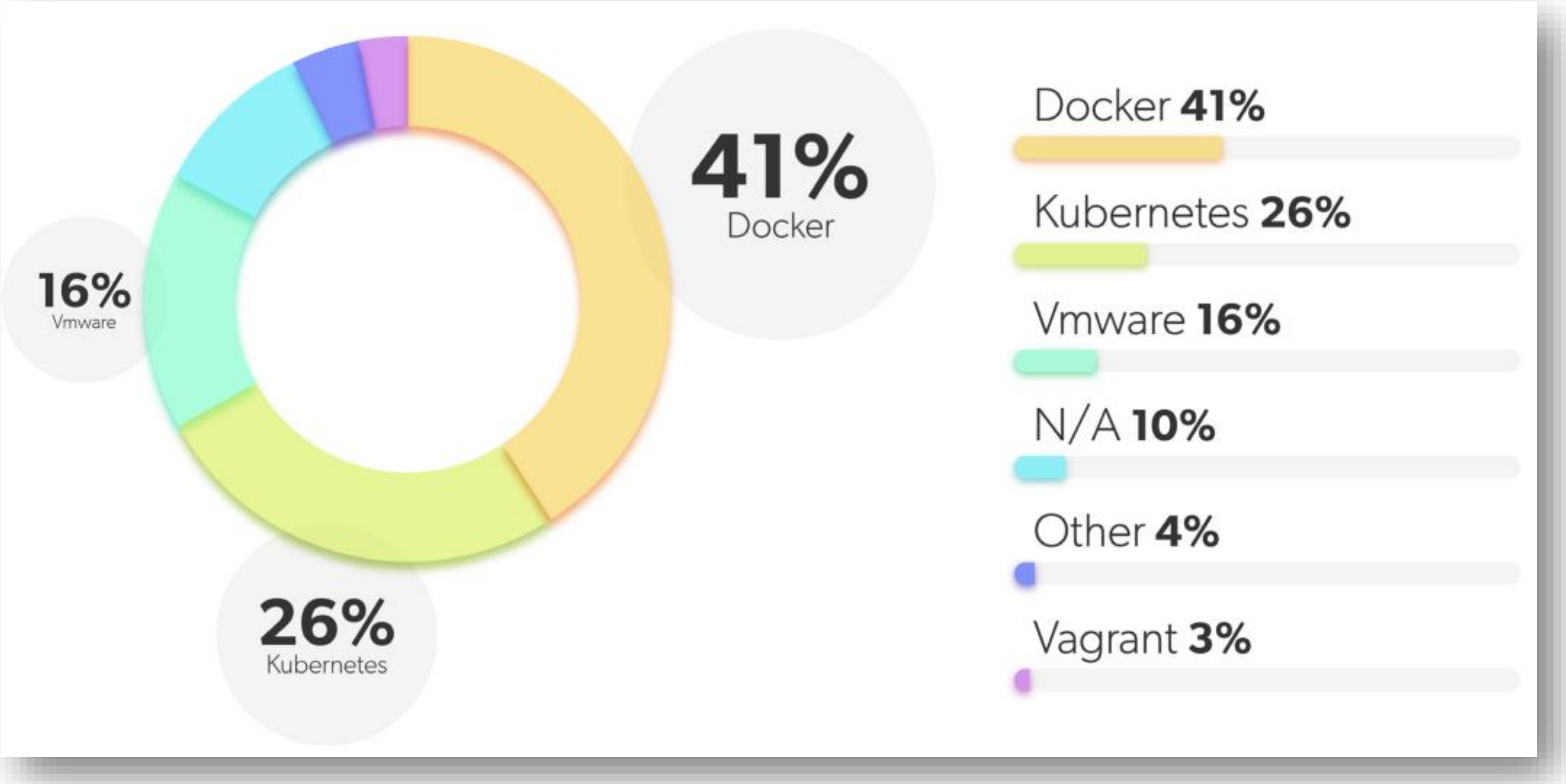
App Servers

- What **Application Server** do you use on your main application? (JRebel)



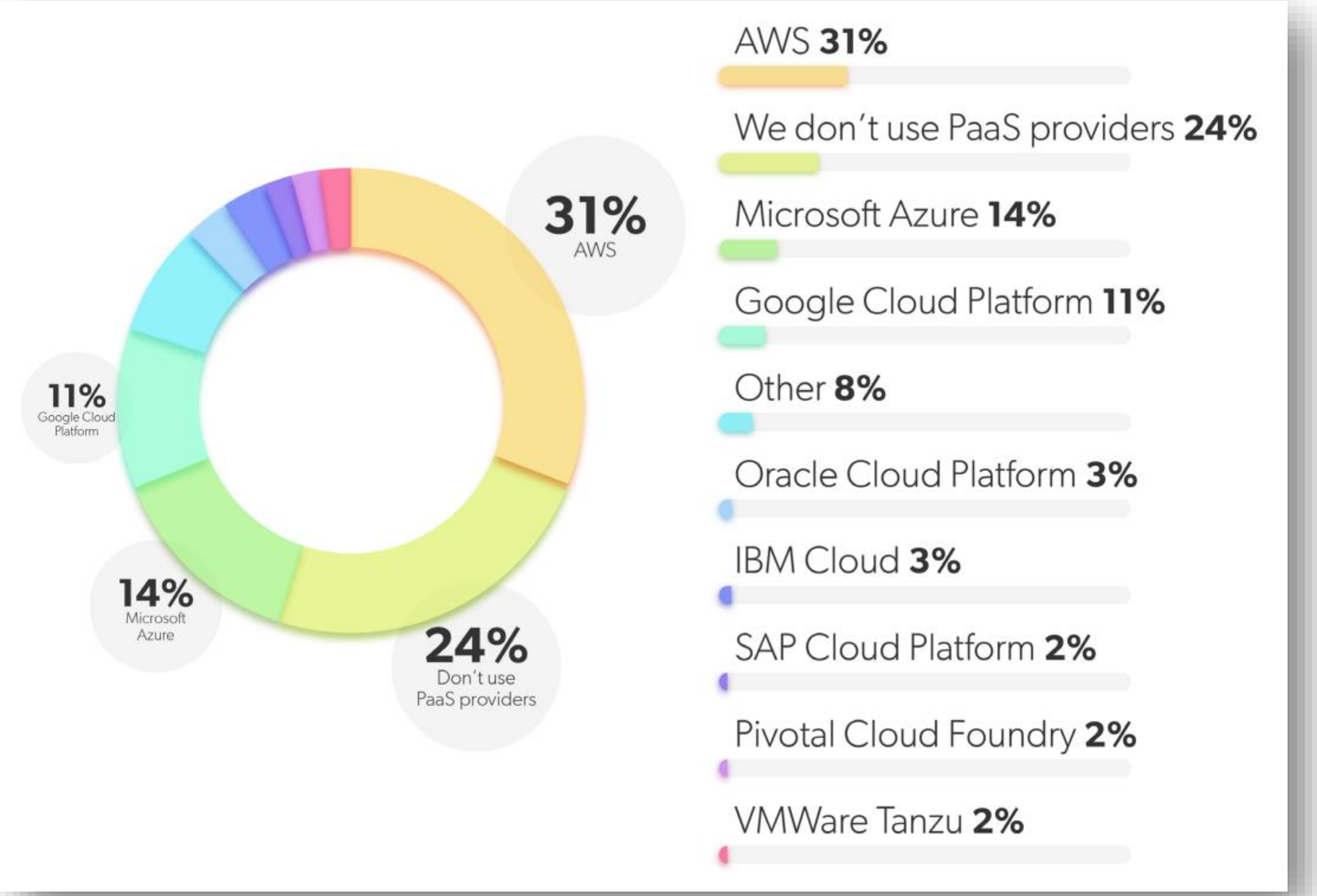
Virtual Machine

- Which **Virtual Machine Platform** do you use? (JRebel)



PaaS Providers

- If you use a platform, who is your **PaaS provider**?



