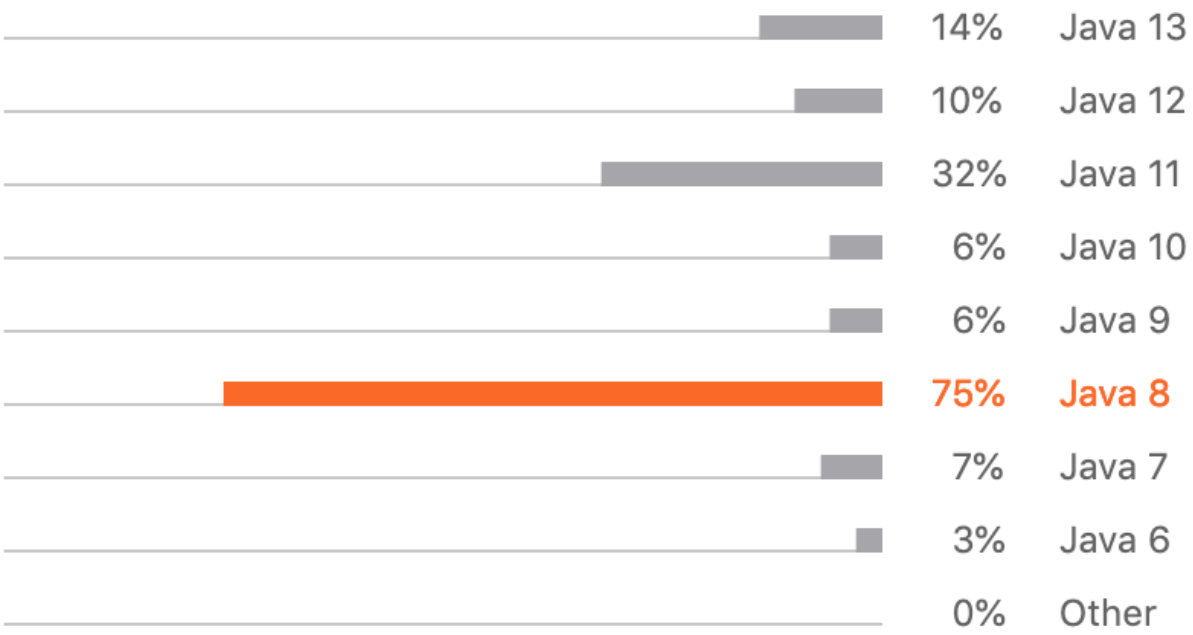
Beyond Java 8

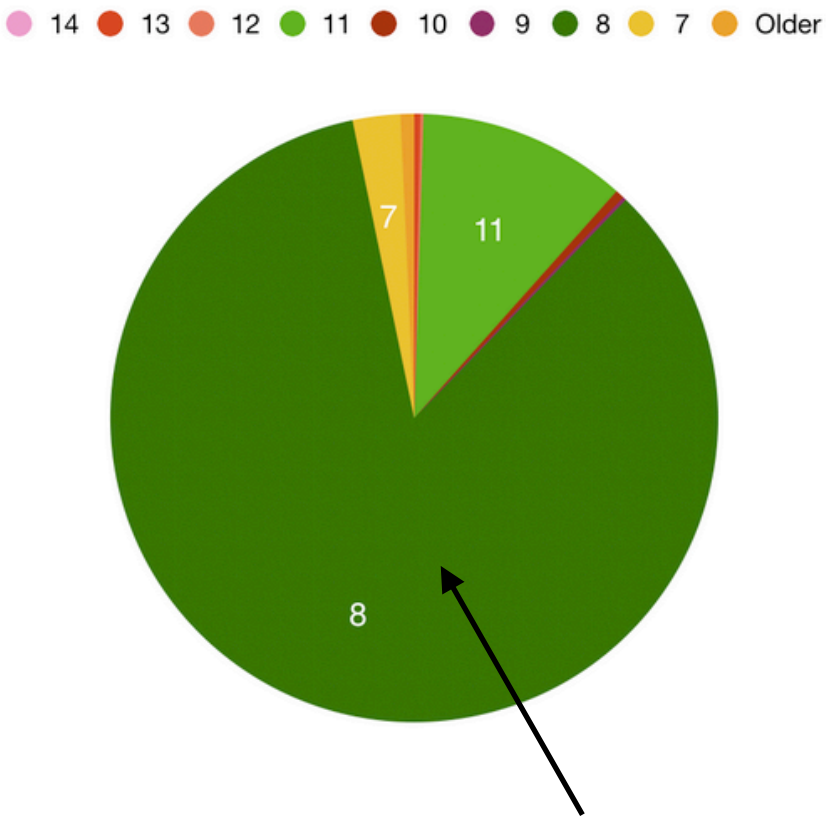
Trisha Gee (@trisha_gee)
Developer & Technical Advocate, JetBrains

