Java

Community Keynote

dr. sc. **Branko Mihaljević**

dr. sc. **Aleksander Radovan**

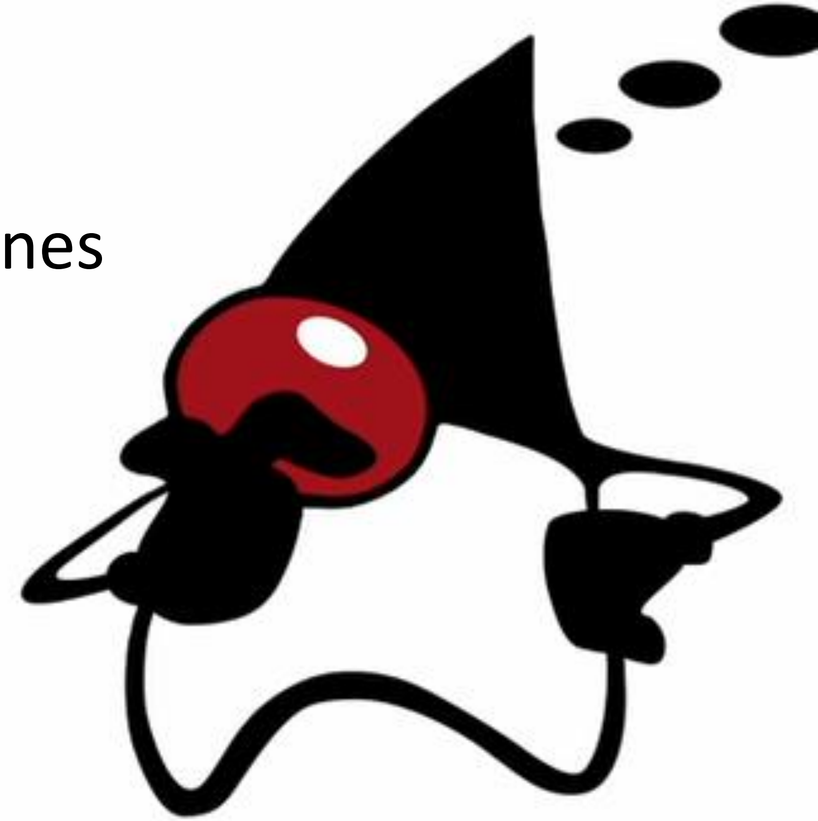
Stjepan Matijašević

dr. sc. **Martin Žagar**

HUJAK

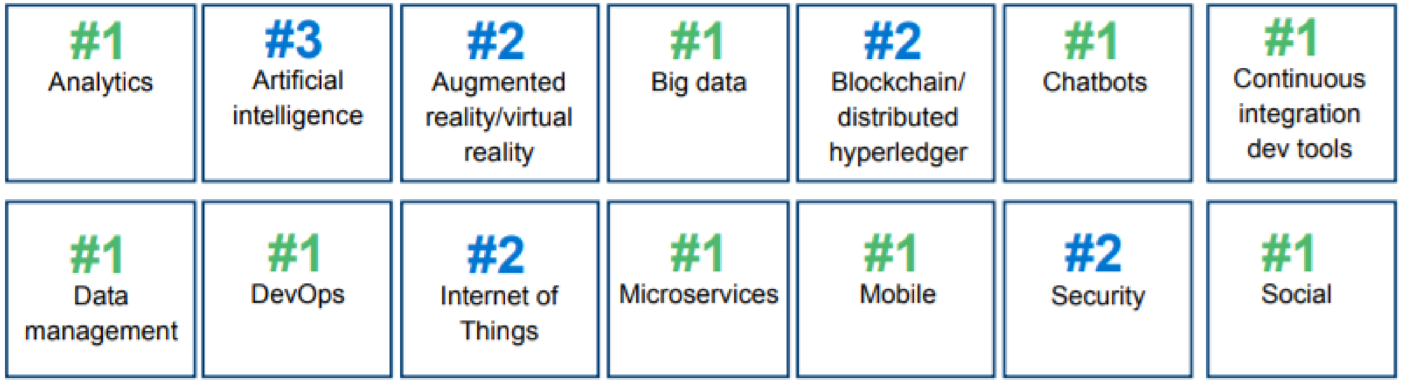
Assessing the New Development Landscape

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

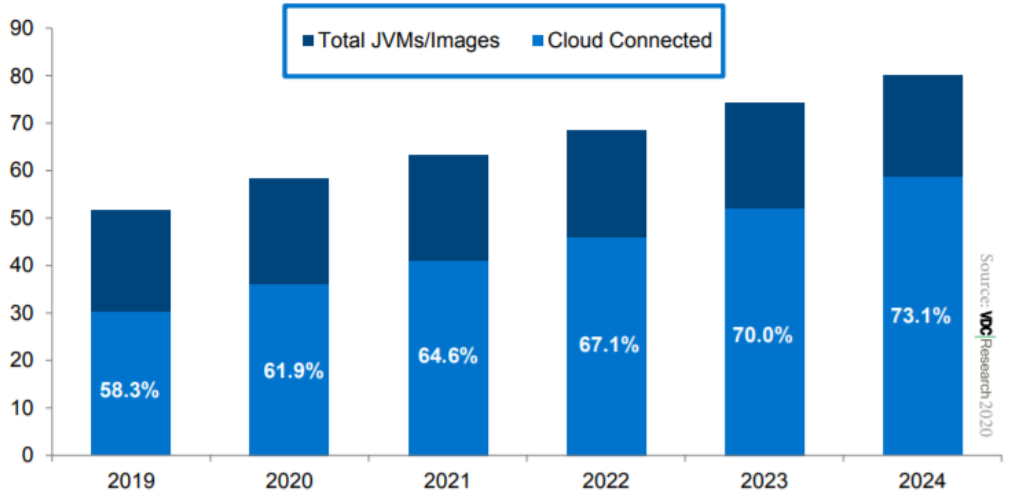


Java **Facts**

- **#1 Development Platform**
  - **Continued growth for 27+ years**
- **#1 Programming Language**
  - **In overall development**
- **38 Billion Cloud-based JVMs**
- **60 Billion Active JVMs**
  - **65%** 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?

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

More Java **Facts**

- **10 Million Java Developers**
    - With many **Java Certificates**

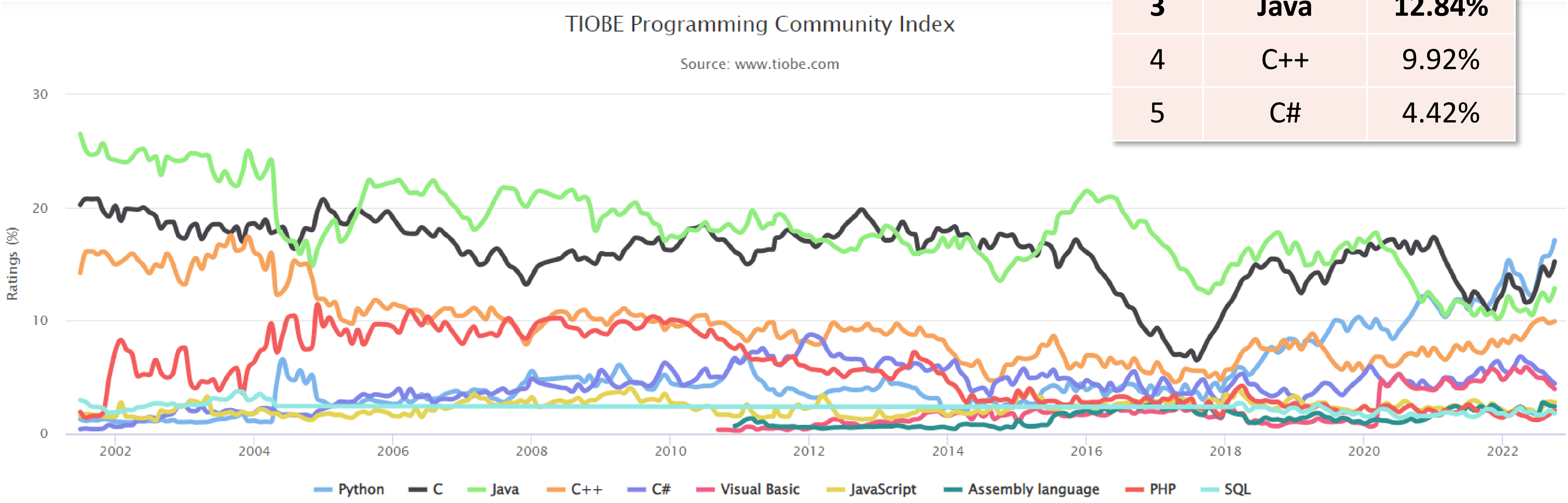
- **69%** of **Software Developers** run (some kind of…) Java

- **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 October 2022

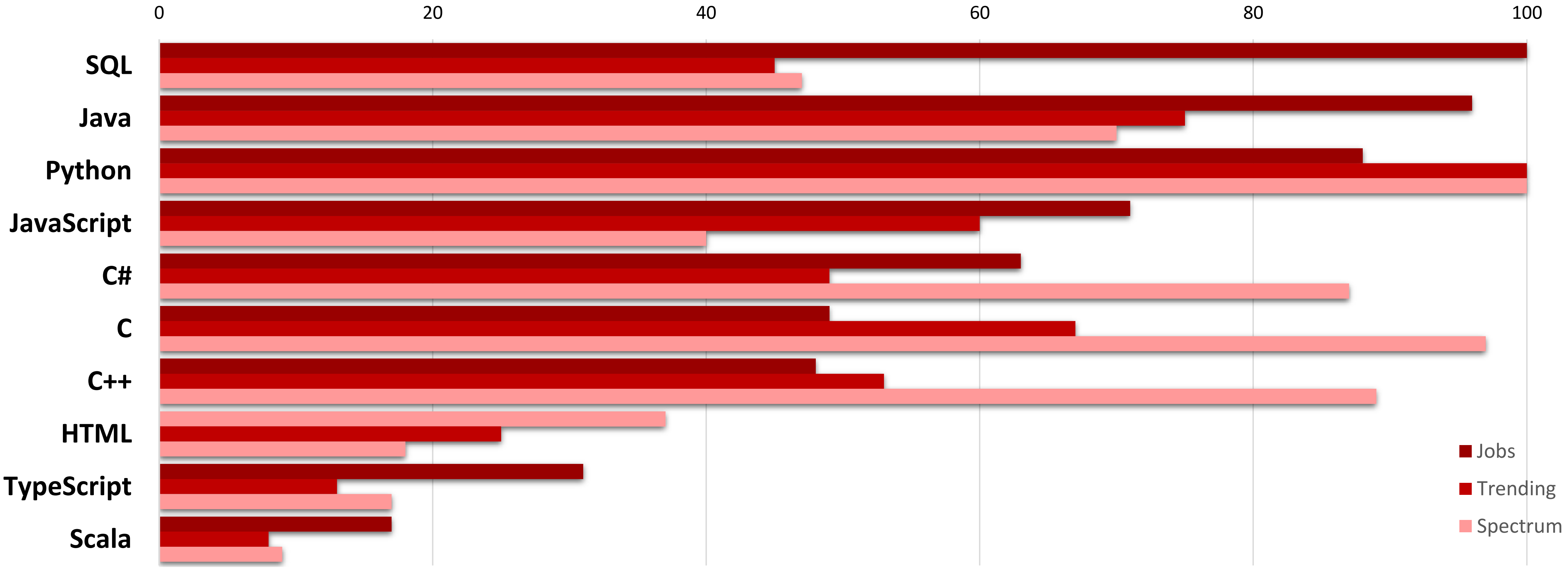
Rank	Language	Ratings
1	Python	17.08%
2	C	15.21%
3	**Java**	**12.84%**
4	C++	9.92%
5	C#	4.42%



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

Is Java still popular? #2

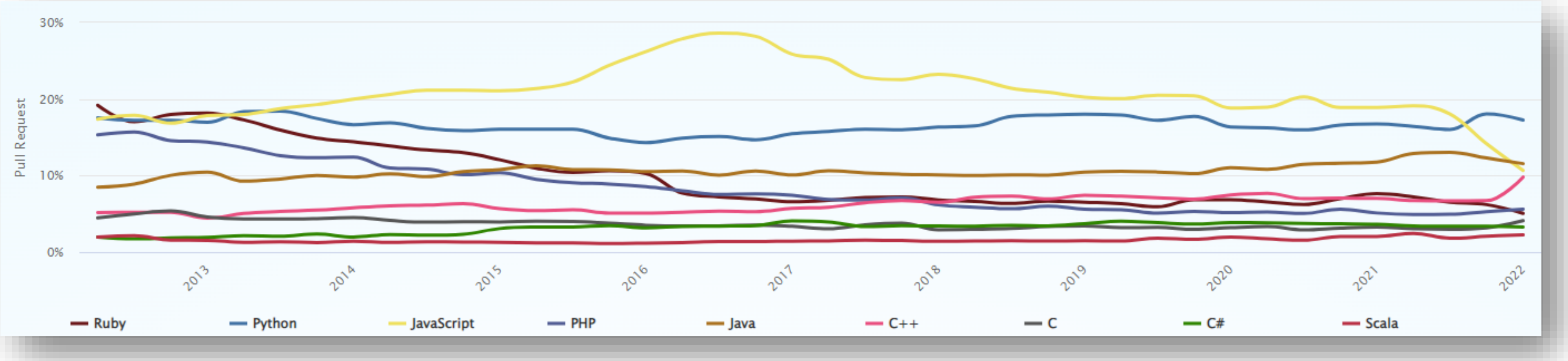
- **Top Programming Languages 2022** by IEEE Spectrum



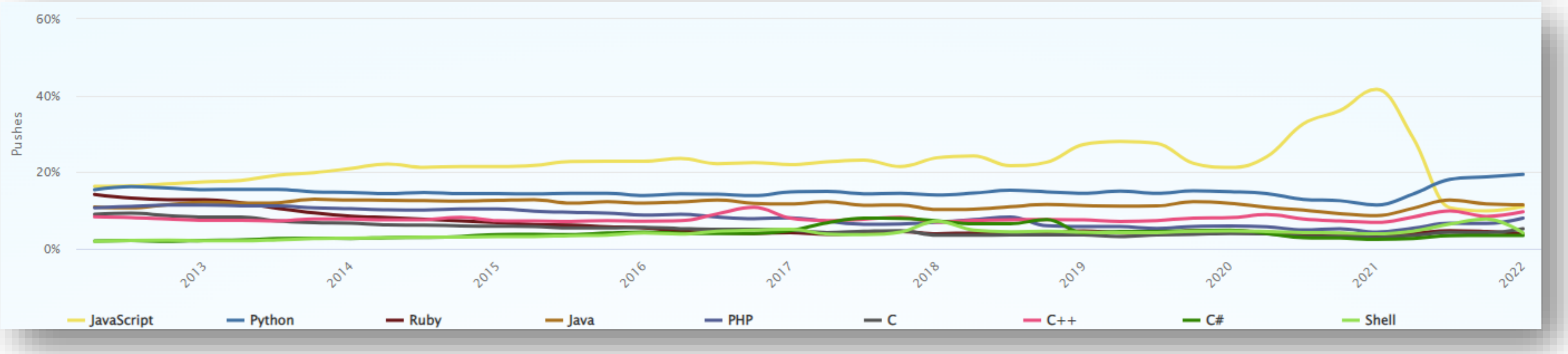
Source: Top Programming Languages 2022, August 2022, IEEE Spectrum, spectrum.ieee.org/top-programming-languages-2022

Is Java **still popular**? #3

- **GitHut 2.0 in 2022 – Pull Requests** and **Pushes**



Rank	Language	Ratings
1	Python	17.2%
2	**Java**	**11.5%**
3	JavaScript	10.6%
4	C++	9.8%
5	Go	8.3%

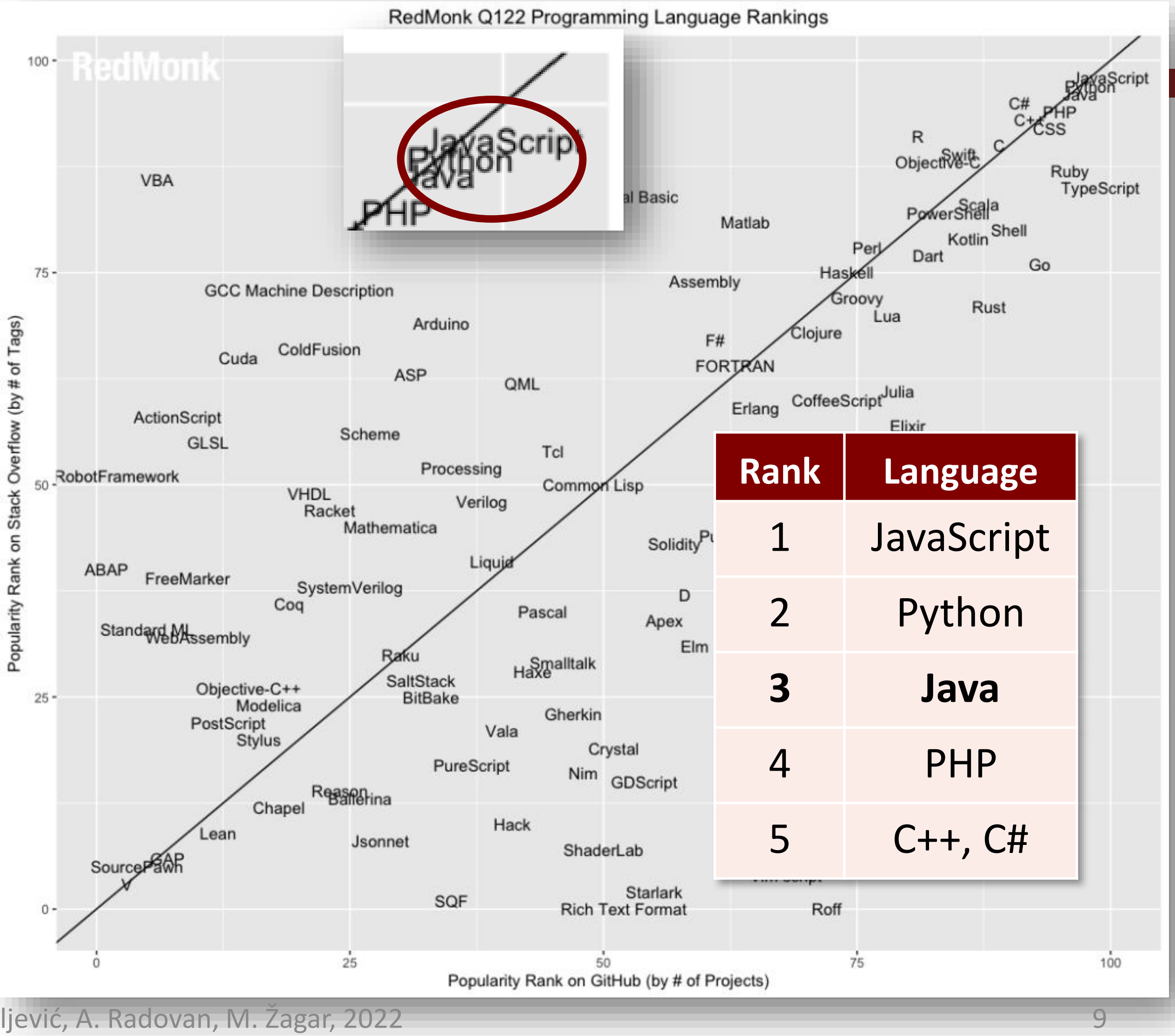


Rank	Language	Ratings
1	Python	19.2%
2	**Java**	**11.4%**
3	JavaScript	10.9%
4	C++	9.6%
5	PHP	7.9%