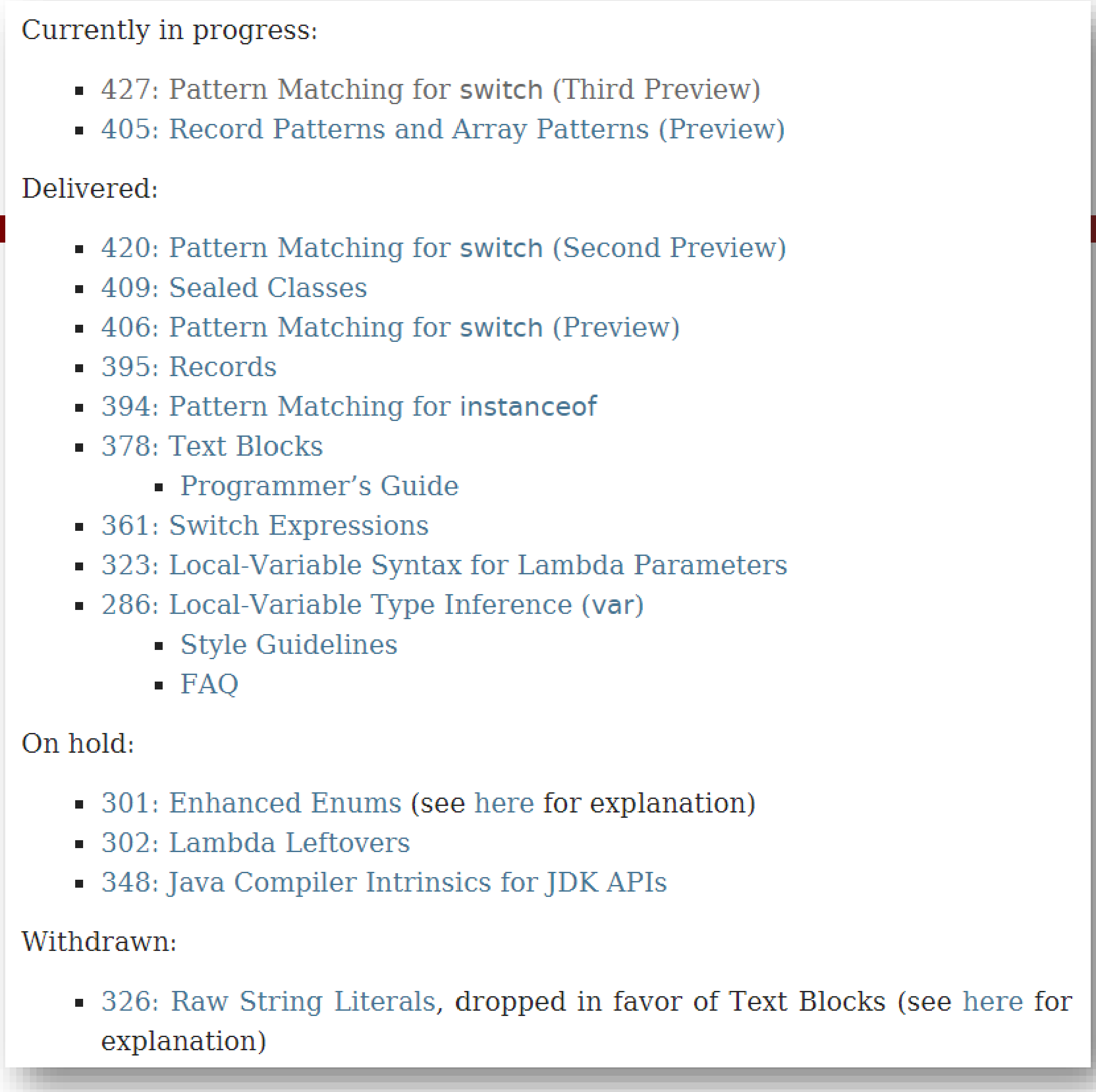
Projects – Long-term Java Future

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 **Loom** – to **increase performance** and **reduce complexity** in writing concurrent applications
- Project **Panama** – to interconnect JVM and **native** code
- Project **Metropolis** – JVM re-written in Java, i.e. "***Java on Java***"
- Project **Wakefield** – implement JDK support for Linux **Wayland** display server
- Project **Leyden** – improve **start-up time** to achieve peak performance

Project Amber

- **Right-sizing language** ceremony
  - Explore and incubate smaller, productivity-oriented Java language features
  - openjdk.java.net/projects/amber/
- Defined process:
  - First, must be 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



Currently in progress:

- 427: Pattern Matching for switch (Third Preview)
- 405: Record Patterns and Array Patterns (Preview)

Delivered:

- 420: Pattern Matching for switch (Second Preview)
- 409: Sealed Classes
- 406: Pattern Matching for switch (Preview)
- 395: Records
- 394: Pattern Matching for instanceof
- 378: Text Blocks
  - Programmer's Guide
- 361: Switch Expressions
- 323: Local-Variable Syntax for Lambda Parameters
- 286: Local-Variable Type Inference (var)
  - Style Guidelines
  - FAQ

On hold:

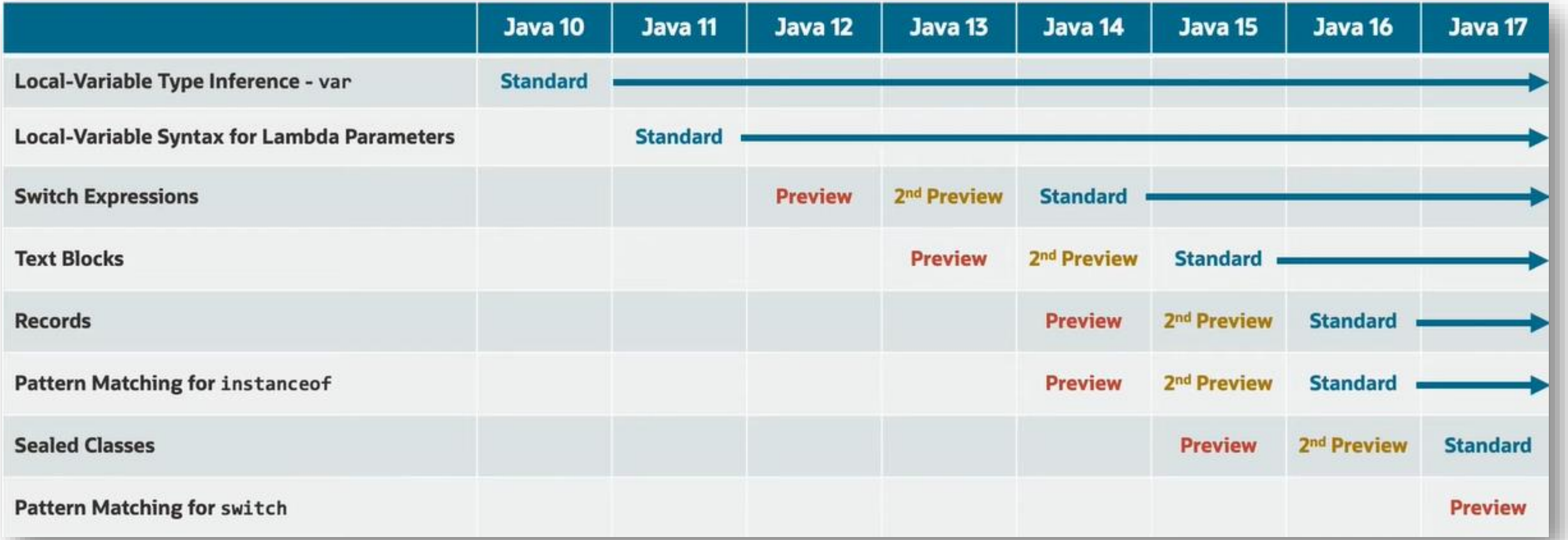
- 301: Enhanced Enums (see here for explanation)
- 302: Lambda Leftovers
- 348: Java Compiler Intrinsics for JDK APIs