Which versions of Java do you regularly use?



Despite the emergence of newer versions, Java 8 is still most beloved. It is used by 3/4 of Java developers. Java 11 is growing more popular. Compared to last year, its usage has increased by 10 percentage points.

https://www.jetbrains.com/lp/devecosystem-2020/java/



"The majority of JVMs (over 85%) are running on Java 8"

Java 8 is still the standard—for now

Let's start with the one question Java developers are always curious about: Which versions are most used in production environments? Consider the following table:

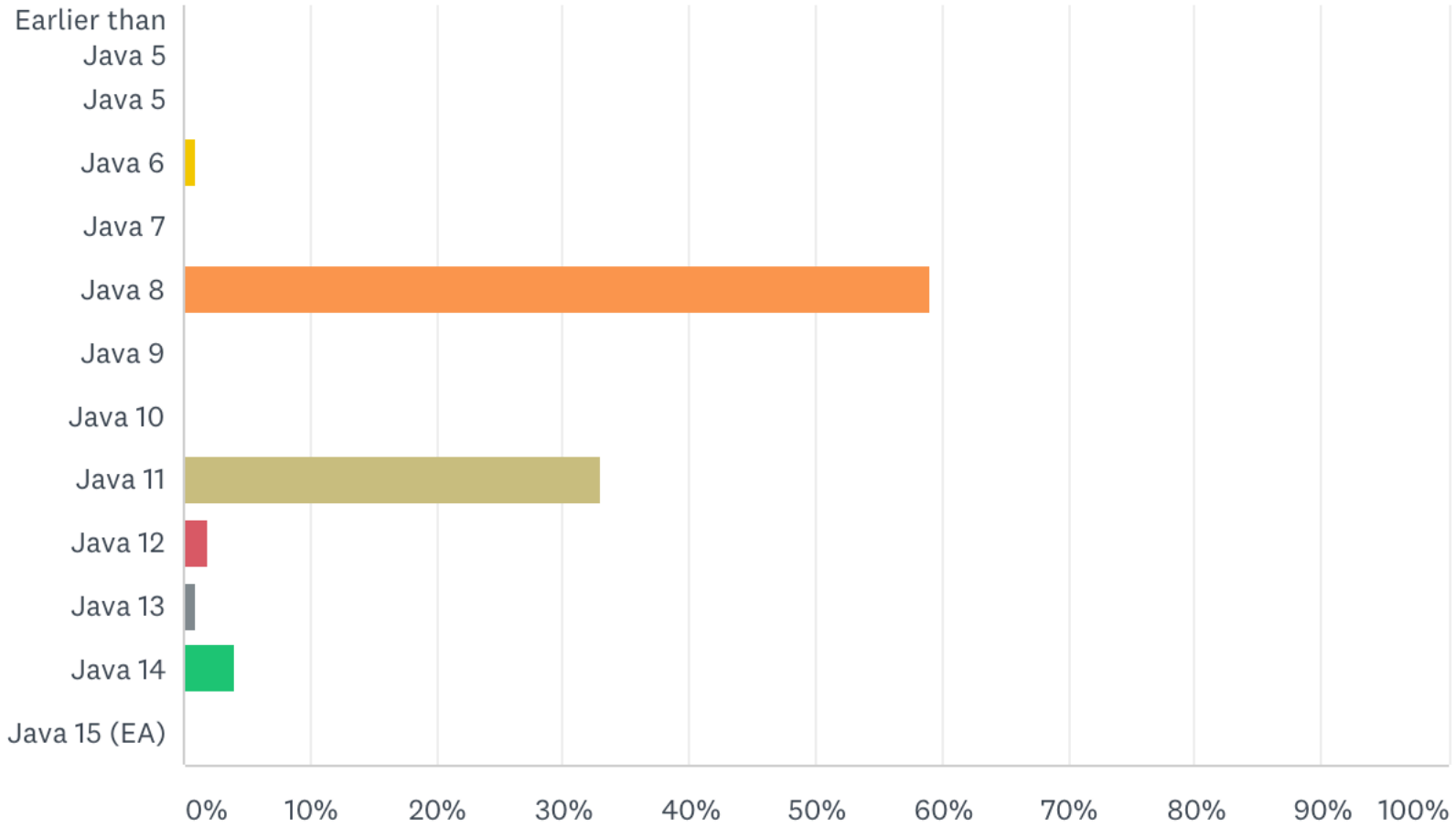
Java version	% in use
14	0.00
13	0.32
12	0.17
11	11.11
10	0.48
9	0.18
8 Current	42.02
8 Lagging	38.63
8 Vulnerable	3.83
7	2.54
pre-7	0.73
Non-LTS	1.14