Source: GitHut 2.0, madnight.github.io/githut/#/pull_requests/2022/1

Is Java still popular? #4

- **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

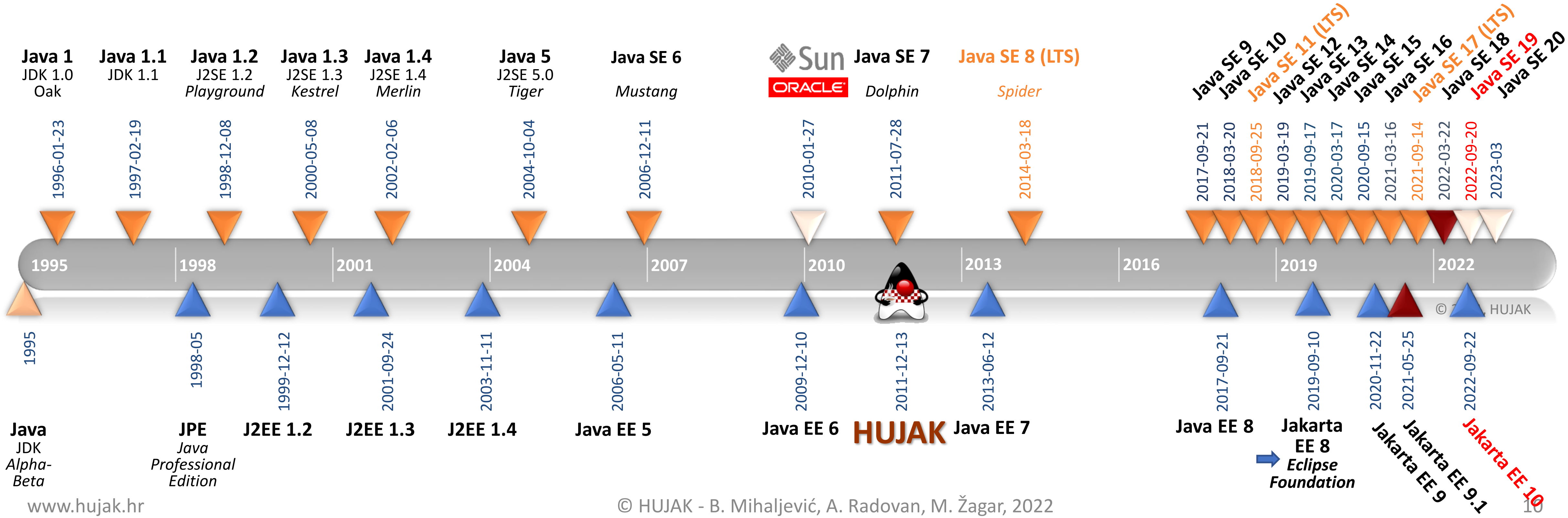


Rank	Language
1	JavaScript
2	Python
3	**Java**
4	PHP
5	C++, C#

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

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 2022-09-20
Java SE 20 2023-03

1995 1998 2001 2004 2007 2010 2013 2016 2019 2022

© 2…, HUJAK

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

2022-09-22
Jakarta EE 10

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

Is Java **Moving Forward**?

Predictability

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

Trust

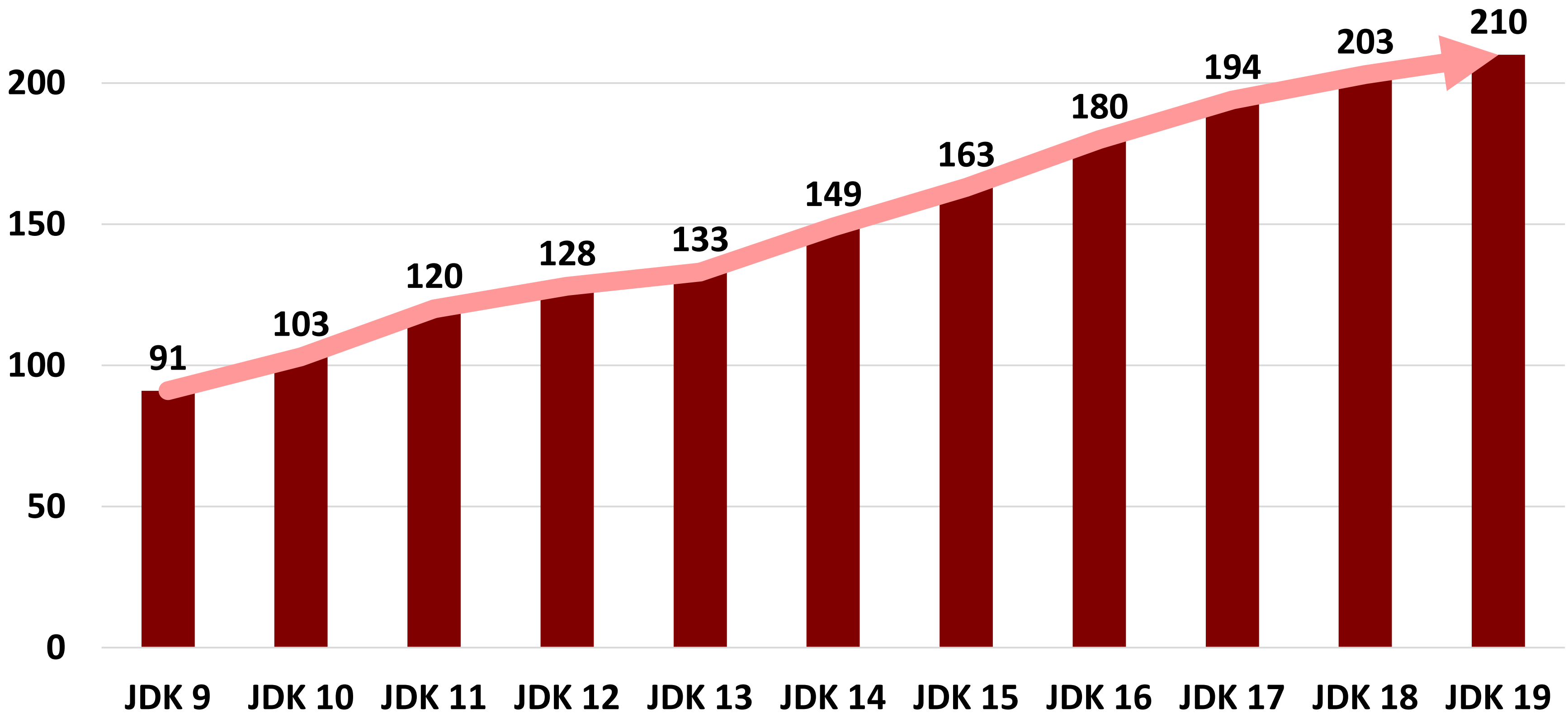
- Open and transparent development model, preserving values – **no surprises**
- Java community suggests and adopts new features – **community involvement**

Innovation

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

Constant Evolution through JEPs

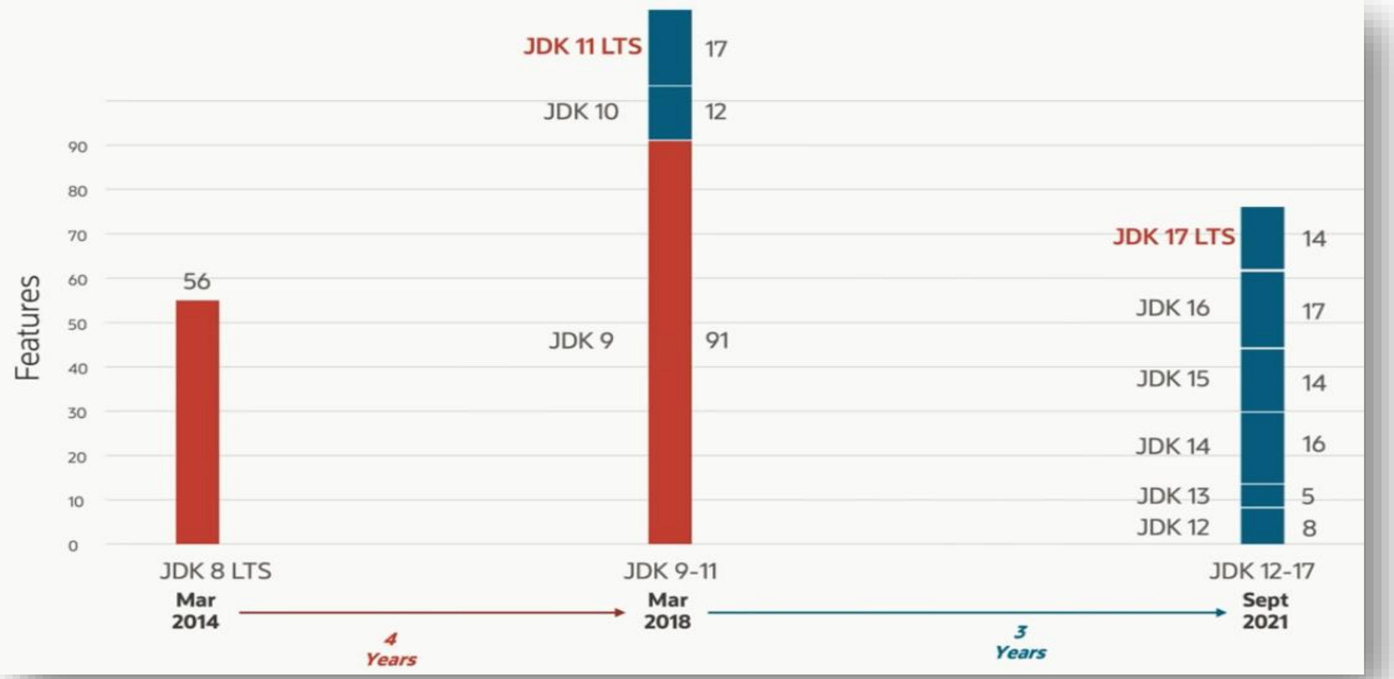
The number of JEPs in JDKs 9-19



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

LTS (Long Term Support) Releases

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



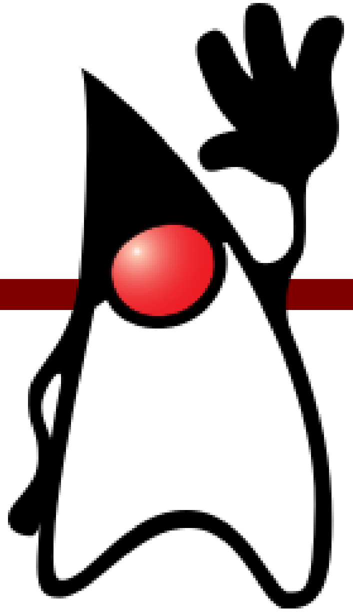
Is Java Really "Free"?

- 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**
  - Without any further security patches and bug fixes
- **Oracle JDK 11-16** under **Oracle Technology Network (OTN)** license in **production** used with **commercial Java SE subscription**
  - Completely free JDK 11-16 are only OpenJDK binaries
- **Oracle JDK 17+** with **Oracle No-Fee Terms and Conditions (NFTC)** licensing
  - Oracle JDK permits **free use** for all users, even commercial and production use

Current State of Java?



Some questions for all of us:

- Still using **Java 8** (2014)?
- Switched to the "**old**" **LTS** version **Java 11** (2018)?
- Upgrading to the **latest LTS** version **Java 17** (2021)?
- Using 6-month releases of **Java 19** (or **12-16**)?

Some Important Features

Important features in Java 8-19:

- **Java Platform Module System** and **6-months OpenJDK** releases
- **Memory Management** – various Garbage Collectors and default G1
- **Language Features** – Local-Variable Type Inference (vars), Sealed Classes, Hidden Classes, Records, Switch Expressions, Pattern Matching, Text Blocks …
- **Libraries and APIs** – Foreign-Memory Access API, Vector API, Pseudo-Random Generator, Deserealization Filters…
- **Easier Debugging** – Flight Recorder, JFT Event Streaming, NullPointerExceptions …
- **Modernizing Infrastructure** – ~~Mercurial~~→Git, GitHub, ports …
- **Deprecations & Removals** – ~~CMS GC, Nashorn, Biased Locking, RMI Activation, Applet API, Security Manager~~…

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

New Garbage Collectors

Many GCs to choose from:

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

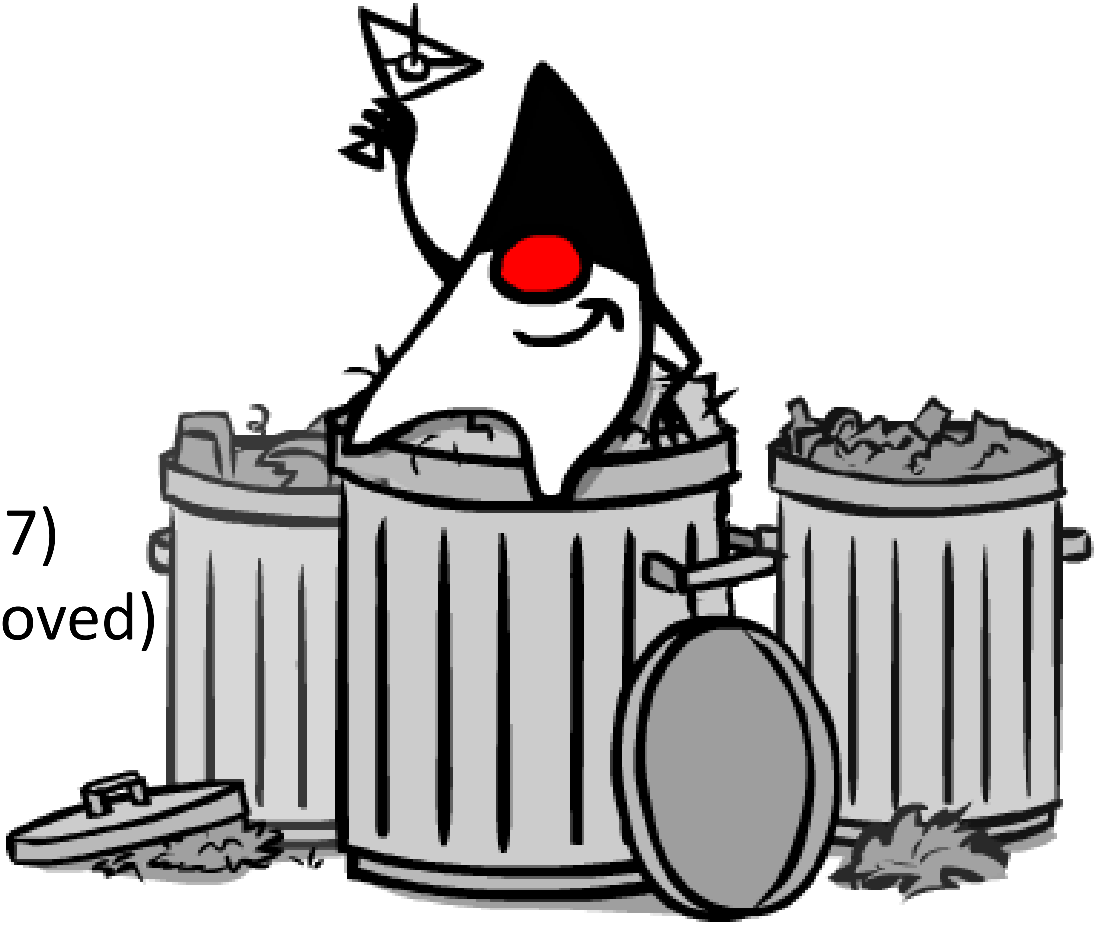
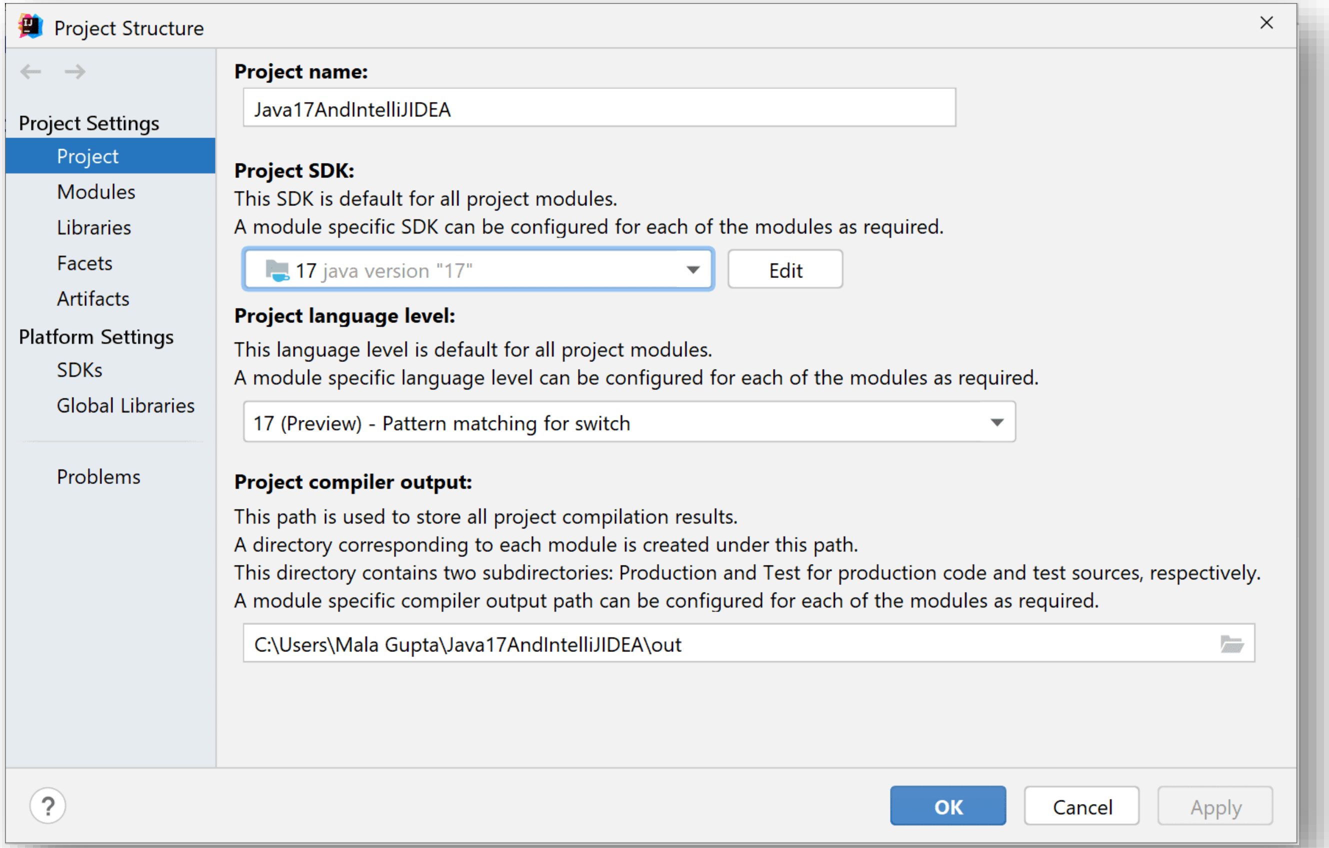


Image Source: 5 Coding Hacks to Reduce GC Overhead, Tall Weiss, OverOps, 2013

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 Structure

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

```
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

Switch Expressions

- 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;
    }
};
```

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();
                         """);
```

Sealed 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

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; }
```

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

@Override
Public double hashCode () {
   return …
}

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

Sometimes data is just … data.