Withdrawn:

- 326: Raw String Literals, dropped in favor of Text Blocks (see here for explanation)

Project Amber – Timeline Example

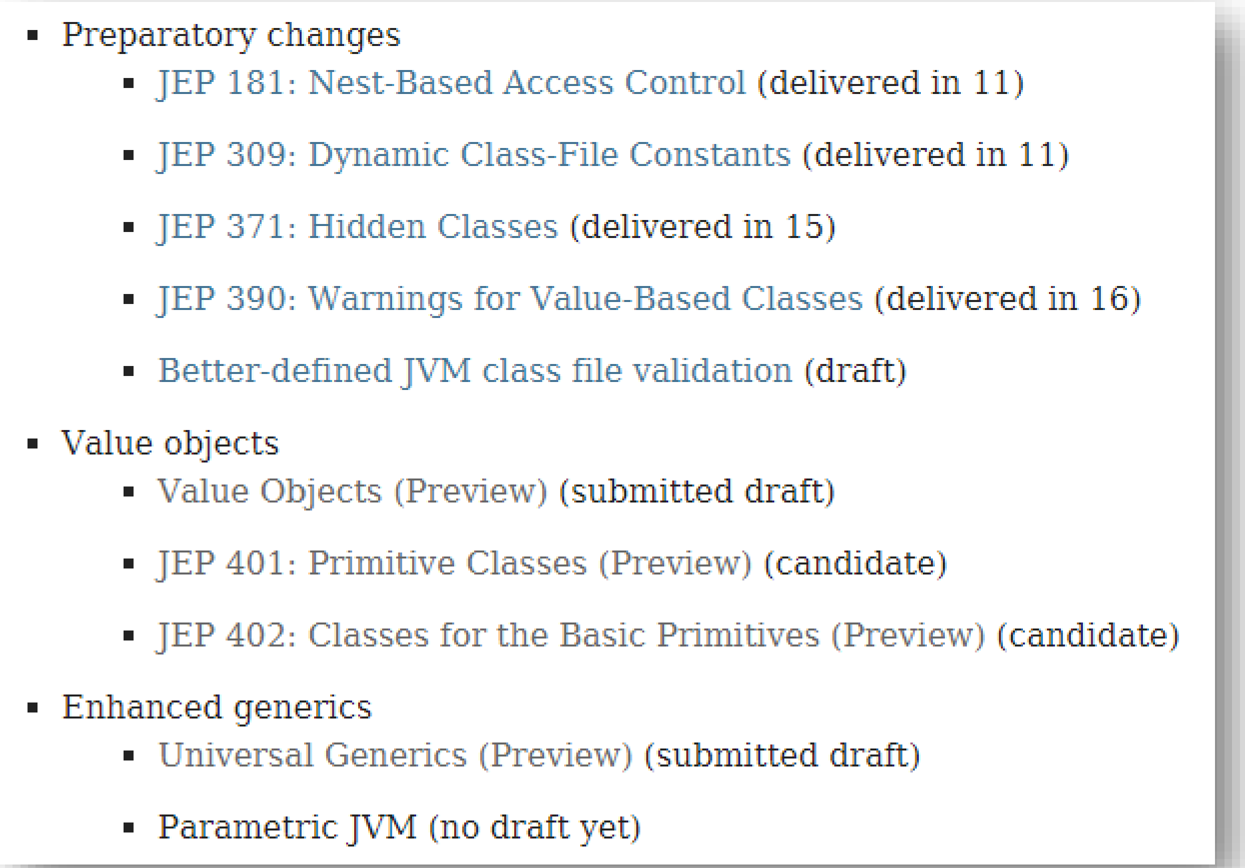
- Improving the programming language continuously



	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	2nd Preview	Standard	→	→	→
Text Blocks				Preview	2nd Preview	Standard	→	→
Records					Preview	2nd Preview	Standard	→
Pattern Matching for `instanceof`					Preview	2nd Preview	Standard	→
Sealed Classes						Preview	2nd Preview	Standard
Pattern Matching for `switch`								Preview

Project Valhalla

- Incubator project for more **advanced Java VM and language feature** candidates
  - openjdk.java.net/projects/valhalla/
- 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…



- Preparatory changes
  - JEP 181: Nest-Based Access Control (delivered in 11)
  - JEP 309: Dynamic Class-File Constants (delivered in 11)
  - JEP 371: Hidden Classes (delivered in 15)
  - JEP 390: Warnings for Value-Based Classes (delivered in 16)
  - Better-defined JVM class file validation (draft)
- Value objects
  - Value Objects (Preview) (submitted draft)
  - JEP 401: Primitive Classes (Preview) (candidate)
  - JEP 402: Classes for the Basic Primitives (Preview) (candidate)
- Enhanced generics
  - Universal Generics (Preview) (submitted draft)
  - Parametric JVM (no draft yet)

Project **Valhalla**

- **Value Objects** (Preview) in Draft – "*codes like a class, works like a primitive*"
  - Supports methods, fields, implements interface, encapsulation, generic, but not support mutation or sub-classes
- **Primitive Classes** (JEP 401, Preview) in Candidate
  - Special kinds of value classes that define new developer-declared primitive types
- **Classes for the Basic Primitives** (JEP 402, Preview) in Candidate
  - Repurpose the primitive wrapper classes to act as declarations for the basic primitives (int, double, etc.)
- **Universal Generics** (Preview) in Draft
  - Unify the treatment of reference and primitive types in generic code by allowing Java type variables to range over both kinds of types