https://www.infoq.com/news/2020/03/new-relic-jvm-report/

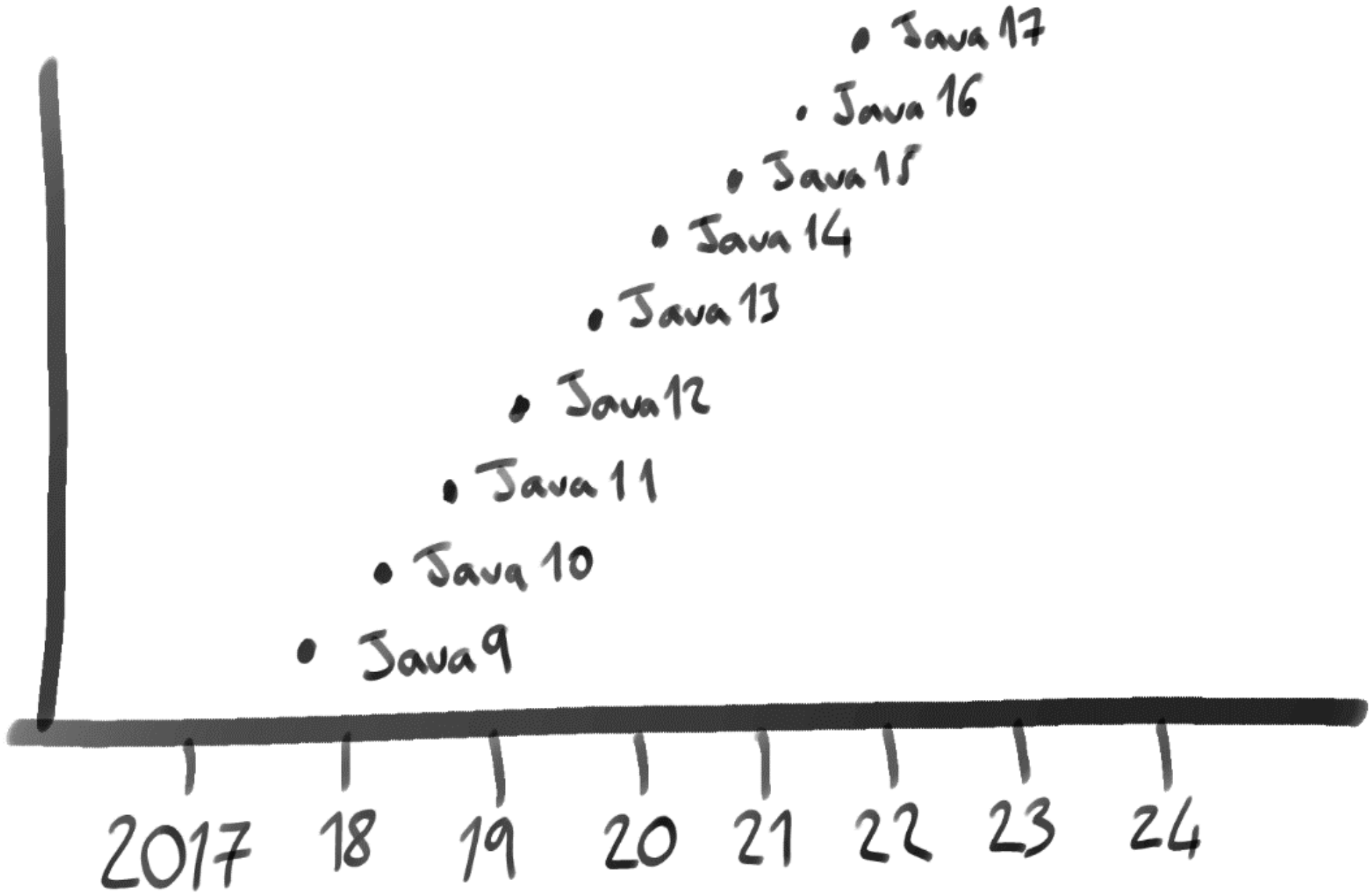
https://blog.newrelic.com/technology/state-of-java/