Mark Reinhold

Pattern Matching for instanceof

- JEP 394: **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);
}
```

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();
    };
}
```

JDK 19

- **JDK 19 0n September 20, 2022**
  - **New features** and **APIs** at openjdk.java.net/projects/jdk/19/
  - This release features JEPs that provide continued contribution toward OpenJDKs projects **Amber, Loom,** and **Panama**
- **JEPs** delivered:
  - 405: **Record Patterns** (Preview)
  - 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)
  - 428: **Structured Concurrency** (Incubator)

Foreign Function & Memory API

- JEP 424: **Foreign Function & Memory API** (Preview) openjdk.java.net/jeps/424
- API for statically-typed, pure-Java access to **native code**
  - It evolves from: JEP 419 in JDK 18 and JEP 412 in JDK 17
- Enables Java to call (any) **native libraries** and **process native data** without the brittleness and danger of JNI (Java Native Interface)
- Efficiently invoking foreign functions (outside of JVM) and by safely accessing foreign memory (not managed by JVM)
  - **Foreign Linker** API supports foreign function support
  - **Foreign Memory Access** API allows access to memory outside of heap
- Examples: Carl Dea (Azul) github.com/carldea/panama4newbies

Vector API

- JEP 426: **Vector API** (Fourth Incubator) openjdk.org/jeps/426
- API for **vector computations** that reliably compile at runtime to optimal vector instructions on supported CPU architectures
  - Achieving performance superior to equivalent scalar computations
- Taking advantage of the Single Instruction Multiple Data (SIMD) instructions
  - Enhancements from: JEP 417 in JDK 18, JEP 414 in JDK 1, and JEP 338 in JDK 16
  - Enhances the Vector API to load and store vectors to and from a MemorySegment (JEP 424)
- Allows developers to **write complex vector algorithms** directly in Java

Structured Concurrency

- JEP 428: **Structured Concurrency** (Incubator) openjdk.org/jeps/428
  - The term was coined by Martin Sústrik and popularized by Nathaniel J. Smith
- Simplify multithreaded programming by **structured concurrency API** – treat multiple tasks running in different threads as a single unit of work
  - Streamlining error handling and cancellation, improving reliability, and enhancing observability
- In structured concurrency, **subtasks work on behalf of a task** – the task awaits the subtasks' results and monitors them for failures
- The power of structured concurrency for multiple threads comes from two ideas:
  a) well-defined **entry and exit points** for the flow of execution through a block of code, and
  b) a strict **nesting of the lifetimes of operations** mirroring their syntactic nesting in the code

Virtual Threads (Preview)

- JEP 425: **Virtual Threads** (Preview)
- Lightweight threads that dramatically reduce effort of writing, maintaining, and observing high-throughput concurrent applications
- Example:

```
public class Main {
    public static void main(String[] args)
      throws InterruptedException {
        var vThread = Thread.startVirtualThread(() -> {
            System.out.println("Hello from virtual thread");
        });
        vThread.join();
    }
}
```

Virtual Threads (Preview)

- Examples:
  - Nicolai Parlog github.com/nipafx/loom-lab
  - Bazlur Rahman github.com/rokon12/project-loom-slides-and-demo-code
- To watch:
  - **Java 19 Virtual Threads** - JEP Café #11 youtu.be/lKSSBvRDmTg
  - **Virtual Thread Deep Dive** - Inside Java Newscast #23 https://youtu.be/6dpHdo-UnCg
  - **Launching 10 millions virtual threads with Loom** - JEP Café #12 youtu.be/UVoGE0GZZPI
  - **Java Asynchronous Programming Full Tutorial with Loom and Structured Concurrency - JEP Café #13** youtu.be/2nOj8MKHvmw

Record **Patterns**

- JEP 405: **Record Patterns** (Preview) openjdk.org/jeps/405 – deconstruct record values
  - Based on **Pattern Matching for switch**
  - In JDK 16 via JEP 395 we had **record classes** as transparent carriers for data
- **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

Record Patterns

- *Example*: pattern variable *p* is used only to invoke the methods x() and y()

```
record Point(int x, int y) {}
static void printSum(Object o) {
    if (o instanceof Point p) {
        int x = p.x();
        int y = p.y();
        System.out.println(x+y);
    }
}
```

- The pattern could not only test whether a value is an instance of Point, but also **extract** the x and y components from the value directly

```
record Point(int x, int y) {}
void printSum(Object o) {
    if (o instanceof Point(int x, int y)) {
        System.out.println(x+y);
    }
}
```

Tooling Support for JDK 17-19

- Timely support for new features by **IDE tools** helps 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 17-19 support today within:
  - **JetBrains IntellliJ IDEA 2022.2.3**
  - **Eclipse IDE 2022-09 (4.25)** via a marketplace solution
  - **Visual Studio Code**
  - **NetBeans 15** with support for JDK 17
- For preview features use:
  - Compile with: **`javac --release 19 --enable-preview Main.java`**
  - Run with: **`java --enable-preview Main`**

JDK 20 – Foreseeable Future

- **JDK 20** to be released in **March 2023**
  - **New features** and **APIs** at https://openjdk.org/projects/jdk/20/
  - **Build 17 available**
- **JEPs** targeted: **No** JEPs **targeted** or **integrated** for inclusion in JDK 20 at this time
- However, we think the following JEPs have the potential to be included:
  - JEP 401, Primitive Classes (Preview)
  - JEP 429, Extent-Local Variables (Incubator)
  - JEP Draft 8277163, Value Objects (Preview)
  - JEP Draft 8273943, String Templates (Preview)
  - JEP Draft 8280836, Sequenced Collections
  - JEP Draft 8284289, Asynchronous Stack Trace VM API
  - JEP Draft 8283227, JDK Source Structure
  - JEP Draft 8280389, ClassFile API
  - JEP Draft 8278252, JDK Packaging and Installation Guidelines

Projects – Longer-term Java Future

OpenDJK Projects – 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

New JEPs

- JEP 429: **Extent-Local Variables** (Incubator) openjdk.org/jeps/429
  - Introduce **extent-local variables**, for sharing of immutable data within and across threads
  - Preferred to thread-local variables, especially when using large numbers of virtual threads
- JEP draft: **Value Objects** (Preview) openjdk.org/jeps/8277163
  - Enhance Java object model with **value objects** – class instances that have only **final** instance fields and **lack object** identity
- JEP 401: **Primitive Classes** (Preview) openjdk.org/jeps/401
  - Support new, **developer-declared primitive types** in Java
  - Introduces **primitive classes**, special kinds of value classes that **define** new primitive types

New JEPs – String Templates

- JEP 430: **String Templates** (Preview) openjdk.org/jeps/430
  - String templates are similar to string literals but contain **embedded expressions**
  - **Interpreted at run time** by replacing each expression with the result of evaluating that expression, possibly after further validation and transformation
  - Making it easy to express **strings that include values computed at run time**
- Example:

```
String name = "Joan";
String info = STR."My name is \{name}";
assert info.equals("My name is Joan");   // true
```

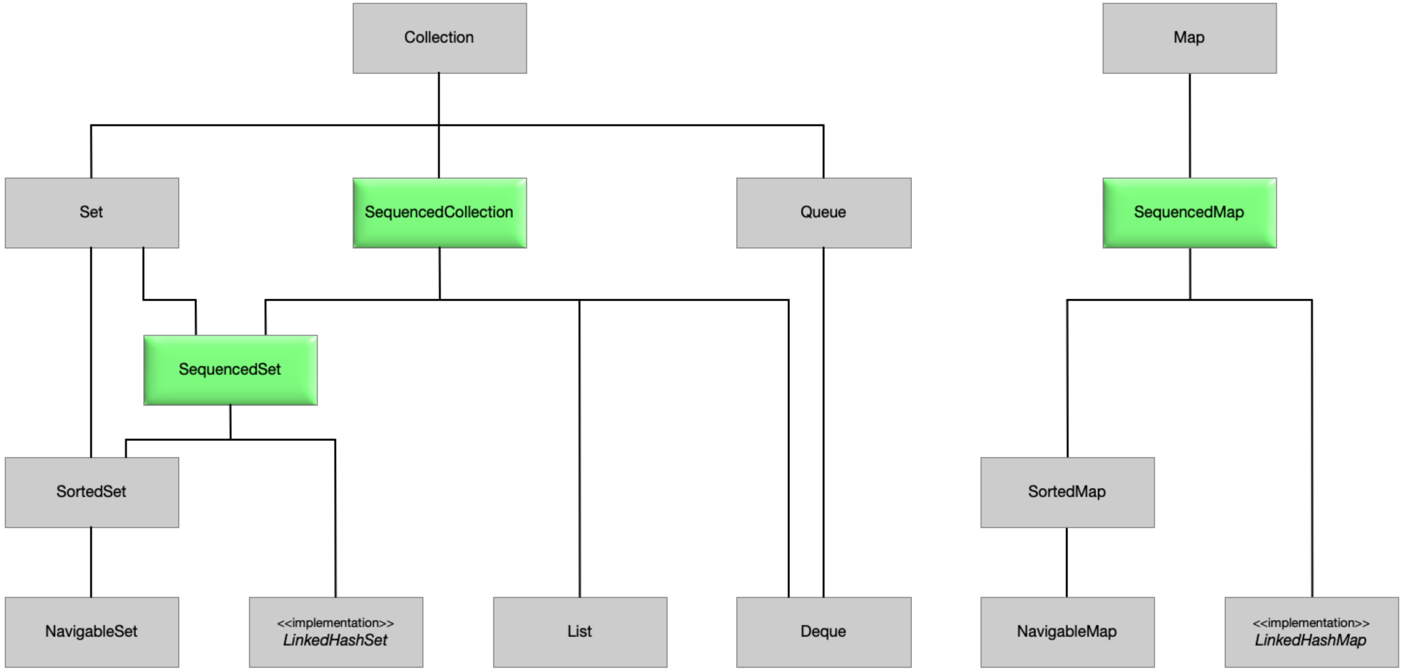
- The string template expression `STR."My name is \{name}"` consists of:
  - Template processor (`STR`);
  - Template (`"My name is \{name}"`) containing an embedded expression (`\{name}`)

New JEPs – Sequenced Collections

- JEP draft: **Sequenced Collections** openjdk.org/jeps/8280836
  - Introduce new interfaces to represent **collections** with a defined encounter **order** (well-defined from first up to the last element)
  - Also provides uniform **APIs for accessing** its first and last elements, and for processing its elements in reverse order
- Examples:
  - List and Deque both define an encounter order but their common supertype is Collection, which does not
  - Set does not define an encounter order, and implementations such as HashSet do not define one, but subtypes such as SortedSet and LinkedHashSet do

New JEPs – Sequenced Collections

- **Sequenced collection's** elements have a defined encounter order
  - First and last elements, all elements have successors and predecessors
  - Supports common operations at either end, and processing forward and reverse
- **Sequenced set** is a Set, a sequenced collection with no duplicate elements
- **Sequenced map** is a Map whose entries have a defined encounter order
- Interfaces for List, Deque, LinkedHashSet, SortedSet, LinkedHashMap, SortedMap …



New JEPs

- JEP draft: **JDK Source Structure** openjdk.org/jeps/8283227
  - Informational JEP describes the overall layout and structure of the JDK source code and related files in the JDK repository
- JEP draft: **Classfile API** openjdk.org/jeps/8280389
  - API for parsing, generating, and transforming Java class files
  - Internal replacement for ASM (all purpose Java bytecode manipulation and analysis framework) in JDK, to be later opened as a public API
- JEP draft: **JDK Packaging and Installation Guidelines** openjdk.org/jeps/8278252
  - Guidelines for creating JDK installers on Linux, Windows, and macOS to reduce risks of collisions between JDK installations by different JDK providers
  - Formalizing installation directory names, package names, and other elements that may lead to conflicts

Unnamed Local Variables and Patterns

- JEP Draft JDK-8294349, **Unnamed Local Variables and Patterns**
  - A new JEP Draft submitted by Angelos Bimpoudis
  - Under the auspices of Project Amber, it proposes to "*enhance the Java language with **unnamed local variables**, which can be initialized but not read from, and with **unnamed patterns**, which match everything and bind nothing. Both are denoted with the underscore (_) token*."
  - Example:

```
HashMap<Integer, String> m = new HashMap<Integer, String>();
...
String _ = m.remove(42);
String _ = m.remove(43);
```

  - More: bugs.openjdk.org/browse/JDK-8294349

Paving the on-ramp

- Brian Goetz (Java language architect, Oracle) proposed simplifying the language:
  - "*The question of which program elements are most and least helpful for students first learning Java.*"
- Some ideas:
  1. **A more tolerant launch protocol**
     - Relax the requirement that the class, and main method, be public
     - Make the “args” parameter to main optional
     - Make the static modifier on main optional
  2. **Unnamed classes**
  3. **Predefined static imports**
  - More: openjdk.org/projects/amber/design-notes/on-ramp

Example

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

- Public, static and args are optional

```
class HelloWorld {
    void main() {
        System.out.println("Hello World");
    }
}
```

- Class wrapper for “main classes” optional (unnamed classes)

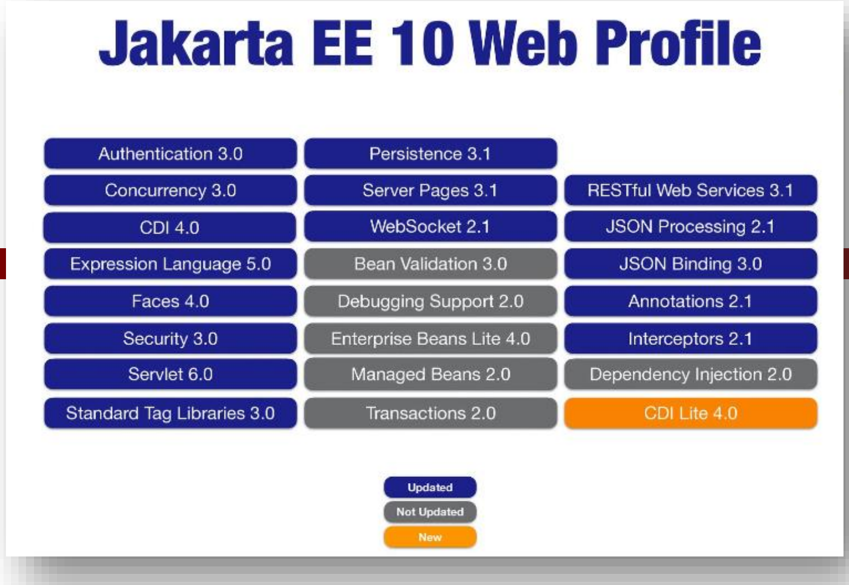
```
String greeting() { return "Hello World"; }

void main() {
    System.out.println(greeting());
}
```

- Automatically static import methods

```
void main() {
    println("Hello World");
}
```

Jakarta EE

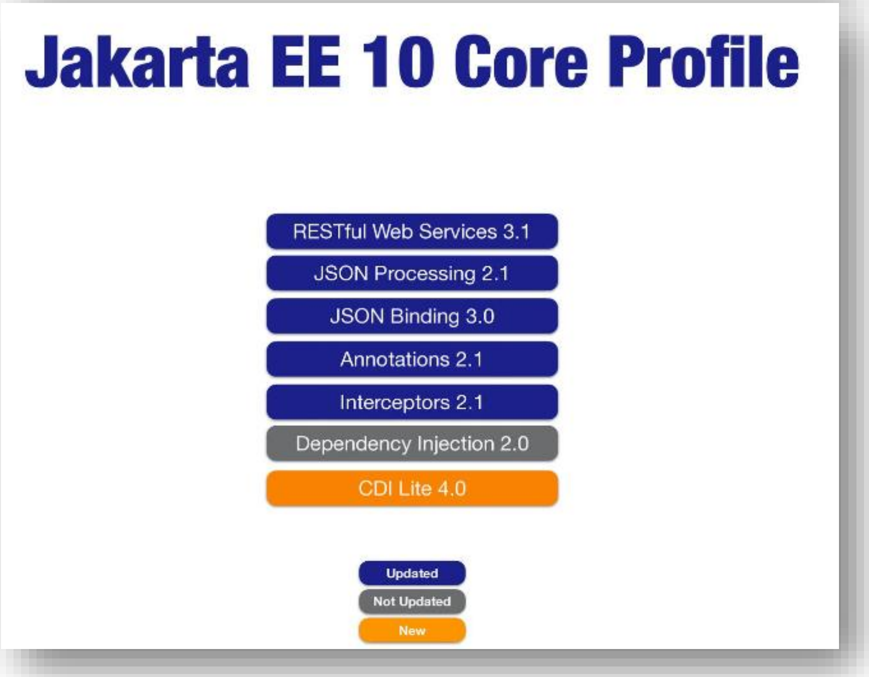
- **Jakarta EE 10 Platform** released on September 22
- **Jakarta EE Web Profile**
- **Jakarta EE Core Profile**
  - A minimal basis for cloud-native runtimes, including runtimes that support build-time applications and microservices
  - New Jakarta CDI Lite 4.0 specification allows compiling to native executables as part of the Jakarta EE Core Profile
- Top priorities from Jakarta EE community:
  1. Native integration with Kubernetes
  2. Better support for microservices
  3. Faster support from existing Java EE / cloud vendors
- Jakarta EE 9/9.1 adoption
  - 36% migrated or plan to adopt, 19% wait for Jakarta EE 10



Jakarta EE 10 Web Profile

Authentication 3.0	Persistence 3.1	
Concurrency 3.0	Server Pages 3.1	RESTful Web Services 3.1
CDI 4.0	WebSocket 2.1	JSON Processing 2.1
Expression Language 5.0	Bean Validation 3.0	JSON Binding 3.0
Faces 4.0	Debugging Support 2.0	Annotations 2.1
Security 3.0	Enterprise Beans Lite 4.0	Interceptors 2.1
Servlet 6.0	Managed Beans 2.0	Dependency Injection 2.0
Standard Tag Libraries 3.0	Transactions 2.0	CDI Lite 4.0

Updated
Not Updated
New



Jakarta EE 10 Core Profile

RESTful Web Services 3.1
JSON Processing 2.1
JSON Binding 3.0
Annotations 2.1
Interceptors 2.1
Dependency Injection 2.0
CDI Lite 4.0

Updated
Not Updated
New

What about **Spring**?