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…

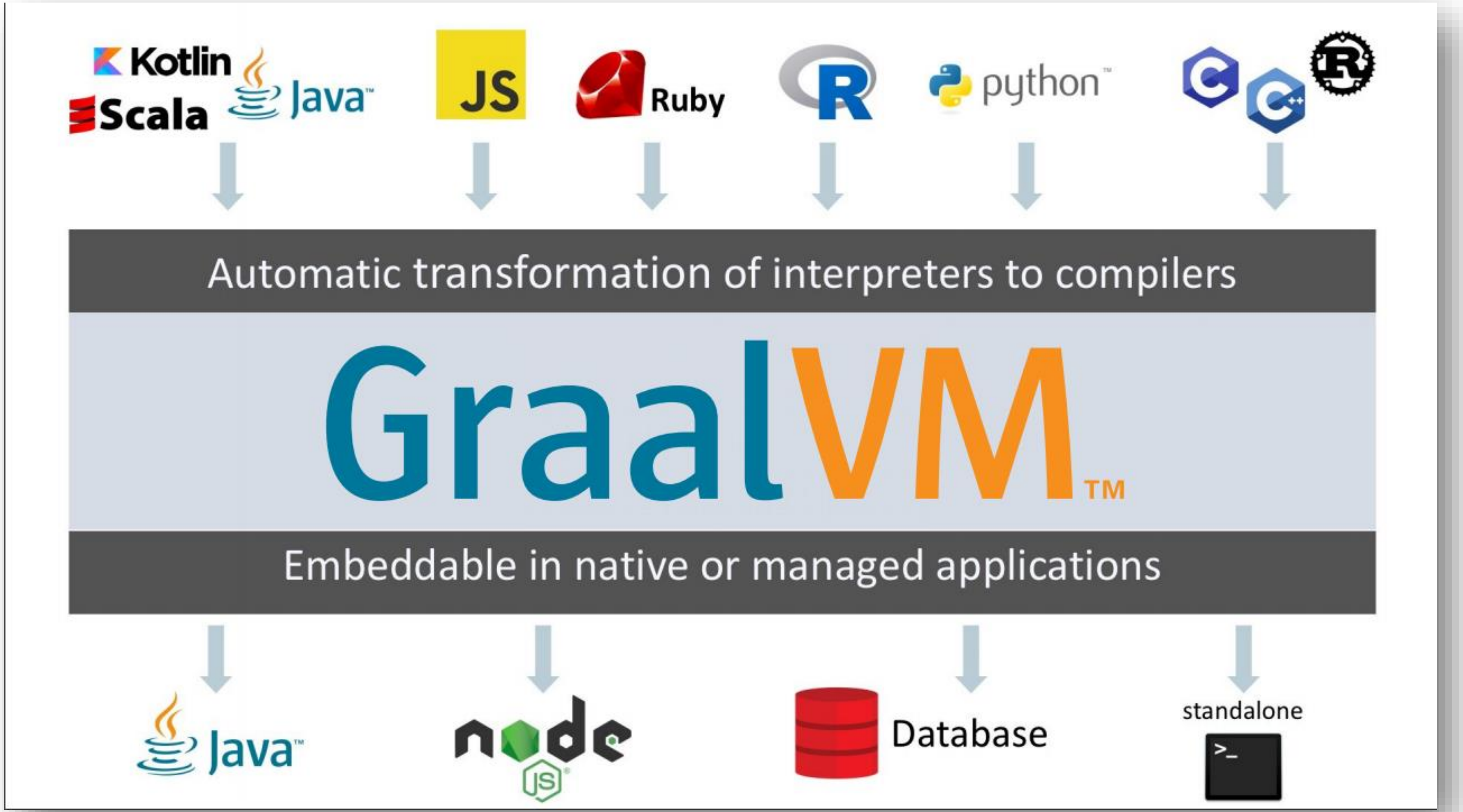
Project Loom

- **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

Programming Polyglotism & other

- **Polyglot programming** and **cross-language interoperability**
- OpenJDK's Project **Metropolis**
- **GraalVM** – high-performance embeddable polyglot virtual machine
  - **combine different programming languages** that incur almost no overhead
  - In the JVM, into standalone native image, or embedded into large application
- **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**
- **+ Quarkus: Kubernetes** Native Java stack tailored for **GraalVM**

GraalVM – architecture



Automatic transformation of interpreters to compilers

GraalVM™

Embeddable in native or managed applications

Where can you learn Java?