Which version of Java are you mainly using in production?

Answered: 100 Skipped: 0

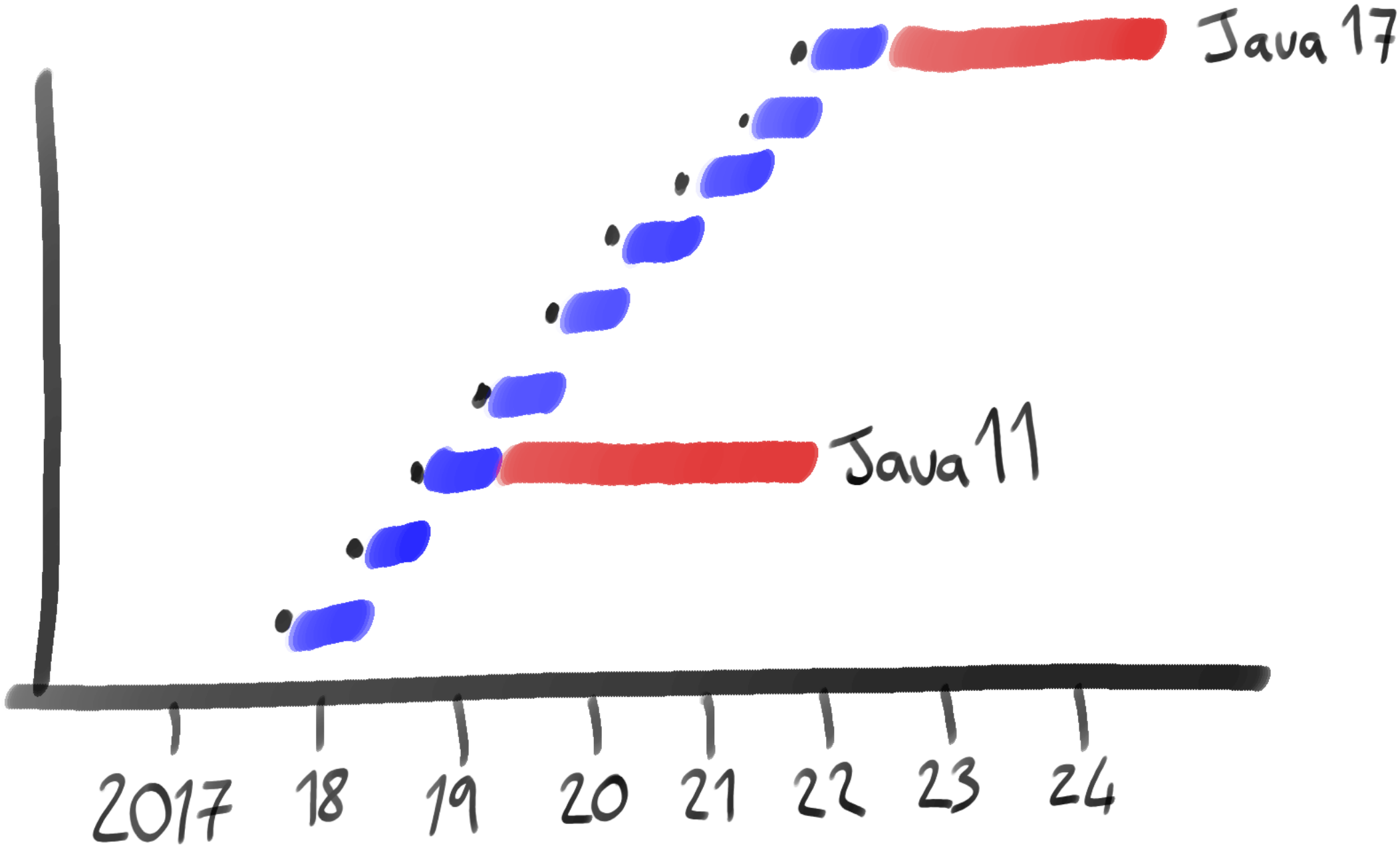


Releases, Updates, Licensing & Support