- **Spring** version **6**
  - Spring 6.0.0-M6 and 5.3.23 are available
  - Waiting for RC and GA
- **Spring Boot** version **3**
  - Spring Boot 3 release in November 2022?
  - Will require Java 17
- **Spring Native**
  - Native images

AI and Code

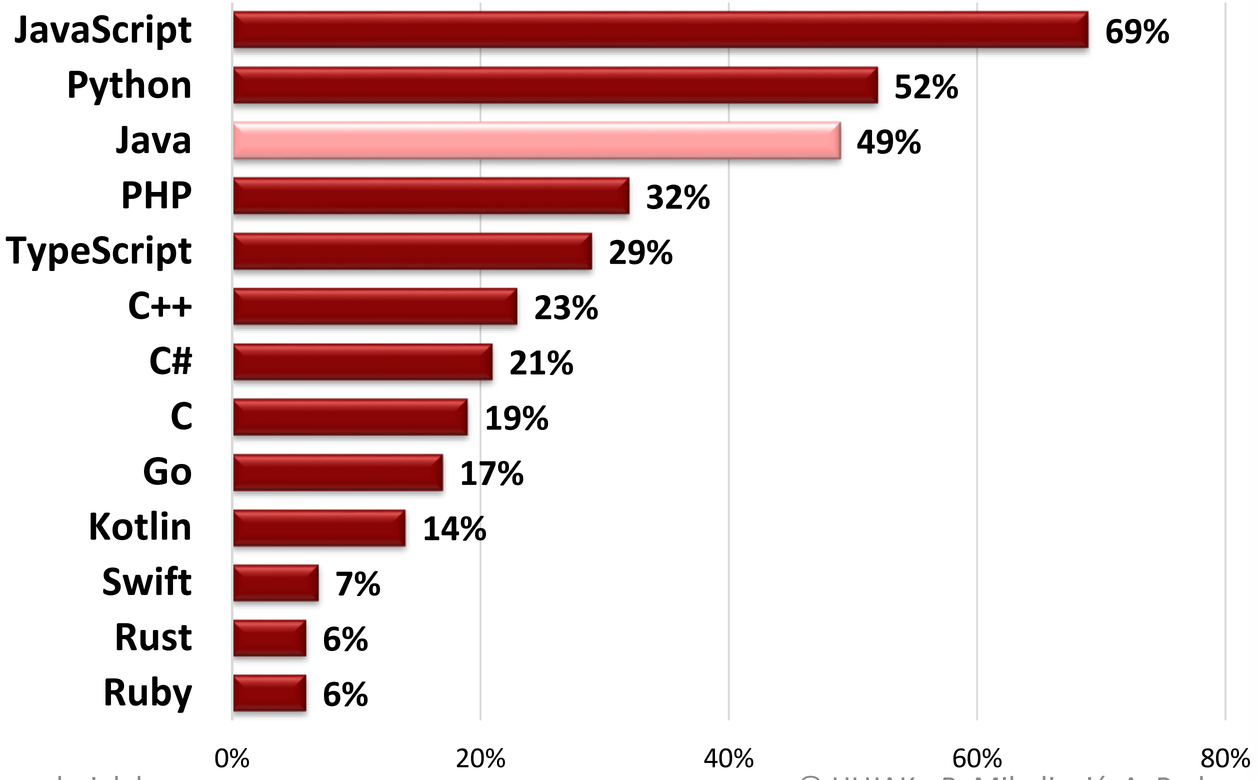
- **AI-generated Source Code**
  - **Predicts your next block of code** delivering accurate code completions
  - Accelerates development by providing **code guidance with patterns** learned from millions of projects
  - **Automates repetitive work** and reduces the need for expensive and distracting code search
  - **Improves code quality** and **consistency** across your project
- **Github Copilot** (copilot.github.com)
  - 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

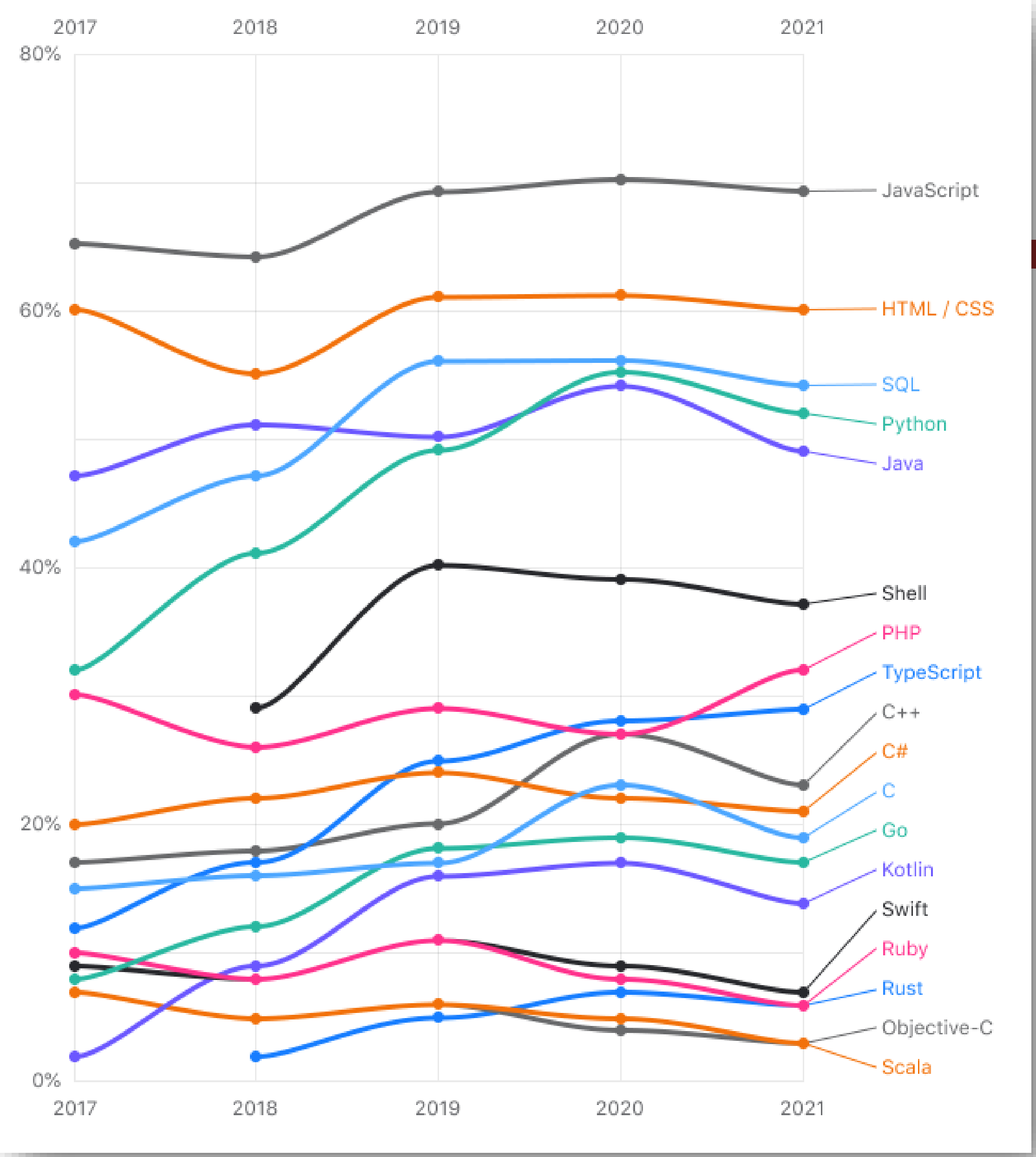
Some **surveys**

- **The State of Developer Ecosystem 2021** by **JetBrains**
  - 31743 developers from 183 countries in 2021
- **2022 State of the Java Ecosystem** by **Eclipse**
  - 1439 developers, March-May 2022
- **The Java Ecosystem 2022** by **Continuum**
  - 200 developers in May 2022
- **2022 Java Developer Productivity Report** by **JRebel**
  - 876 developers in 2022
- **JVM Ecosystem Report 2021** by **Snyk**
  - 2000 Java developers in 2021
- **2021 Developer Survey** by **Stack Overflow**
  - 70000+ developers in May 2022
- **2022 State of the Java Ecosystem** by **New Relic**

Top Languages

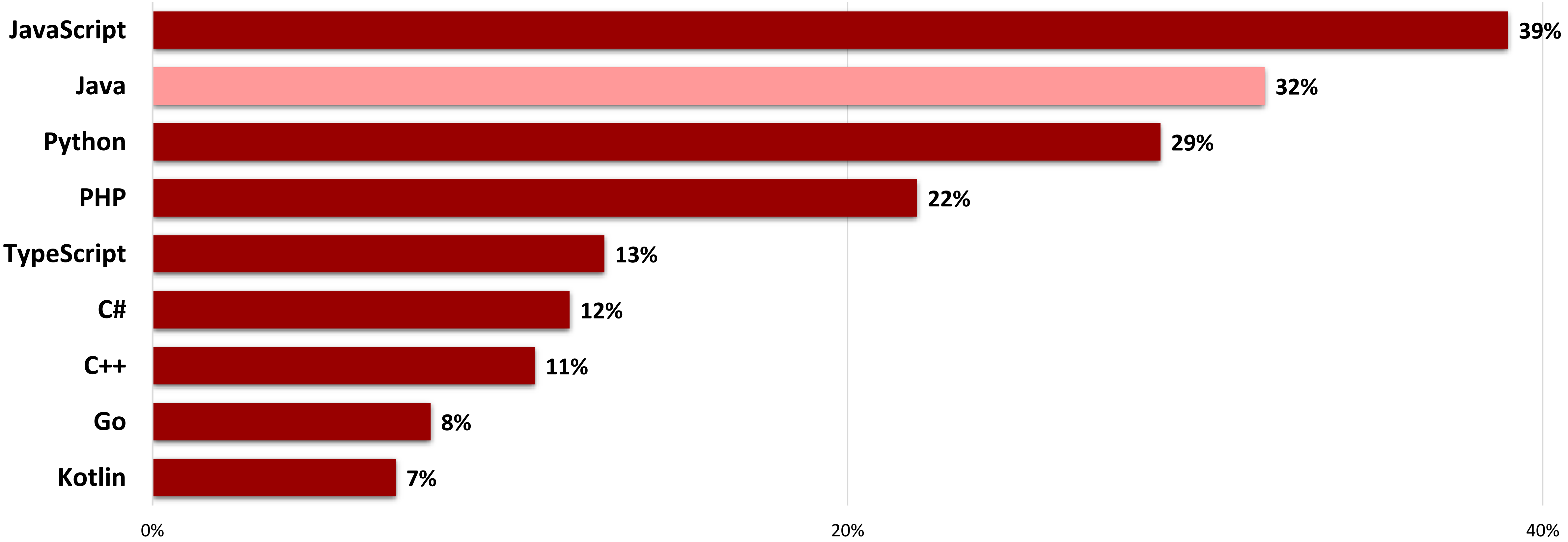
- What programming languages have you used (JetBrains)?





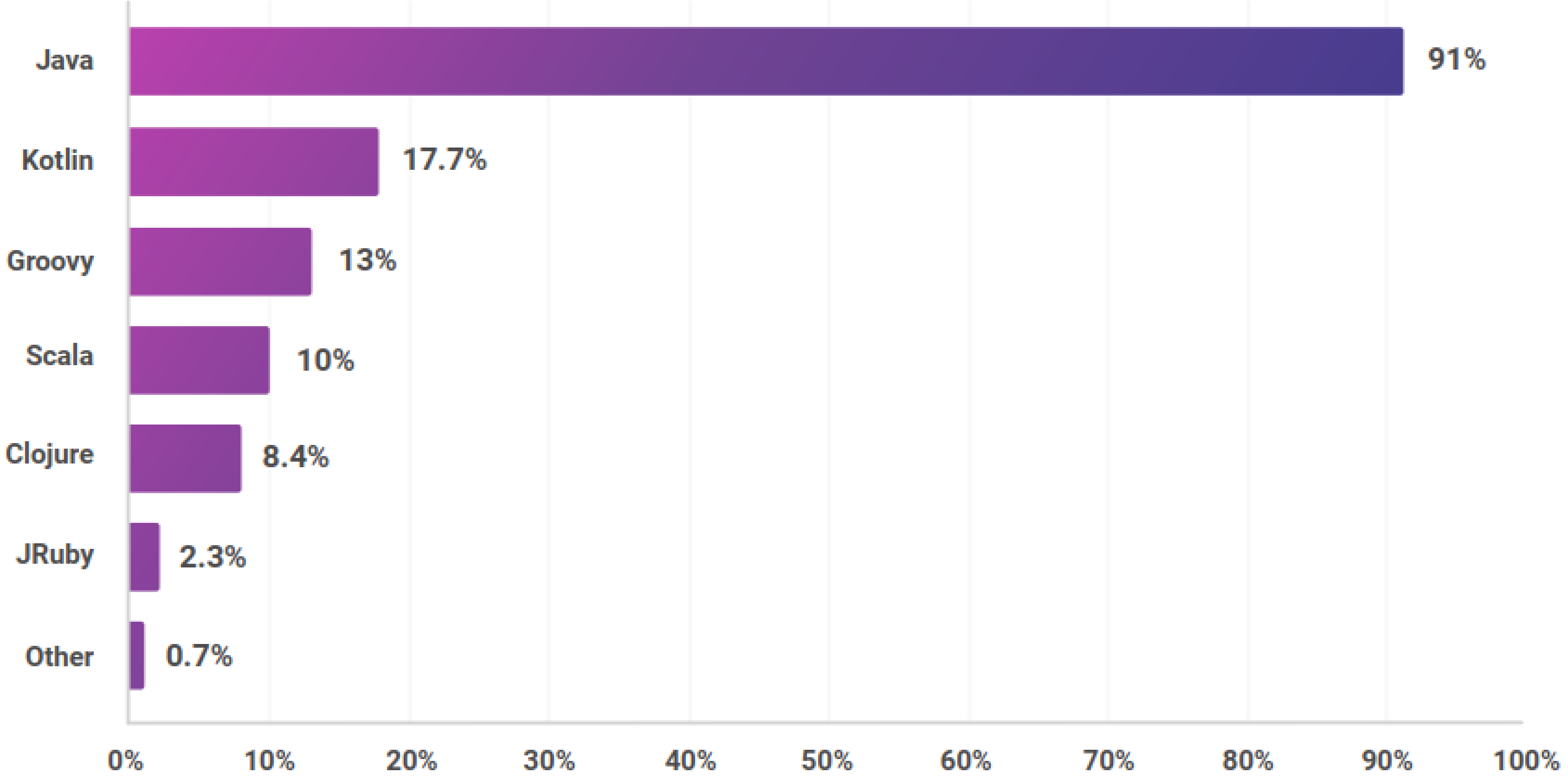
Main Language

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



Other JVM Languages

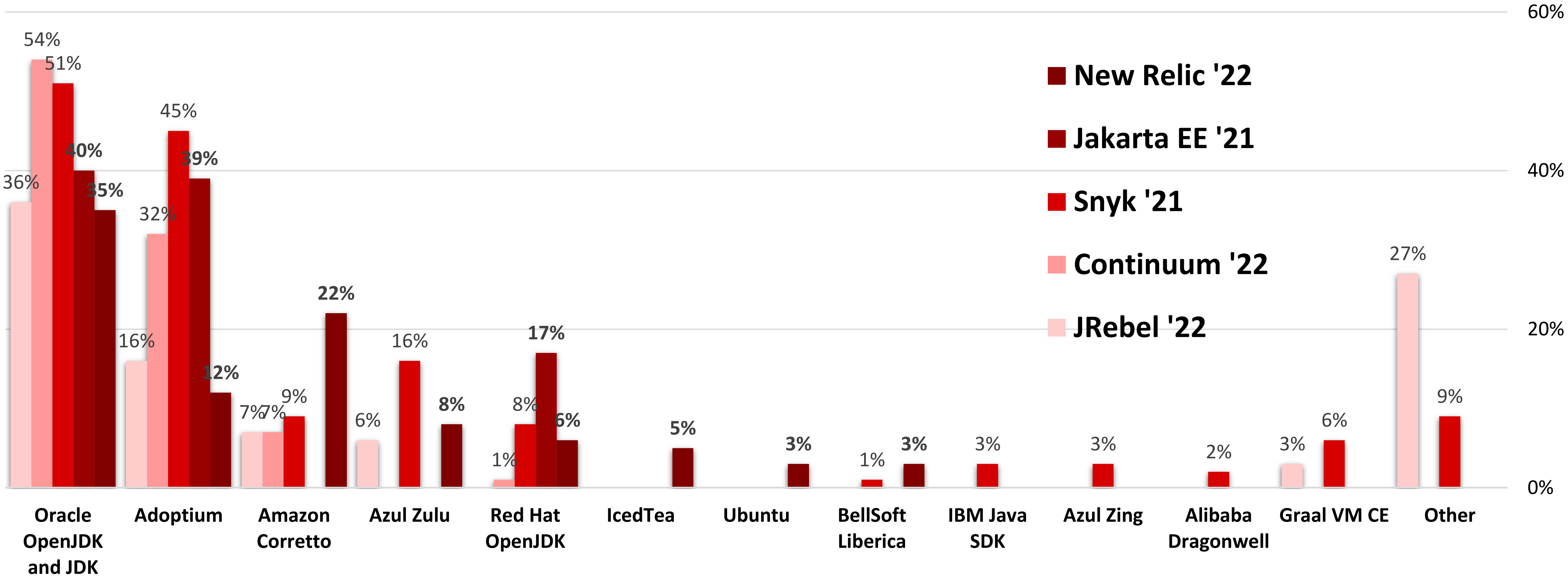
- **JVM languages** used for applications in production (Snyk)