- On every major university in the world

MIT

STANFORD

Caltech

VE RI TAS HARVARD

ETH zürich

- On all major online learning and MOOC platforms

edX® & RITX

coursera

ORACLE® UNIVERSITY

Udemy

treehouse™

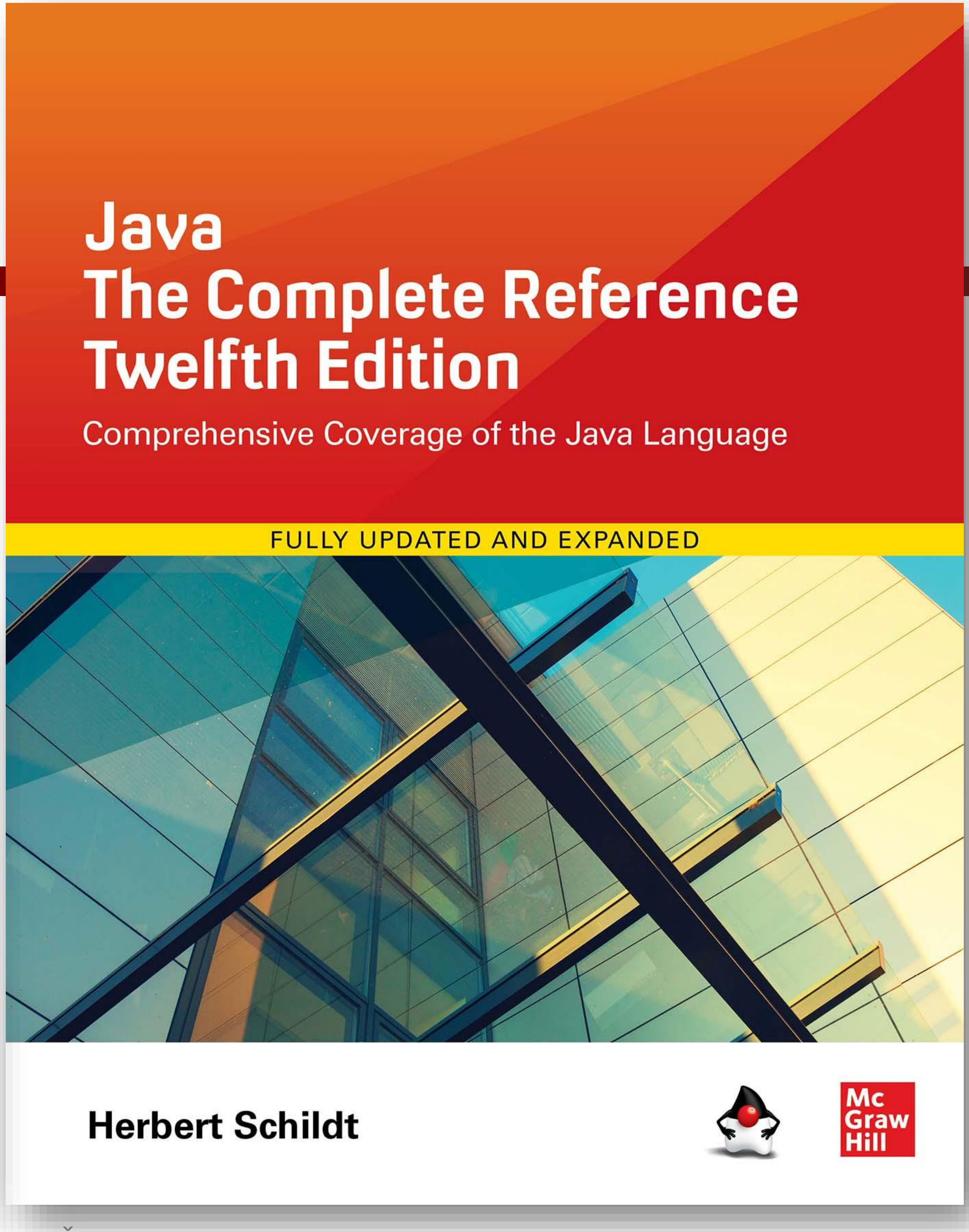
in LEARNING WITH Lynda.com CONTENT

PLURALSIGHT

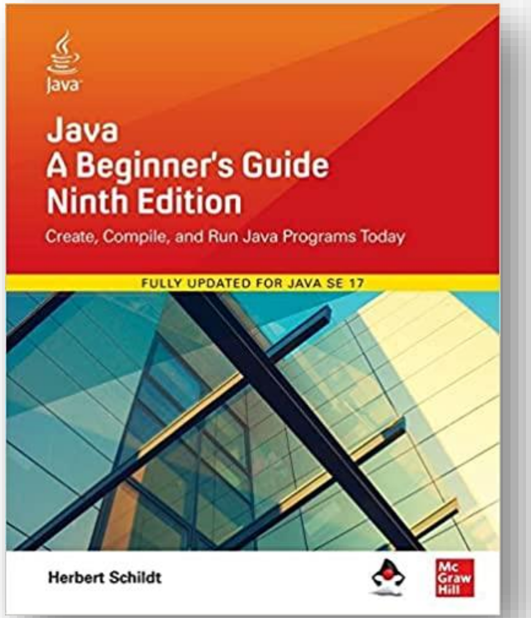
UDACITY

New Books

- **Java: The Complete Reference, 12th ed.,** Herbert Schildt, MGH, November 2021
  - ISBN: 9781260463415
  - 1280 pages
  - 45-60 €

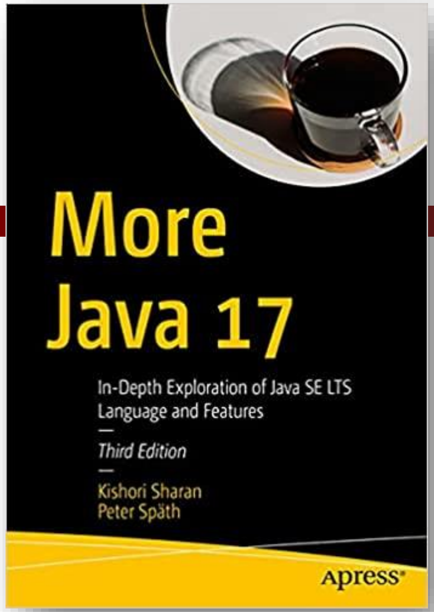

Java
The Complete Reference
Twelfth Edition
Comprehensive Coverage of the Java Language
FULLY UPDATED AND EXPANDED
Herbert Schildt
Mc Graw Hill

- **Java: A Beginner's Guide, 9th ed.**, Herbert Schildt, MGH, January 2022
  - ISBN: 9781260463552
  - 728 pages
  - 30-35 €


Java
Java
A Beginner's Guide
Ninth Edition
Create, Compile, and Run Java Programs Today
FULLY UPDATED FOR JAVA SE 17
Herbert Schildt
Mc Graw Hill

More New Books

- **Beginning Java 17 Fundamentals, 3rd ed.** by Kishori Sharan, Adam L. Davis, Apress, Nov 2021
  - 9781484273067, 800 pages, approx. 45 €
- **More Java 17, 3rd ed.** by Kishori Sharan, Peter Späth, Apress, Dec 2021
  - 9781484271346, approx. 64 €
- **Java 17 Quick Syntax Reference** by Mikael Olsson, Apress, Oct 2021
  - 9781484273708, 218 pages, approx. 26 €
- **Java 17 for Absolute Beginners** by Iuliana Cosmina, Apress, Dec 2021
  - 9781484270790, 600 pages, approx. 42 €