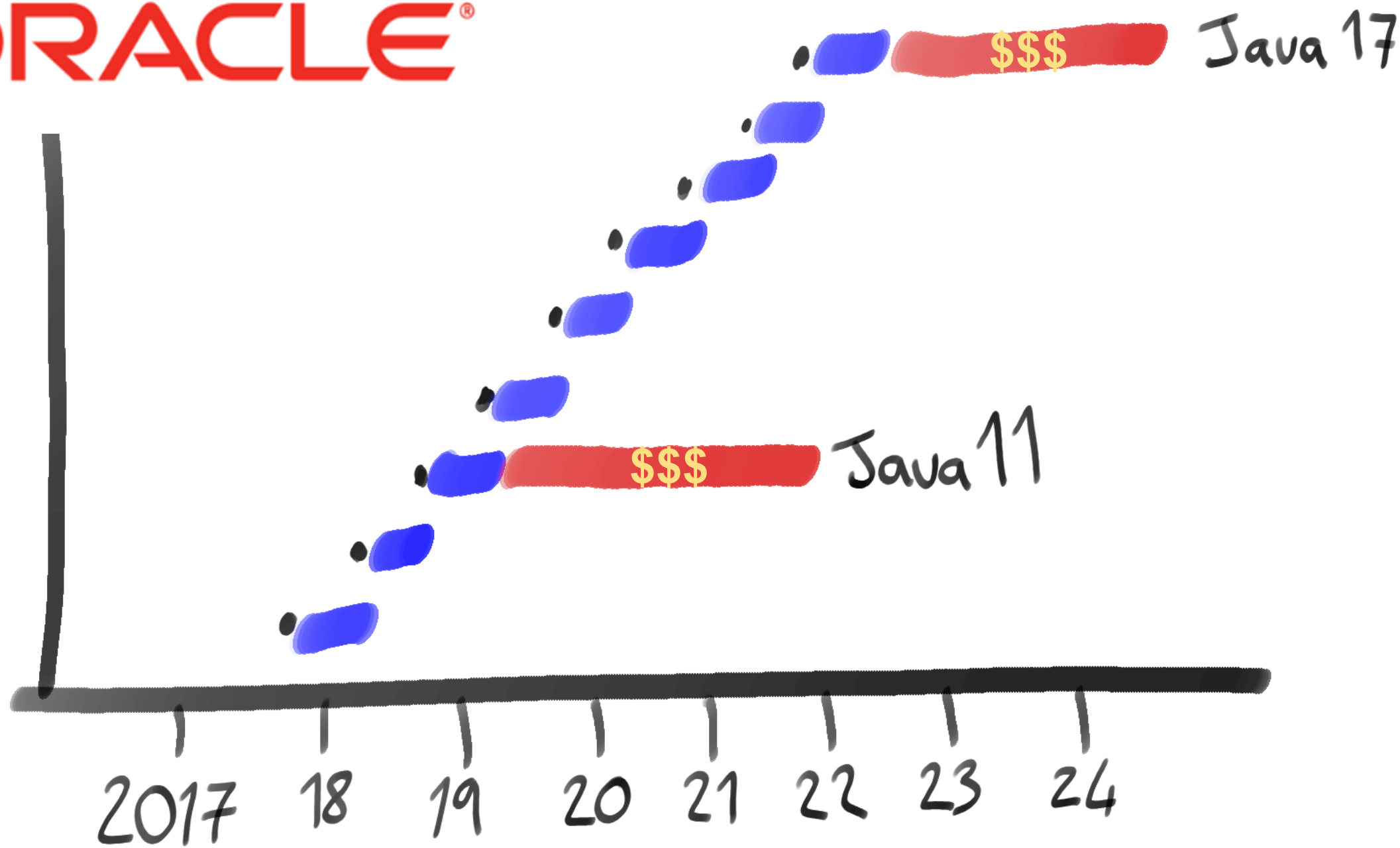

Java 17
Java 16
Java 15
Java 14
Java 13
Java 12
Java 11
Java 10
Java 9
2017
18
19
20
21
22
23
24