- GitHub repos:
  - Java – 9.5M
  - Kotlin – 531k
  - Scala – 194k
  - Clojure – 76k
  - Groovy – 64k

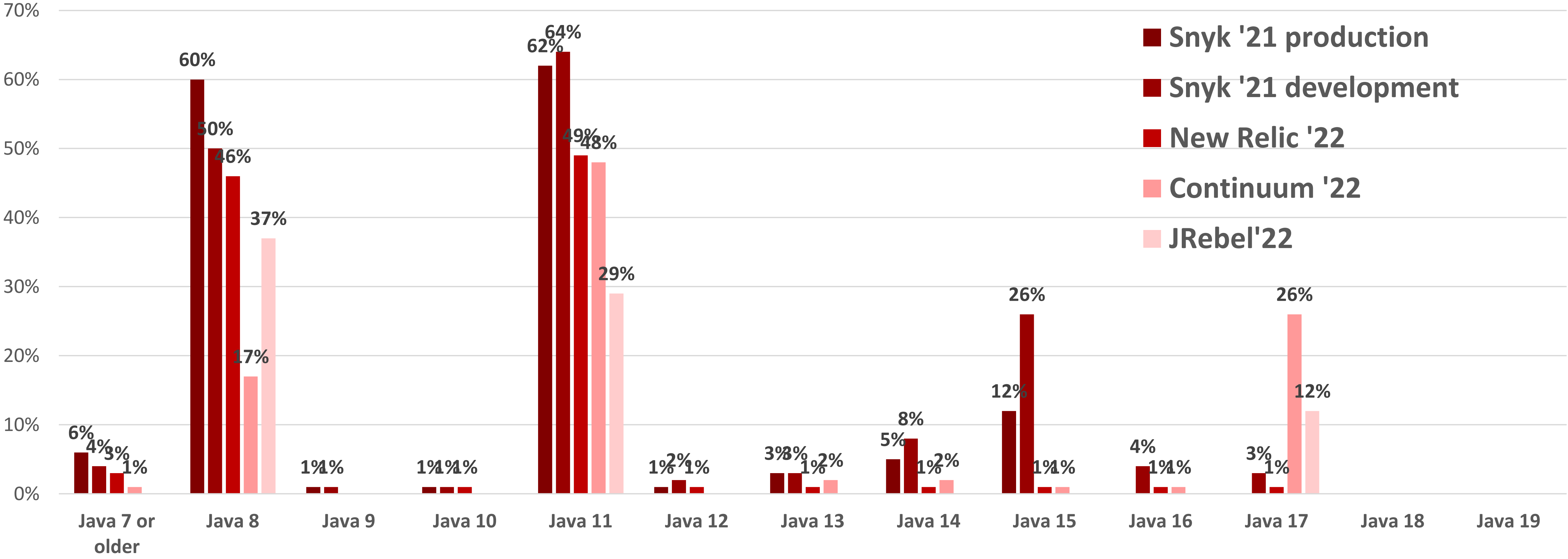
JDK Distributions

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



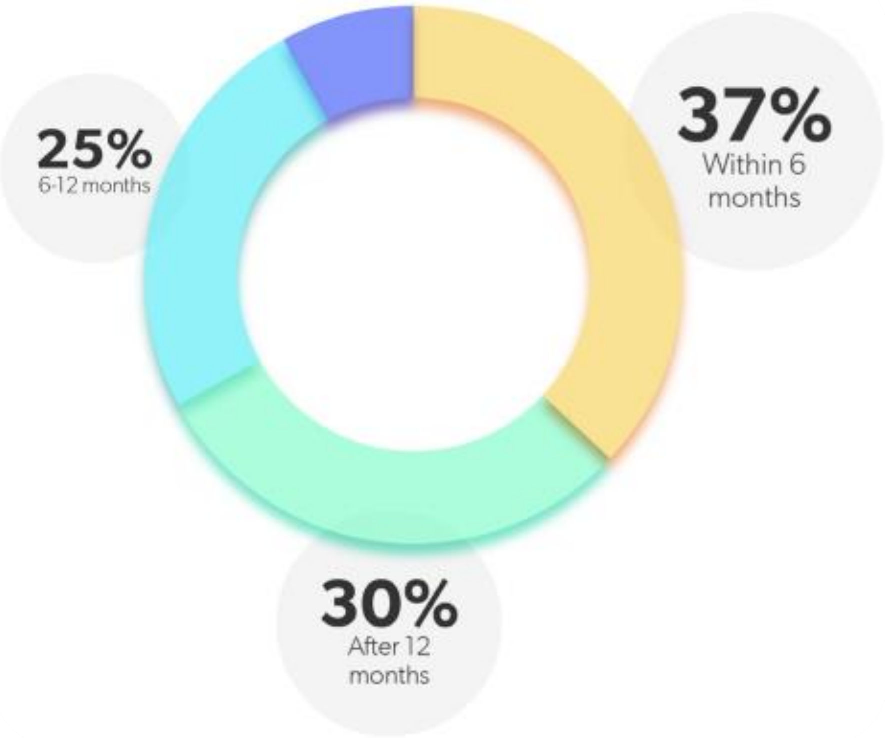
Versions of Java

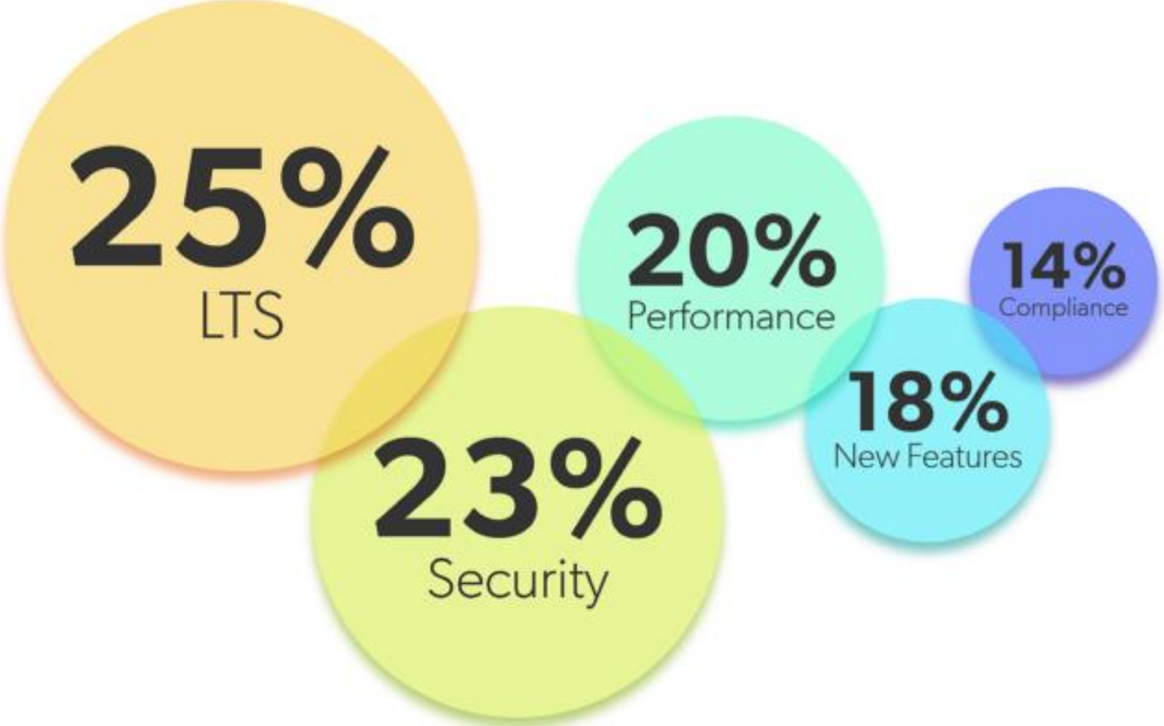
- **Java platform versions** used in projects (in production)



Upgrade to JDK 17

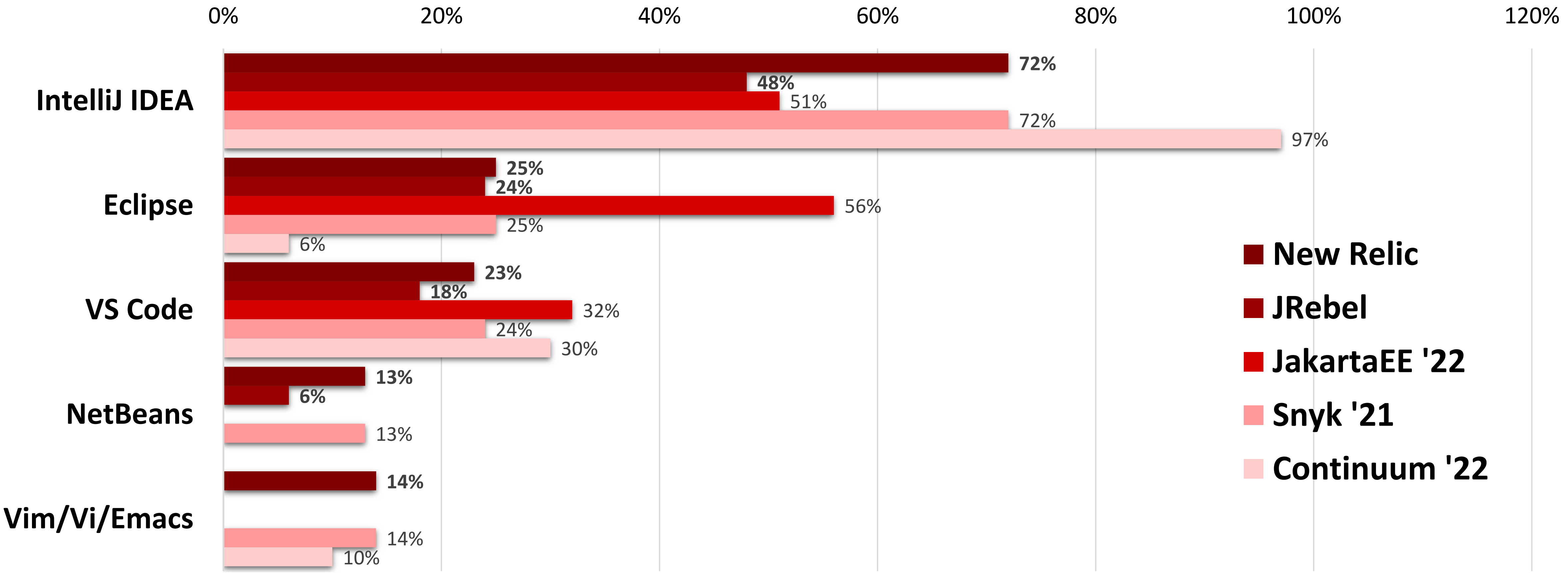
- Which **factors** influence your decision to upgrade JDK Versions? (JRebel)
- When will you upgrade to **JDK 17**? (JRebel)





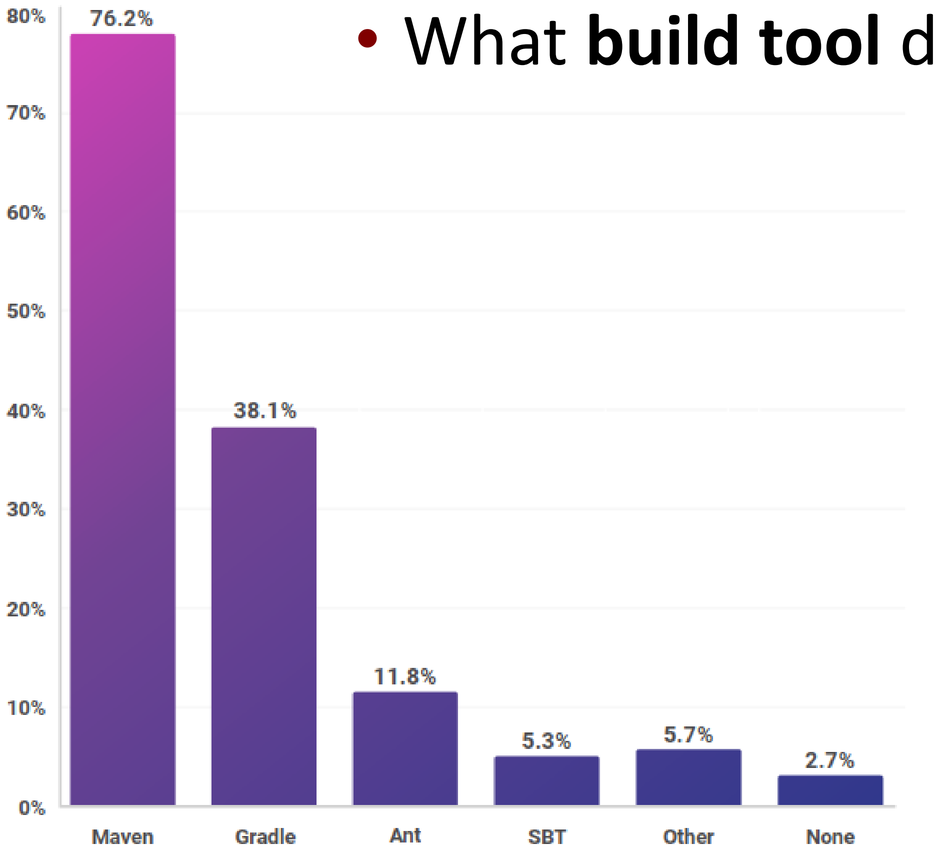
Most popular IDEs

- The most **popular IDEs** used?

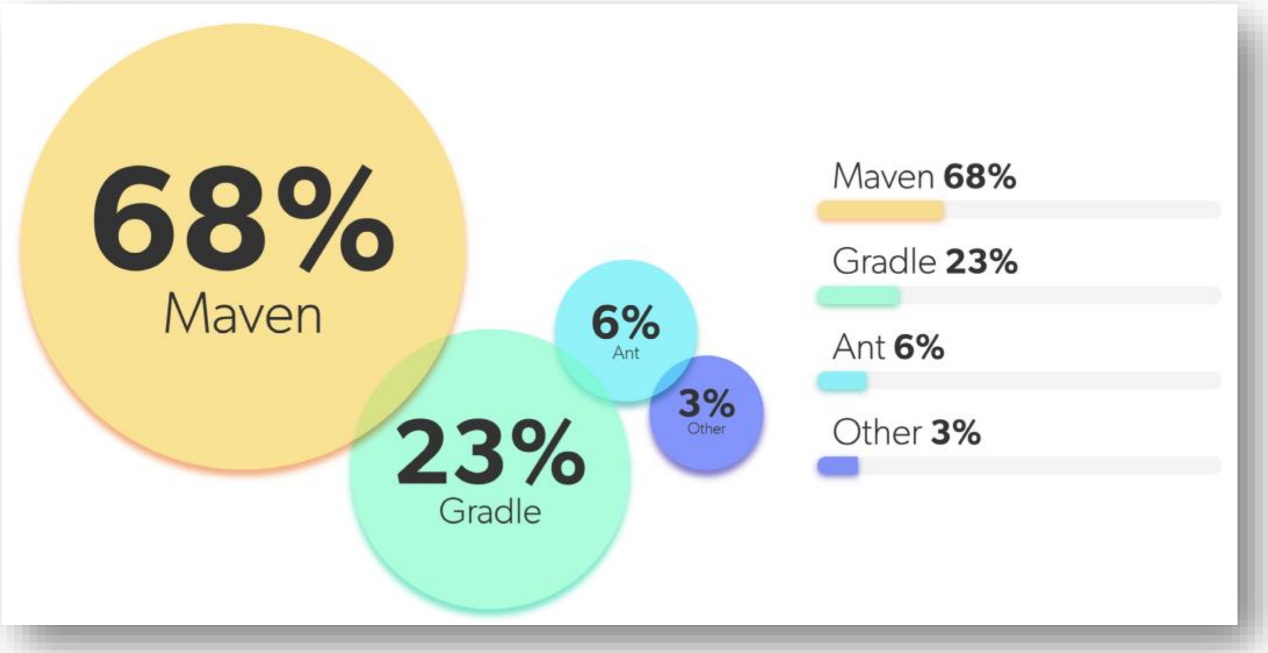


Build Tools

- **Tools for building** applications? (Snyk)