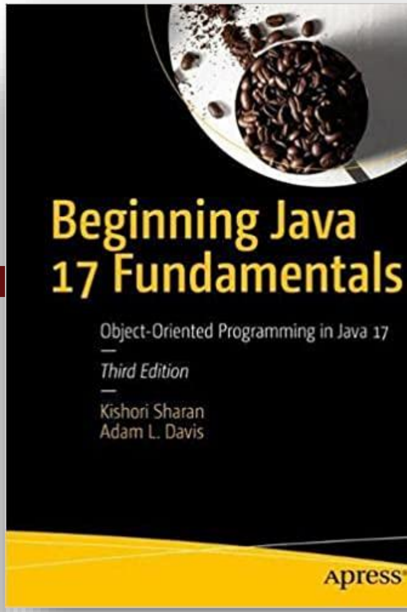
More Java 17

In-Depth Exploration of Java SE LTS Language and Features

Third Edition

Kishori Sharan
Peter Späth

Apress®



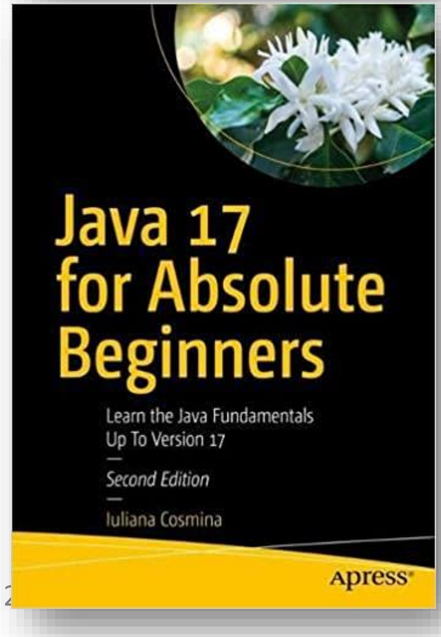
Beginning Java 17 Fundamentals

Object-Oriented Programming in Java 17

Third Edition

Kishori Sharan
Adam L. Davis

Apress®



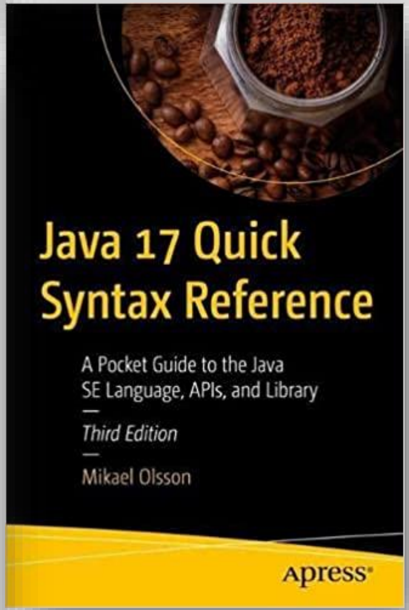
Java 17 for Absolute Beginners

Learn the Java Fundamentals Up To Version 17

Second Edition

Iuliana Cosmina

Apress®



Java 17 Quick Syntax Reference

A Pocket Guide to the Java SE Language, APIs, and Library

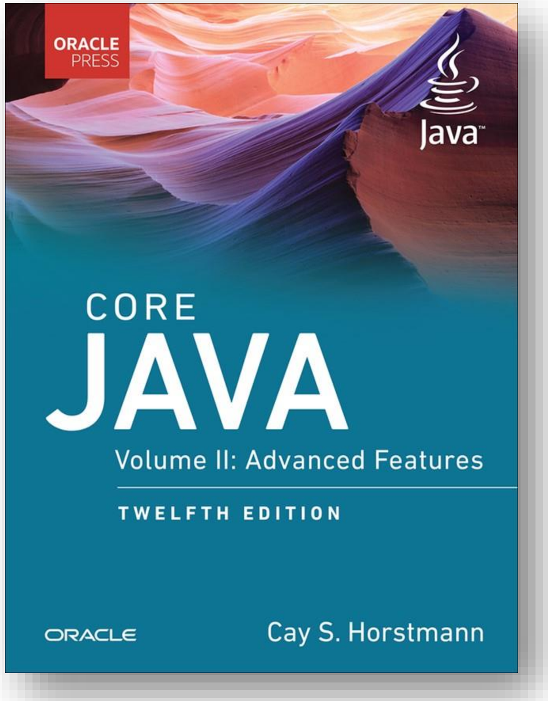
Third Edition

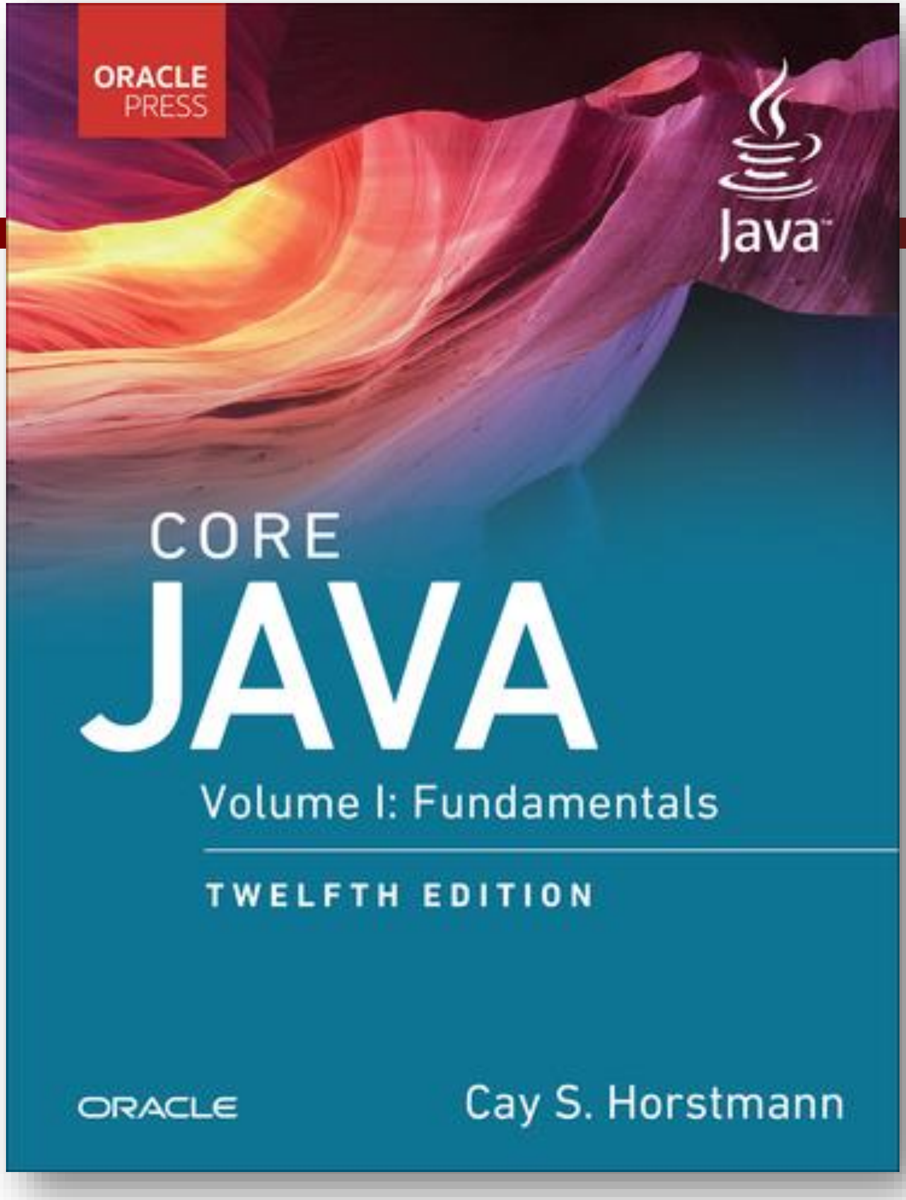
Mikael Olsson

Apress®

More New **Books**

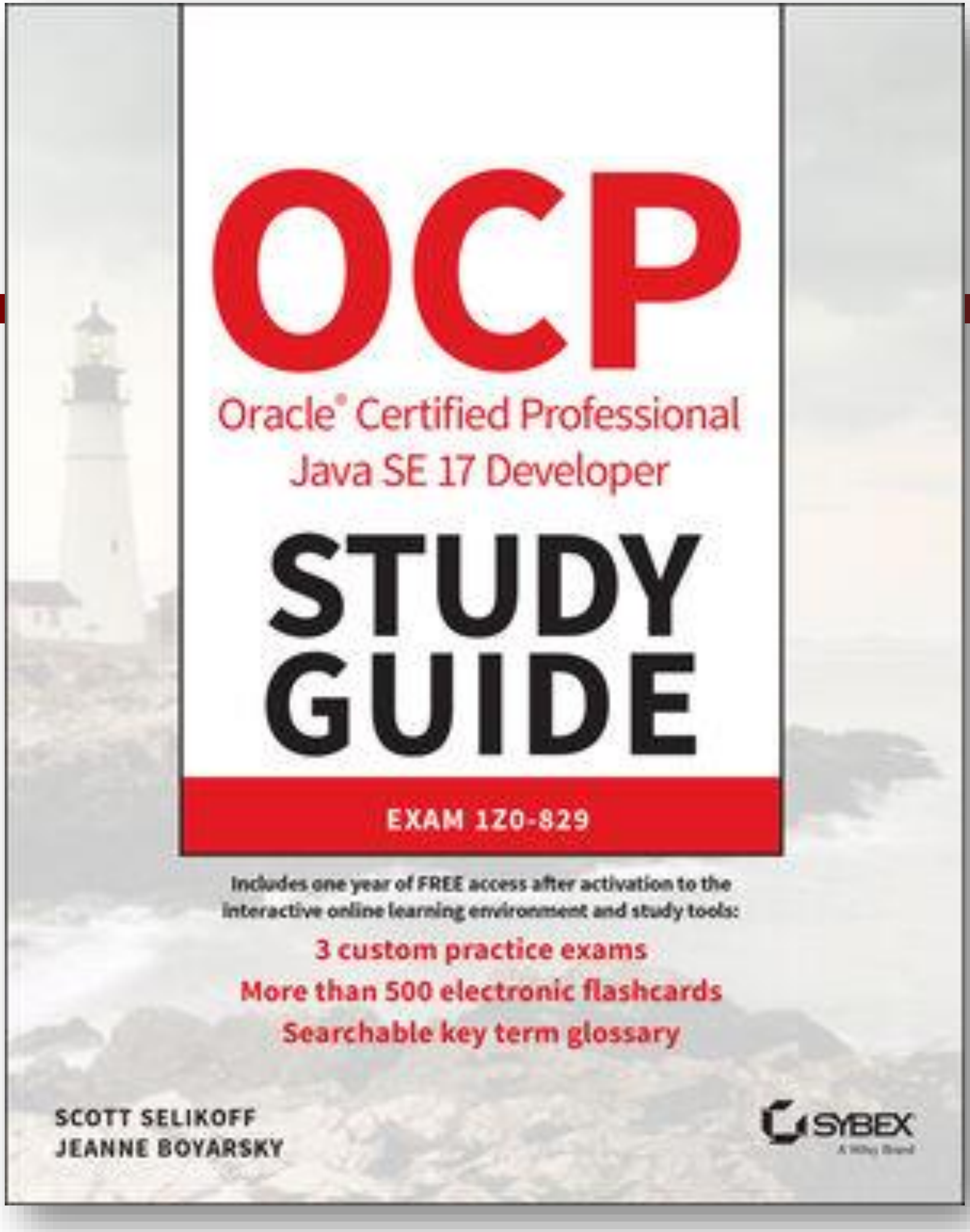
- **Core Java, Volume I: Fundamentals, 12th ed.,** Cay S. Horstmann, December 2021, Oracle Press
  - ISBN: 9780137673629
  - 861 pages
  - Approx. 60 €
- **Core Java, Volume II: Advanced Features, 12th ed.,** Cay S. Horstmann, April 2022, Oracle Press
  - ISBN: 9780137870899
  - 1185 pages
  - Approx. 50 €