We have two types of releases

Releases and LTS (Long Term Support) releases


Java 17
Java 11
2017
18
19
20
21
22
23
24

ORACLE



https://flic.kr/p/8xhk74

Use $free Oracle OpenJDK

But upgrade 6 months

Java 14

Use an LTS version

But you may have to pay to use it in production

Java 11

https://jdk.dev

redhat.

ORACLE

aws

IBM

AdoptOpenJDK

AdoptOpenJDK to join the Eclipse Foundation

June 19, 2020 – posted by Technical Steering Committee 


```
var newProjectName = "Eclipse Adoptium";
```

Why bother?

Language features in Java 11

(The current Long Term Support release)

JShell

var

Convenience Factory Methods for Collections

Collecting to Unmodifable Collections

```
items.stream()
    .filter(Objects::nonNull)
    .map(Object::toString)
    .collect(Collectors.toUnmodifiableList());
```

New Methods on Stream API

```
items.stream()
    .takeWhile(user -> user.count() < maxCount)
    .forEach(user -> position.incrementAndGet());
```

Predicate.not()

New Methods on Optional

Built in Http Client

Multi Release Jar Files

- https://blog.jetbrains.com/idea/2017/10/creating-multi-release-jar-files-in-intellij-idea/

Jigsaw

Java Module System

JLink

Language features in Java 14

(The latest version of Java)

Switch Expressions

https://blog.jetbrains.com/idea/2019/02/java-12-and-intellij-idea/

Text Blocks (Preview)

https://blog.jetbrains.com/idea/2019/11/java-13-and-intellij-idea

Records (Preview)

https://blog.jetbrains.com/idea/2020/03/java-14-and-intellij-idea

Pattern Matching for instanceof (Preview)

https://blog.jetbrains.com/idea/2020/03/java-14-and-intellij-idea/

Beyond Java 14

Java 15

- JEP 378: Text Blocks (Standard)
- JEP 375: Pattern Matching for instanceof (Second Preview)
- JEP 384: Records (Second Preview)
- JEP 360: Sealed Types (Preview)
- JEP 371: Hidden Classes
- JEP 377: ZGC: A Scalable Low-Latency Garbage Collector (Production)
- https://openjdk.java.net/projects/jdk/15/

And in the future?

- Project Amber
- Valhalla
- Loom
- More Garbage Collection improvements
- Proposed New Project: Leyden

The Business Doesn’t Care About Language Features

Performance

Use of Memory

Garbage Collection (Java 11)

- Java 9: JEP 248: G1 the Default GC
- Java 10: JEP 307: Parallel Full GC for G1
- Java 11: JEP 318: Epsilon (Experimental)
- Java 11: JEP 333: ZGC (Experimental)

Garbage Collection (Java 14)

- Java 12: Shenandoah (Experimental)
- Java 12: More Updates to G1
- Java 12: More Improvements to ZGC
- Java 13: ZGC: Uncommit Unused Memory
- Java 14: ZGC on macOS and Windows
- Java 14: Deprecate the ParallelScavenge + SerialOld GC Combination

Cost

£$€

Tips for Migration

Run on updated JDK

It might “just work”

Address compiler warnings

…they are there for a reason

Update your dependencies

And add new ones

Update your build tool

Updated Maven and Gradle required

Compile against updated JDK

…and start using the shiny new features

https://www.infoq.com/articles/upgrading-java-8-to-12/

In Summary

Java Is Changing

…fast

Modern Java Can Help You

Performance, cost, maintenance…

There are two upgrade options

To Java 11 (LTS) or to Java 14 (upgrade every 6 months)

Upgrade Now And Reduce Future Pain

…and keep upgrading, at least in CI

http://bit.ly/beyond-j8

@trisha_gee