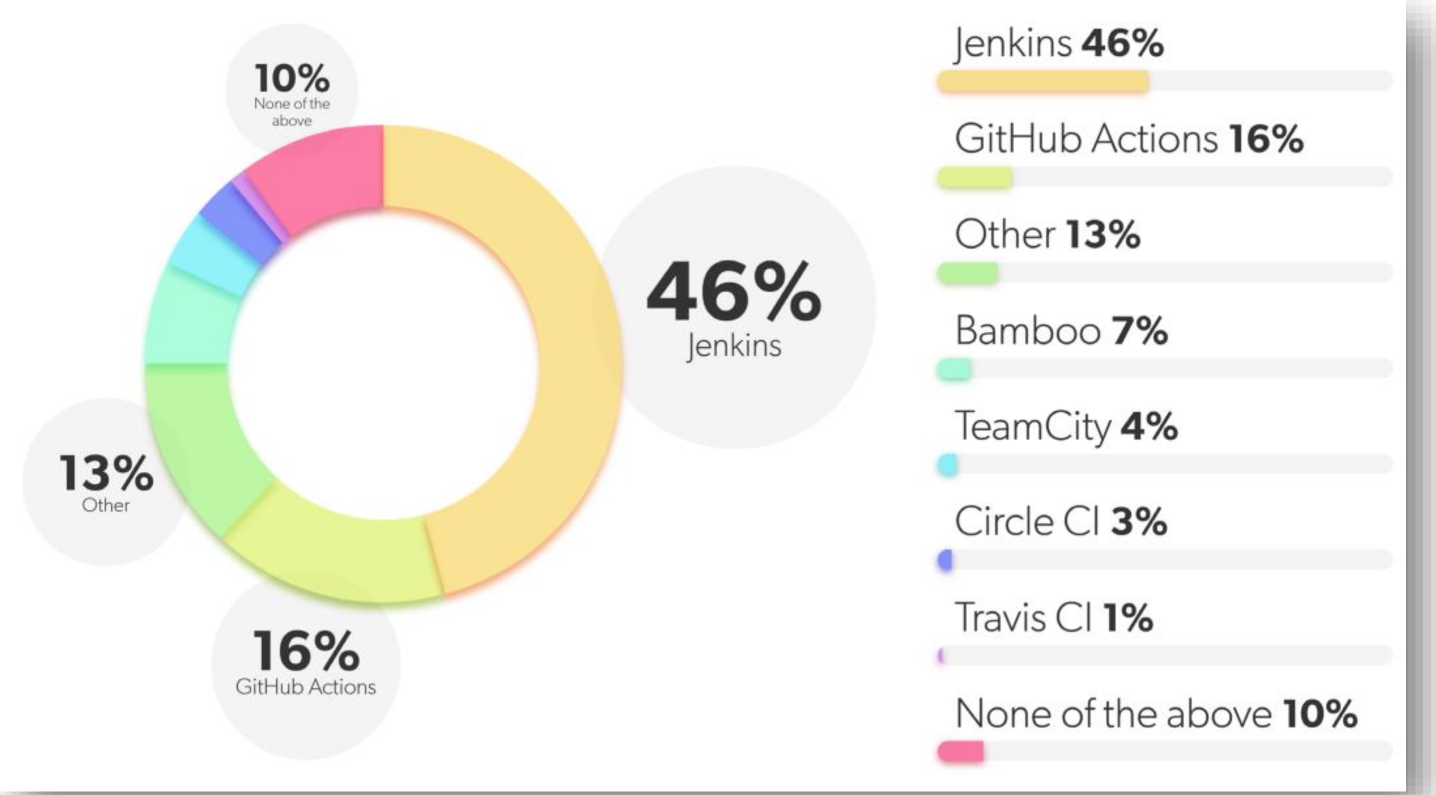


- What **build tool** do you use in your main application? (JRebel)

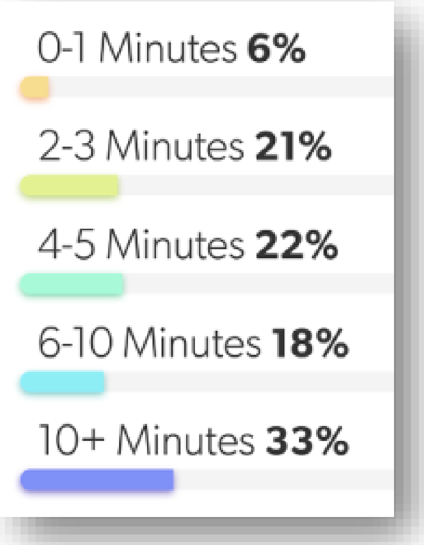


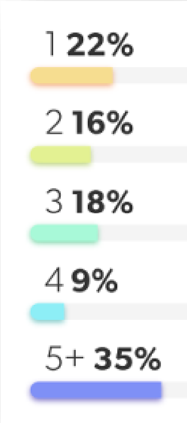
CI/CD

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



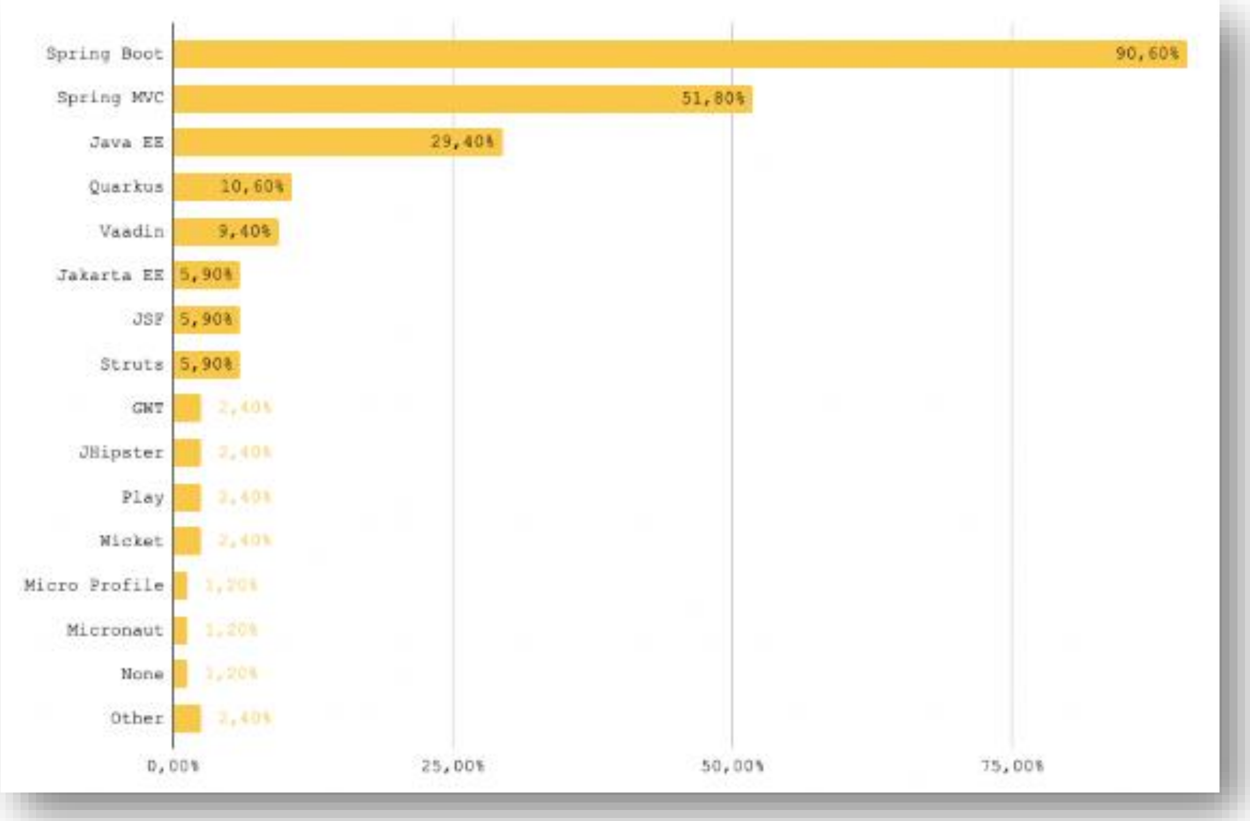
- **How long** does it take to complete CI/CD build?
- Commit **frequency** (per day)?

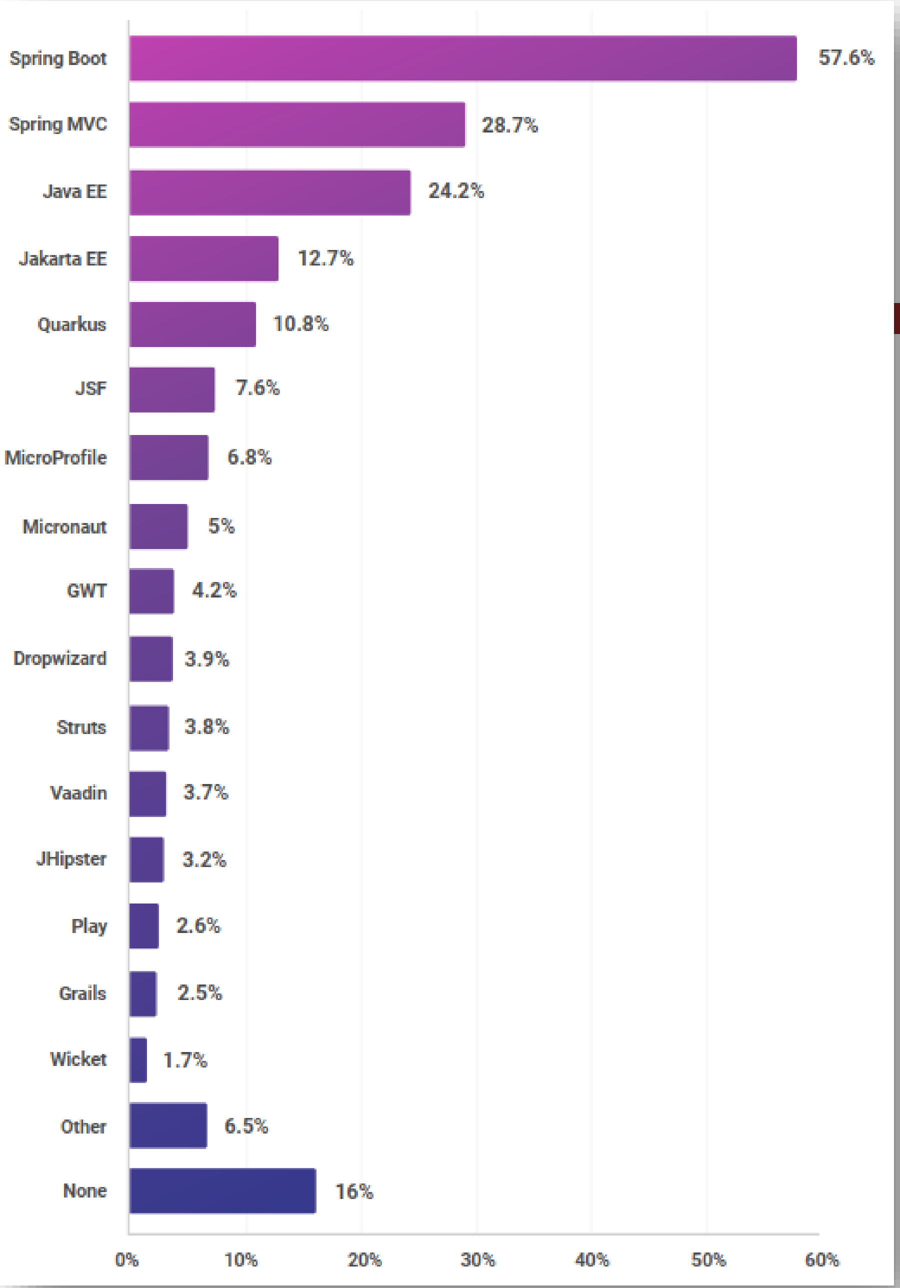




Application frameworks

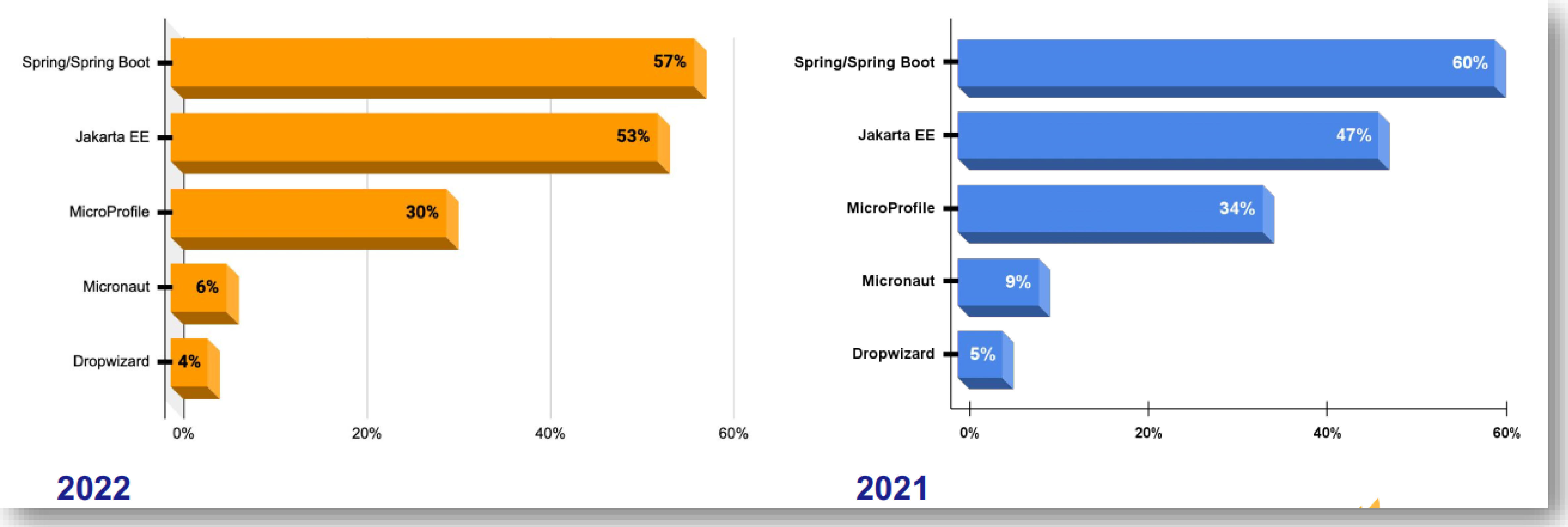
- **App frameworks** (Snyk and Continuum)





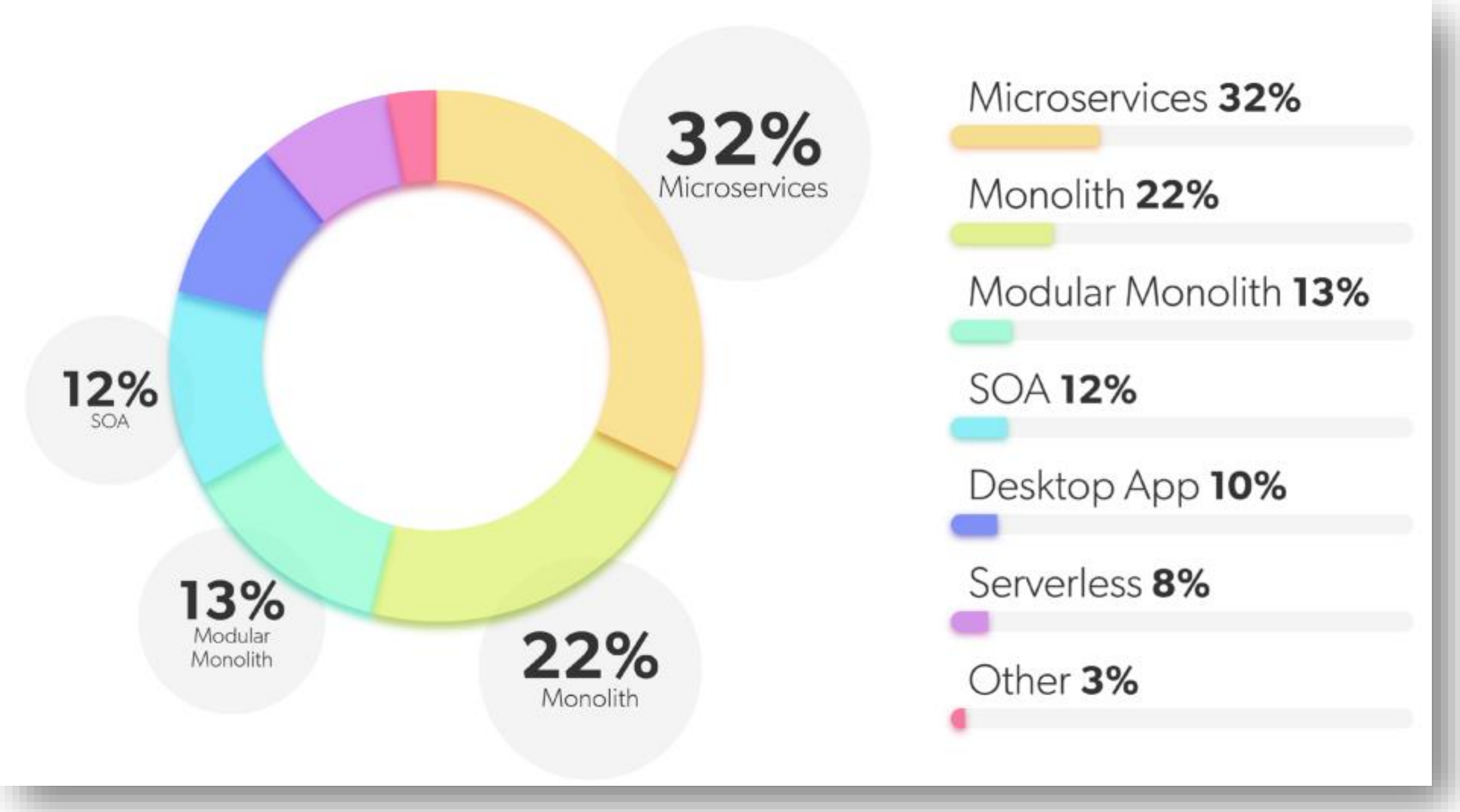
Application frameworks

- **App frameworks** (JakartaEE)