OCP Books

- **OCP Oracle Certified Professional Java SE 17 Developer Study Guide: Exam 1Z0-829,** by Scott Selikoff and Jeanne Boyarsky, May 2022, Wiley
  - ISBN: 9781119864585
  - 1056 pages
  - Approx. 60 €



OCP

Oracle® Certified Professional
Java SE 17 Developer

STUDY GUIDE

EXAM 1Z0-829

Includes one year of FREE access after activation to the interactive online learning environment and study tools:

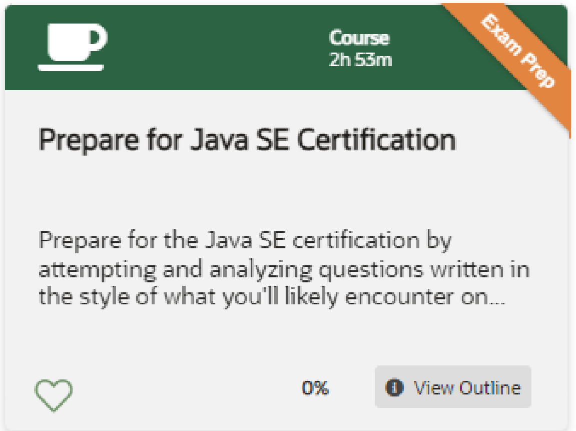
3 custom practice exams
More than 500 electronic flashcards
Searchable key term glossary

SCOTT SELIKOFF
JEANNE BOYARSKY

SYBEX
A Wiley Brand

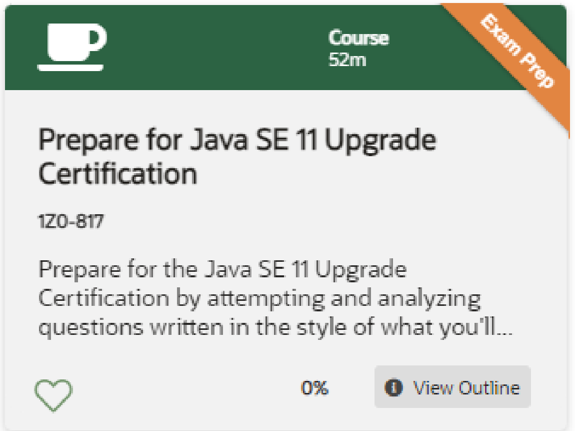
Learning Java

- Free **basic Java training** and accreditation
  - education.oracle.com/learning-explorer
- **Java Learning Subscription** by Oracle
  - education.oracle.com/java-programming-learning-subscription/ls_40805
  - Entry level training is free
- Learning paths, Courses, Live sessions, User Forum
  - learn.oracle.com/ols/live-events/java-learning-subscription/40805
- **Java Explorer** Learning Path
  - 7.5 hours of free training & free assessment
- Java **Exam Preparation**
  - Only for Java 11 ☹
- Oracle University Free Resources



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...

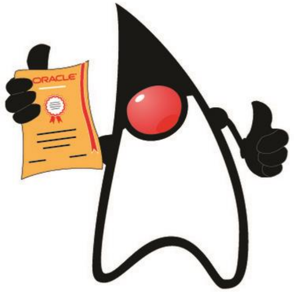


Prepare for Java SE 11 Upgrade Certification

1Z0-817

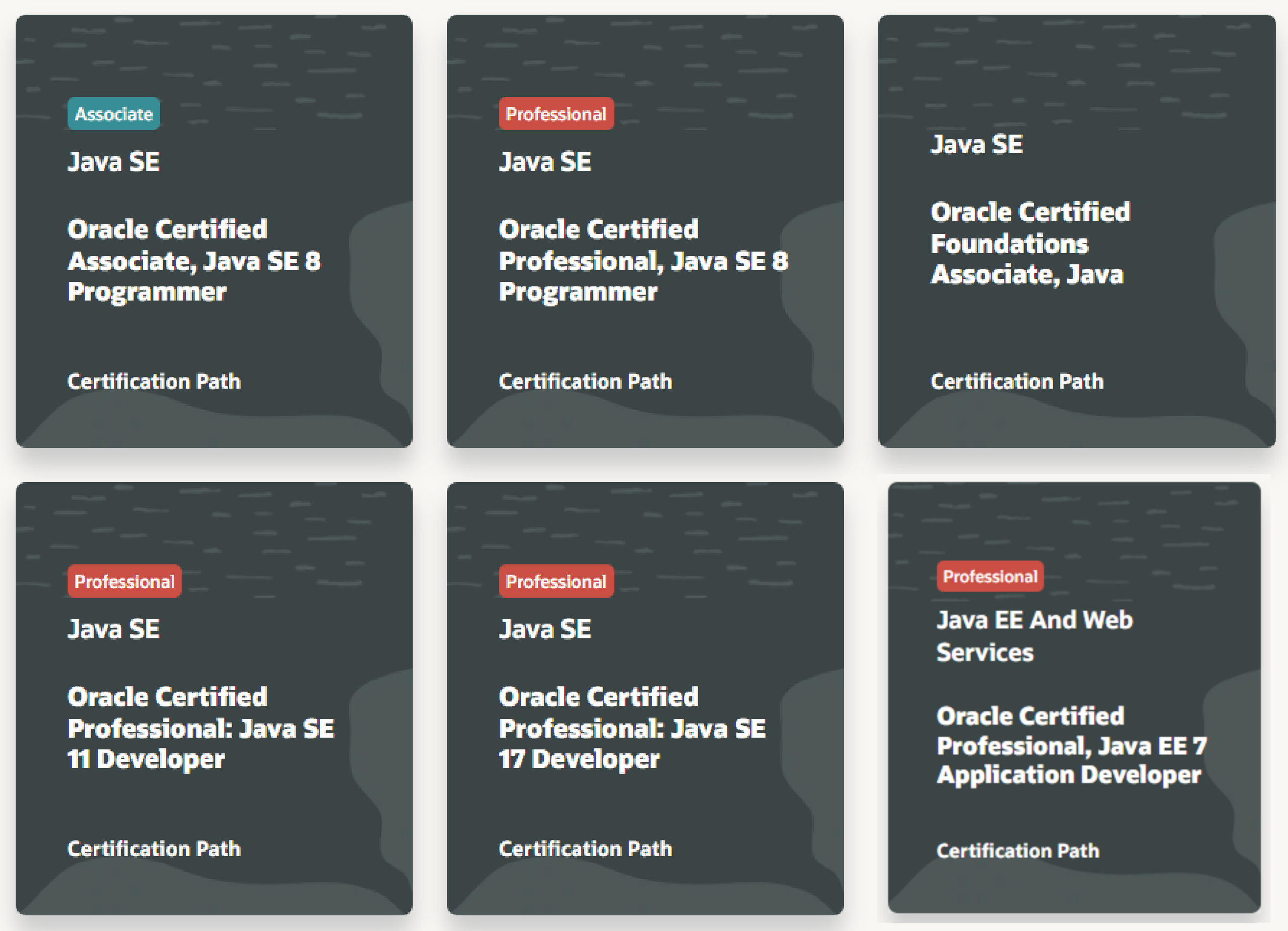
Prepare for the Java SE 11 Upgrade Certification by attempting and analyzing questions written in the style of what you'll...