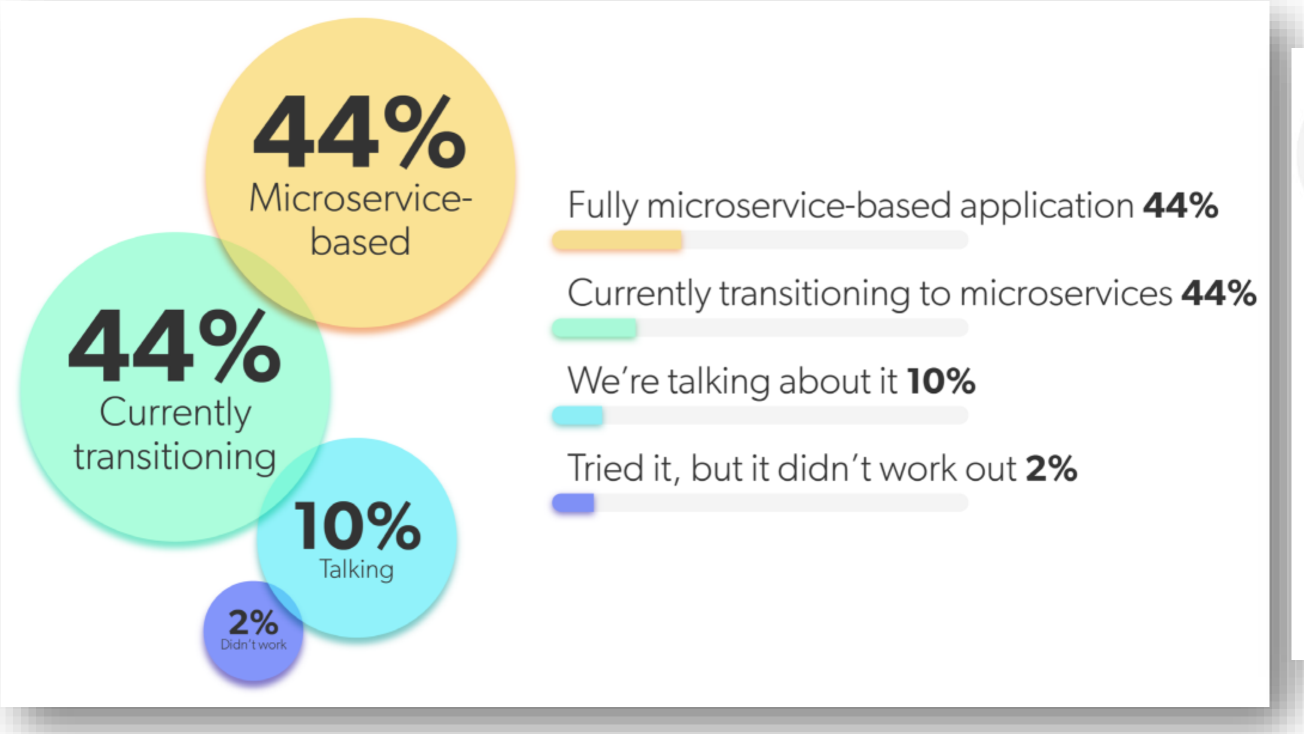
Architecture Trends

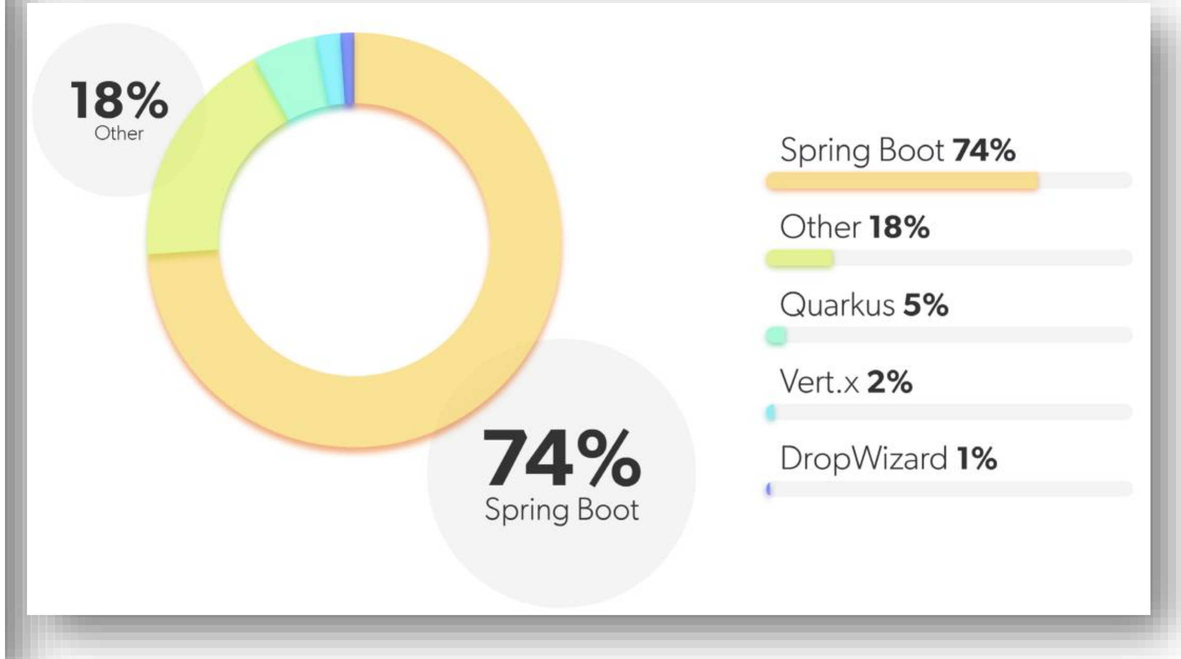
- What is the **Architecture** of the main application you develop?



Microservices

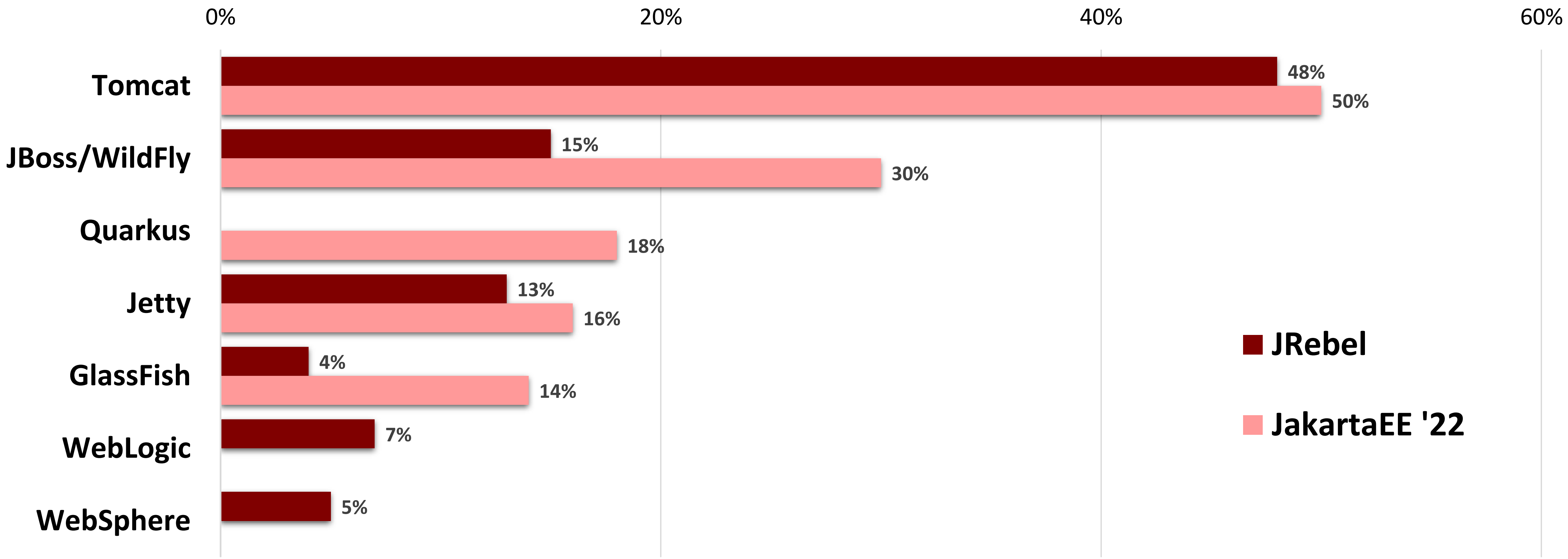
- What is your status for **Microservice** adoption? (JRebel)
- What **Microservice Application Framework** on your main project? (JRebel)





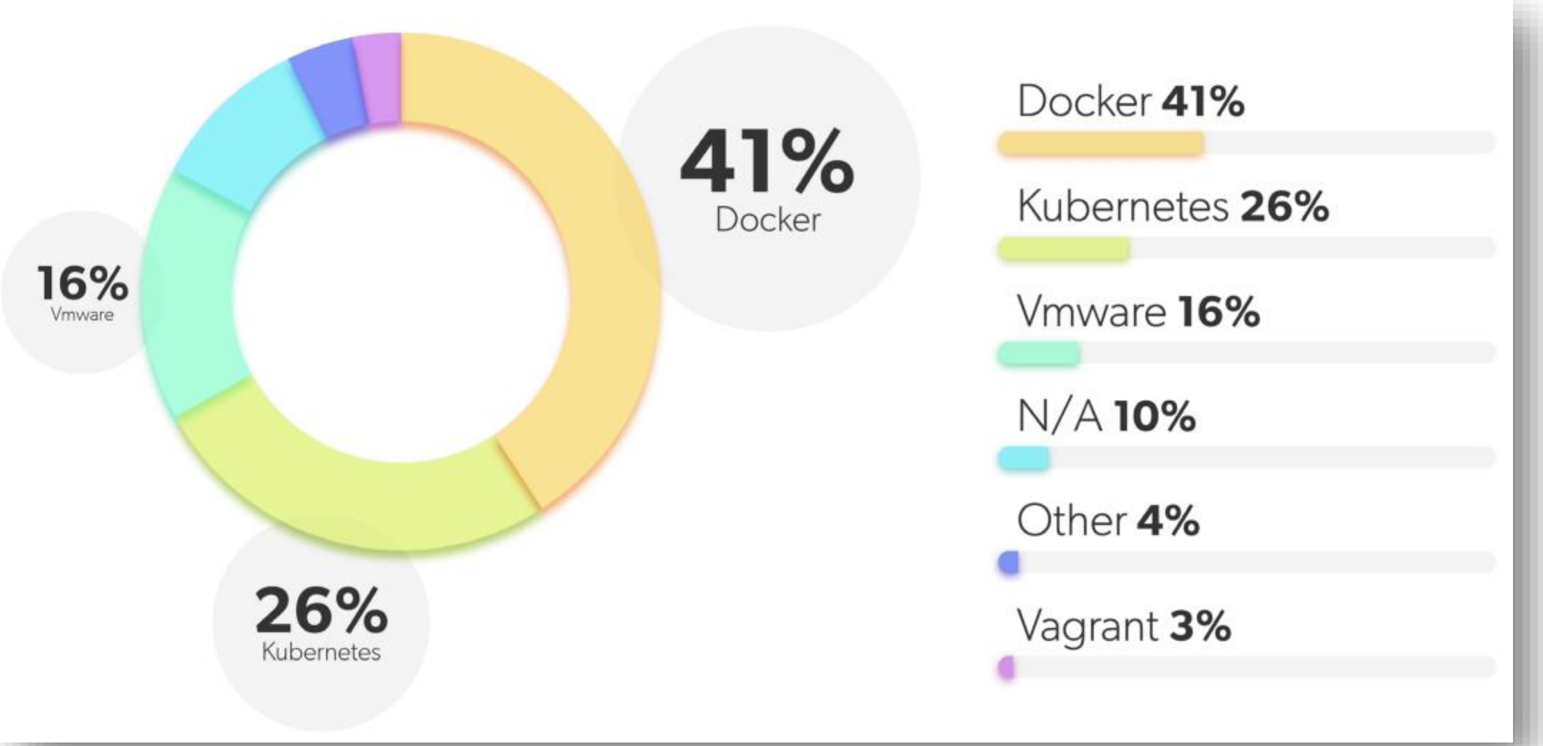
App Servers

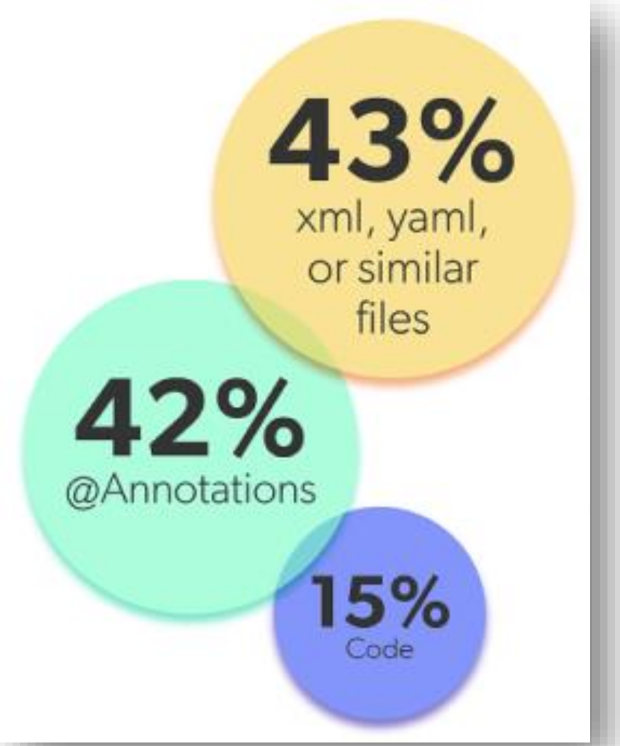
- What **Application Server** do you use on your main application? (JRebel)



Virtual Machines and Framework Configuration

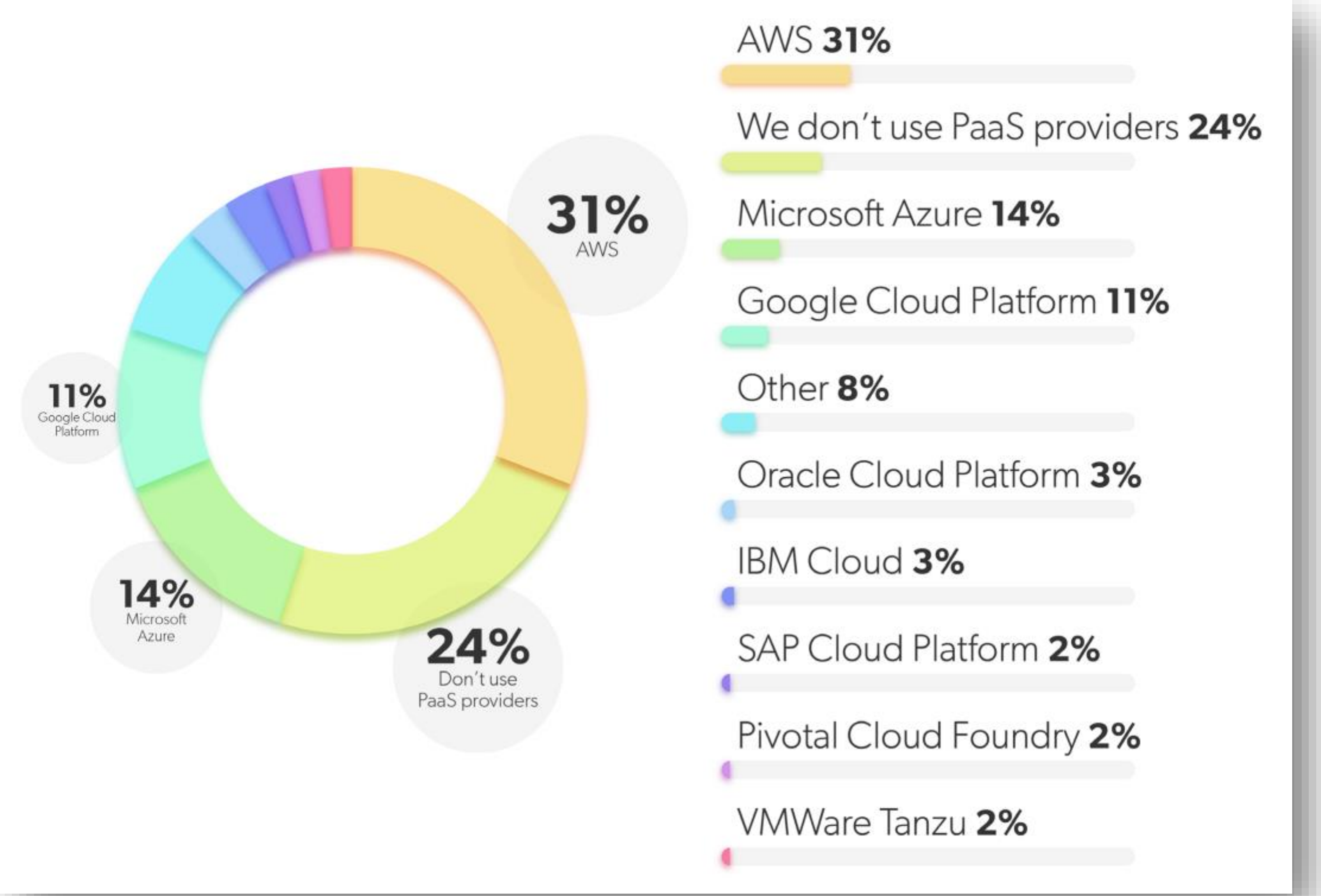
- Which **Virtual Machine Platform** do you use? (JRebel)
  - How do you **configure** most of your frameworks? (JRebel)





PaaS Providers

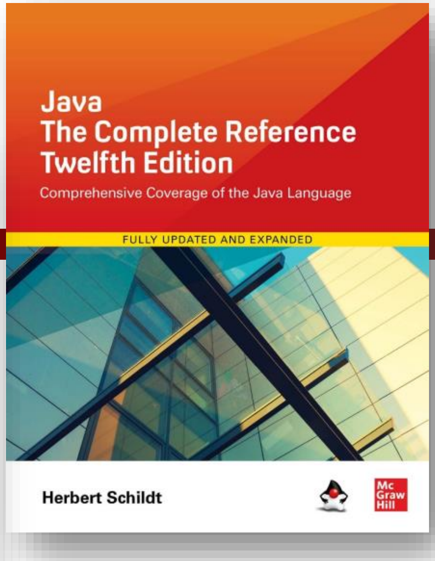
- If you use a platform, who is your **PaaS provider**? (JRebel)

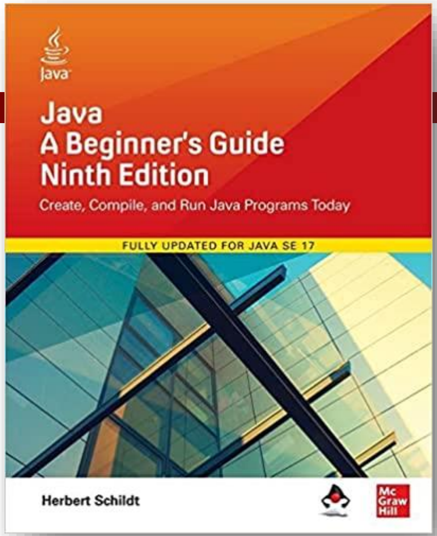


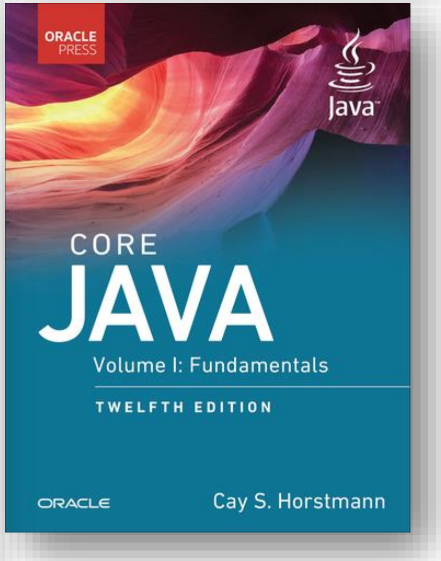
- How to learn?