Java **Certification**



- **Oracle Java Certification Paths**

- **Oracle Certified Professional (OCP): Java SE 17 Developer**
- Oracle Certified Professional (OCP): Java SE 11 Developer
- Oracle Certified Professional (OCP): Java SE 8 Developer
- Oracle Certified Associate (OCA), Java SE 8 Programmer
- Oracle Certified Foundations Associate, Java



Associate
Java SE
Oracle Certified Associate, Java SE 8 Programmer
Certification Path

Professional
Java SE
Oracle Certified Professional, Java SE 8 Programmer
Certification Path

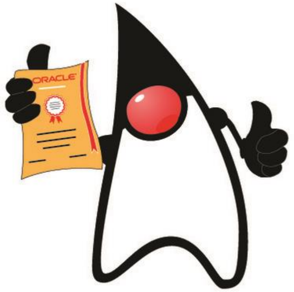
Java SE
Oracle Certified Foundations Associate, Java
Certification Path

Professional
Java SE
Oracle Certified Professional: Java SE 11 Developer
Certification Path

Professional
Java SE
Oracle Certified Professional: Java SE 17 Developer
Certification Path

Professional
Java EE And Web Services
Oracle Certified Professional, Java EE 7 Application Developer
Certification Path

Java Certifications



- **Oracle Certified Professional (OCP): Java SE 17 Developer**
  - Exam: Java SE 17 Developer 1Z0-829
  - Price: 1634 kn | Duration: 90 Minutes | Passing Score: 68%
- **Oracle Certified Professional (OCP): Java SE 11 Developer**
  - Exam: Java SE 11 Developer 1Z0-819
  - Price: 1634 kn | Duration: 90 Minutes | Passing Score: 68%
- For students - **Oracle Certified Foundations Associate, Java**
  - Exam: Java Foundations 1Z0-811
  - Price: 634 kn | Duration: 150 Minutes | Passing Score: 65%
  - Part of Oracle Academy offerings for students

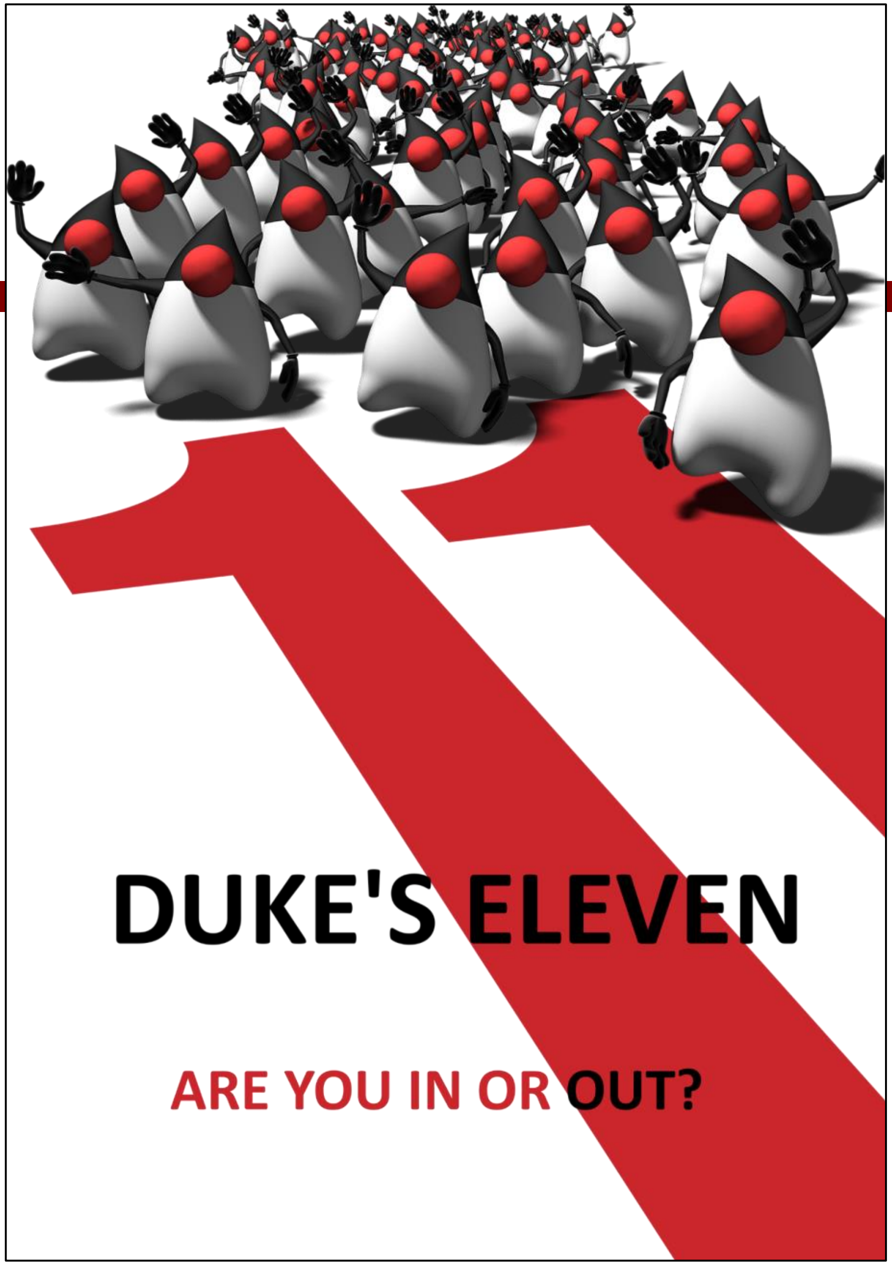
Is Java really "Moving Forward Faster"?

- Community opinion ☺
- More frequent Java releases every 6 months
- Java LTS releases every 2 years
- Faster access to new features
- Many new improvement ideas
- A lot of maintenance and housekeeping
- Java is (finally) free

Looking forward to new things!

What is our **advice**?

- Obviously – **use Java 17 LTS** ☺
  or the latest **Java 18**
  or at least use **Java 11 LTS**
- **Any JDK** or any other – **it's up to you** ☺
- Try to **abandon older versions** (Java 8 or older)
- **Check** what is **@Deprecated**
- **Migrate** every **6 months** or **2 years** (with LTS)
- **Get involved** more with **HUJAK** and
  visit more to **conferences**!



Instead of conclusion…

Call for Speakers
will open soon for
JavaCro'22 Fall
in **October** in **Rovinj**

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