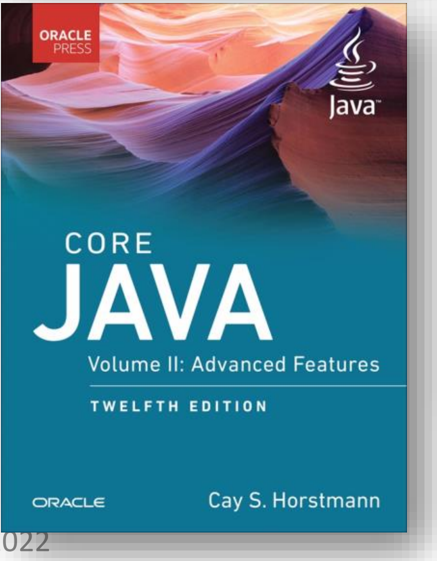
New Books

- **Java: The Complete Reference, 12th ed.**, Herbert Schildt, MGH, November 2021
  - 1280 pages, 45-60 €
- **Java: A Beginner's Guide, 9th ed.**, Herbert Schildt, MGH, January 2022
  - 728 pages, 30-35 €
- **Core Java, Volume I: Fundamentals, 12th ed.**, Cay S. Horstmann, December 2021, Oracle Press
  - 861 pages, approx. 60 €
- **Core Java, Volume II: Advanced Features, 12th ed.**, Cay S. Horstmann, April 2022, Oracle Press
  - 1185 pages, approx. 50 €


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

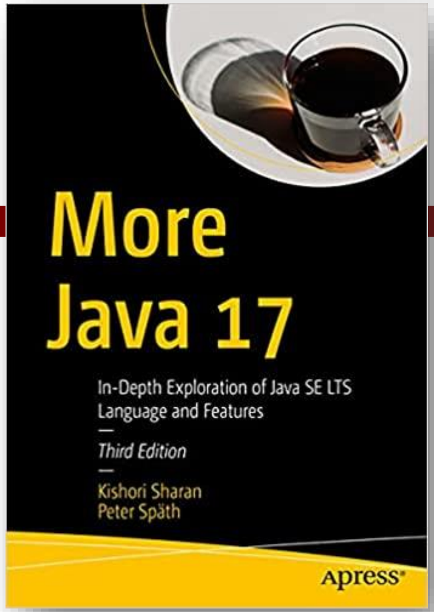

Java
Java
A Beginner's Guide
Ninth Edition
Create, Compile, and Run Java Programs Today
FULLY UPDATED FOR JAVA SE 17
Herbert Schildt
Mc Graw Hill





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 €



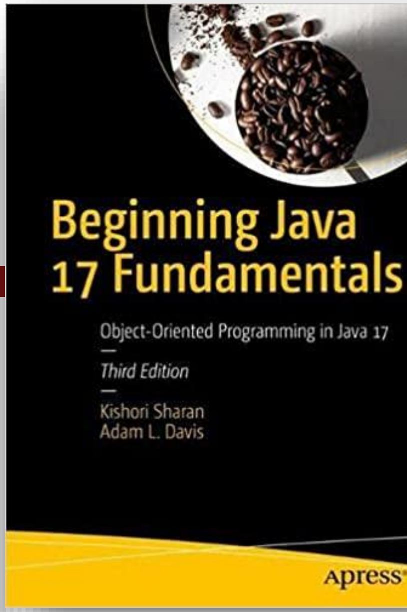
More Java 17

In-Depth Exploration of Java SE LTS Language and Features

Third Edition

Kishori Sharan
Peter Späth

Apress



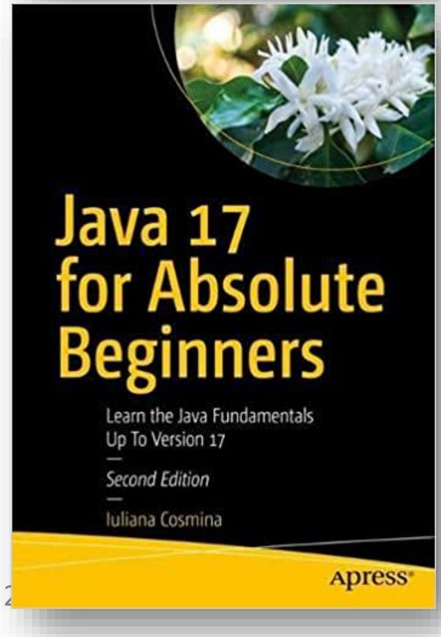
Beginning Java 17 Fundamentals

Object-Oriented Programming in Java 17

Third Edition

Kishori Sharan
Adam L. Davis

Apress



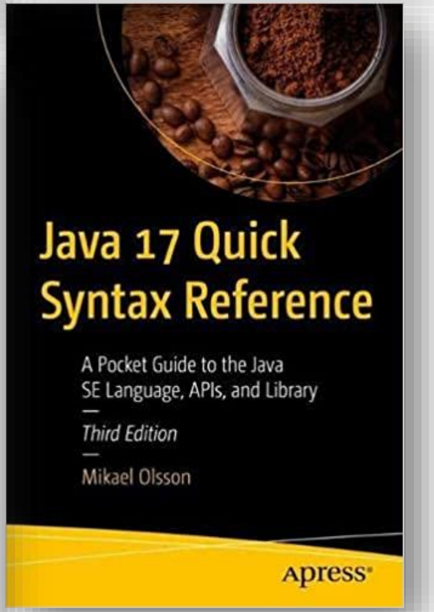
Java 17 for Absolute Beginners

Learn the Java Fundamentals Up To Version 17

Second Edition

Iuliana Cosmina

Apress



Java 17 Quick Syntax Reference

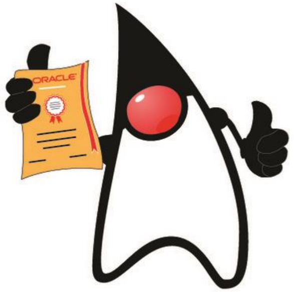
A Pocket Guide to the Java SE Language, APIs, and Library

Third Edition

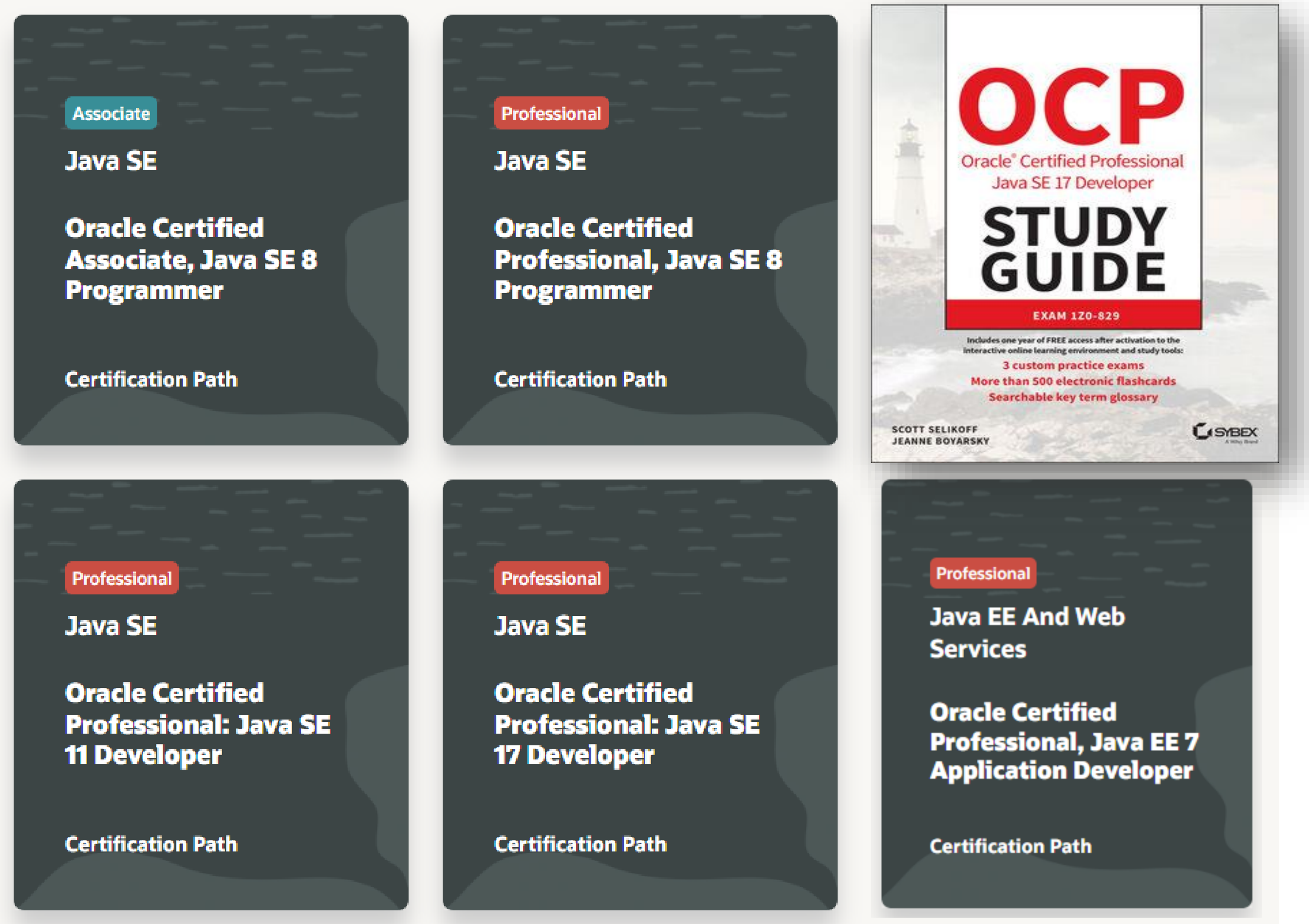
Mikael Olsson

Apress

Java Certification



- **Oracle Certified Professional (OCP): Java SE 17 Developer**
- Oracle Certified Professional (OCP): Java SE 11 Developer
- **OCP Oracle Certified Professional Java SE 17 Developer Study Guide: Exam 1Z0-829,** by Scott Selikoff and Jeanne Boyarsky, May 2022, Wiley
  - 1056 pages, approx. 60 €



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

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

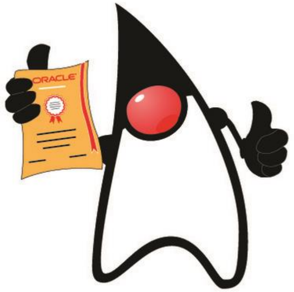
OCP
Oracle® Certified Professional
Java SE 17 Developer
STUDY GUIDE
EXAM 1Z0-829
Includes one year of FREE access after activation to the interactive online learning environment and study tools:
3 custom practice exams
More than 500 electronic flashcards
Searchable key term glossary
SCOTT SELIKOFF
JEANNE BOYARSKY
SYBEX

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%

- But, how much do they (we) earn?

Java Development Salaries

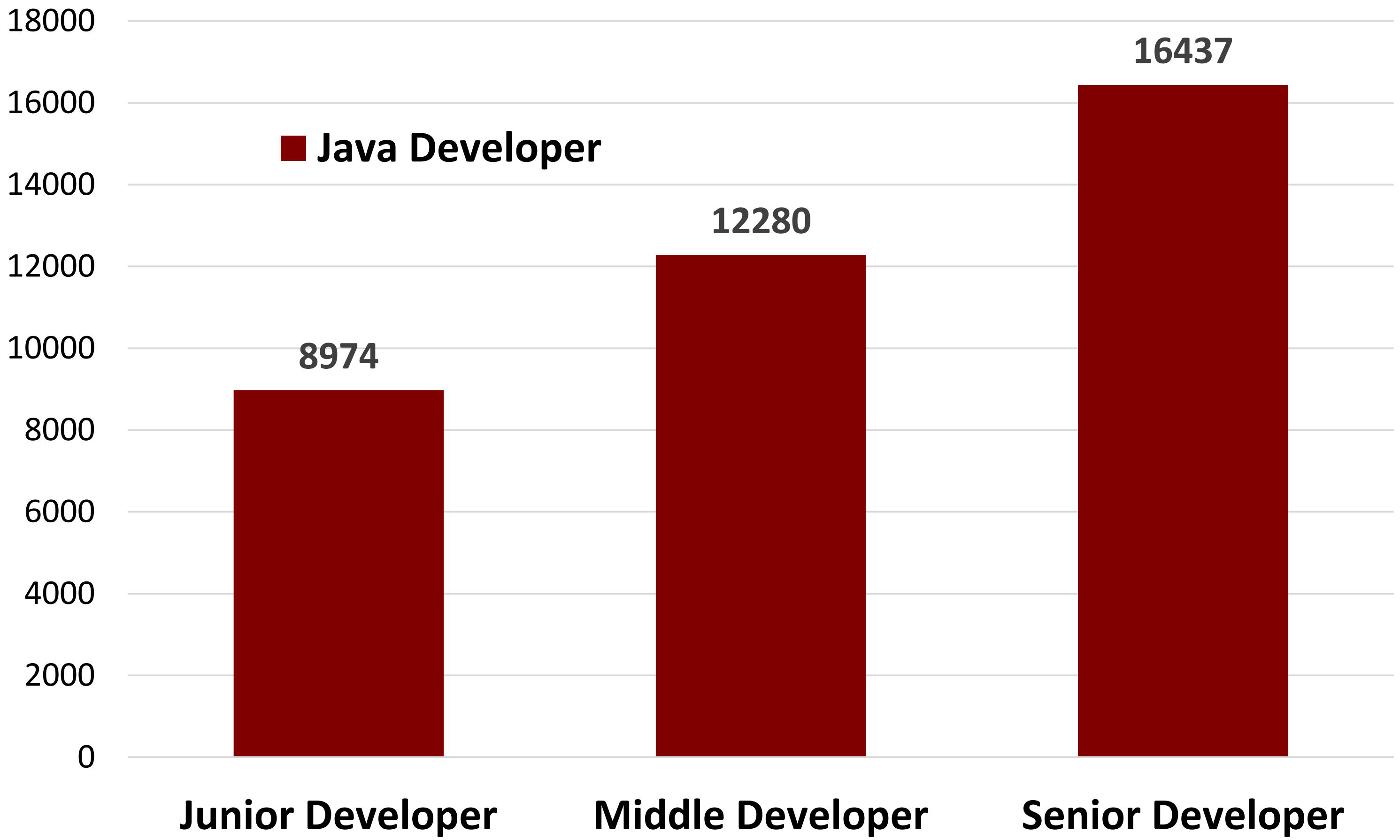
- **Indeed** – Java Developer in US – **$ 105 000**
- **Glassdoor** – Java Developer in US – **$ 99 500**
  - Range **$ 80 000 – 125 000**
  - Senior Java Developer in US – **$ 138 725**
- **Salary.com** – Java Developer in US – **$ 98 821**
  - Range **$ 83 000 – 111 000**
- **PayScale** – Java Developer – **$ 80 000**
  - Senior Java Developer – **$ 109 000**
- **Arc.dev** – Java Developer – **$ 73 000**
  - Junior Java Developer **$ 60 000**, Senior Jav Developer **$ 88 000**
- But no data for Croatia ☹

Java Development Salaries in Croatia

- According to site **Moja Plaća** about average salaries
- **Java Programmer**: 6840 - 15720 HRK
  - .NET Programmer: 7087 - 14288 HRK
  - C Programmer: 6300 - 16150 HRK
  - Frontend developer: 6032 - 13845 HRK
  - IT Product Manager: 6,719 - 20,414 HRK
  - IT Project Manager: 6,578 - 18,994 HRK
- However, it does not tell us a lot ☹
- So, we have asked those who may know more - TABU
  - TABU.hr collected more than 19000 salaries in Croatia

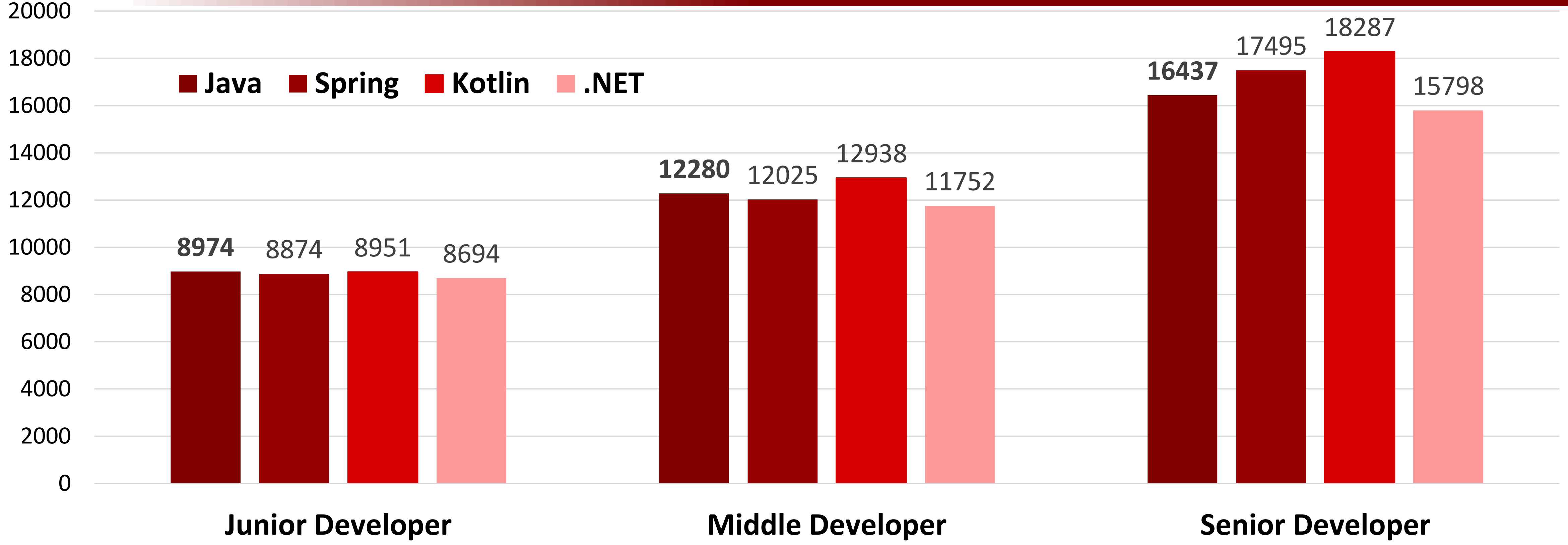
Java Development Salaries in Croatia

- **Exclusive!**
- **Average salary** for **Java Developers** in **Croatia**
  - Number of entries in TABU.hr: **791**
  - Data only for people living in Croatia, getting payed in Croatia and with contract for an indefinite period
  - Thank you TABU.hr!



TABU

Software Development Salaries in Croatia



- Comparison with Spring, Kotlin, and .NET – number of entries in TABU.hr: Java 791, Spring 117, Kotlin 140, .NET 612
- Data only for people living in Croatia, getting payed in Croatia and with contract for an indefinite period

Is Java really "Moving Forward Faster"?

- This is only a 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!

Instead of the conclusion

Use Java 17 LTS ☺

or the latest **Java 19**

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**!

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