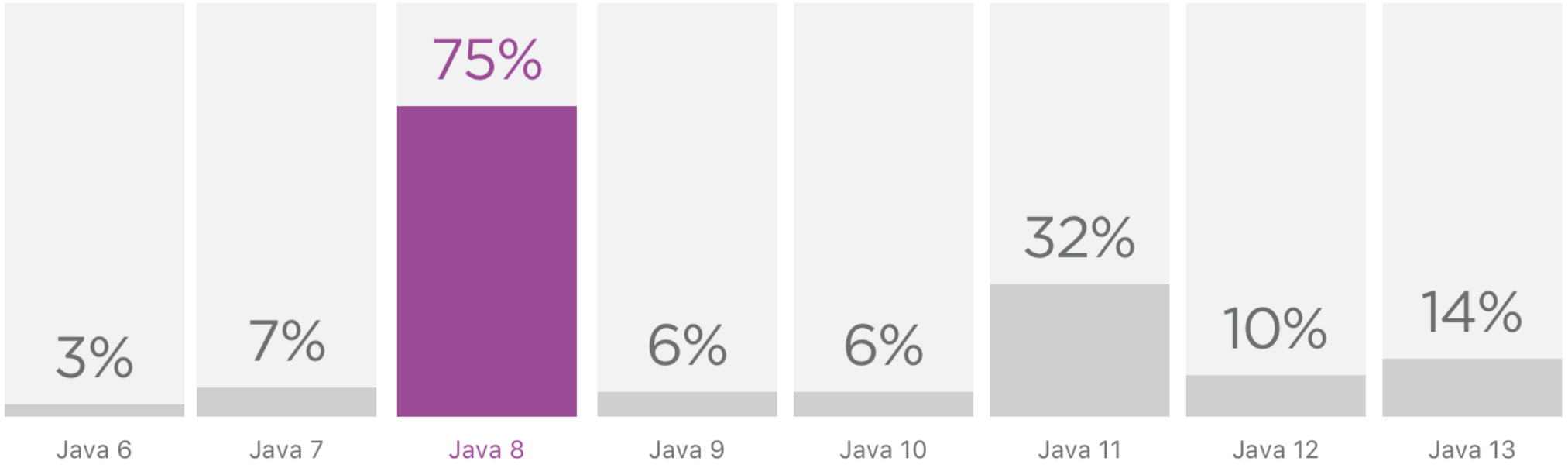
IntelliJ IDEA

Life Beyond Java 8

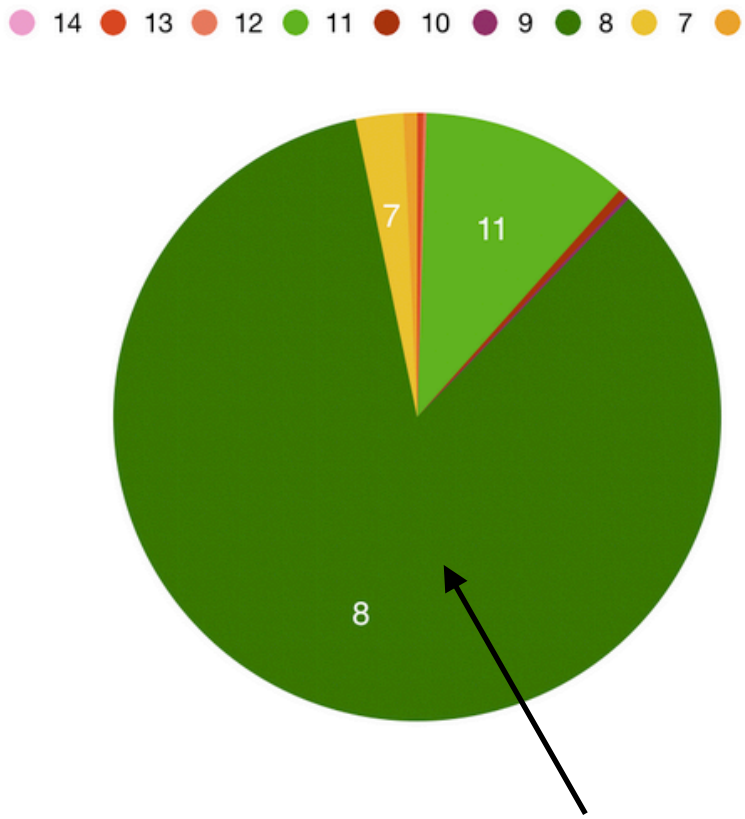
Trisha Gee (@trisha_gee)
Java Developer Advocacy Lead, JetBrains

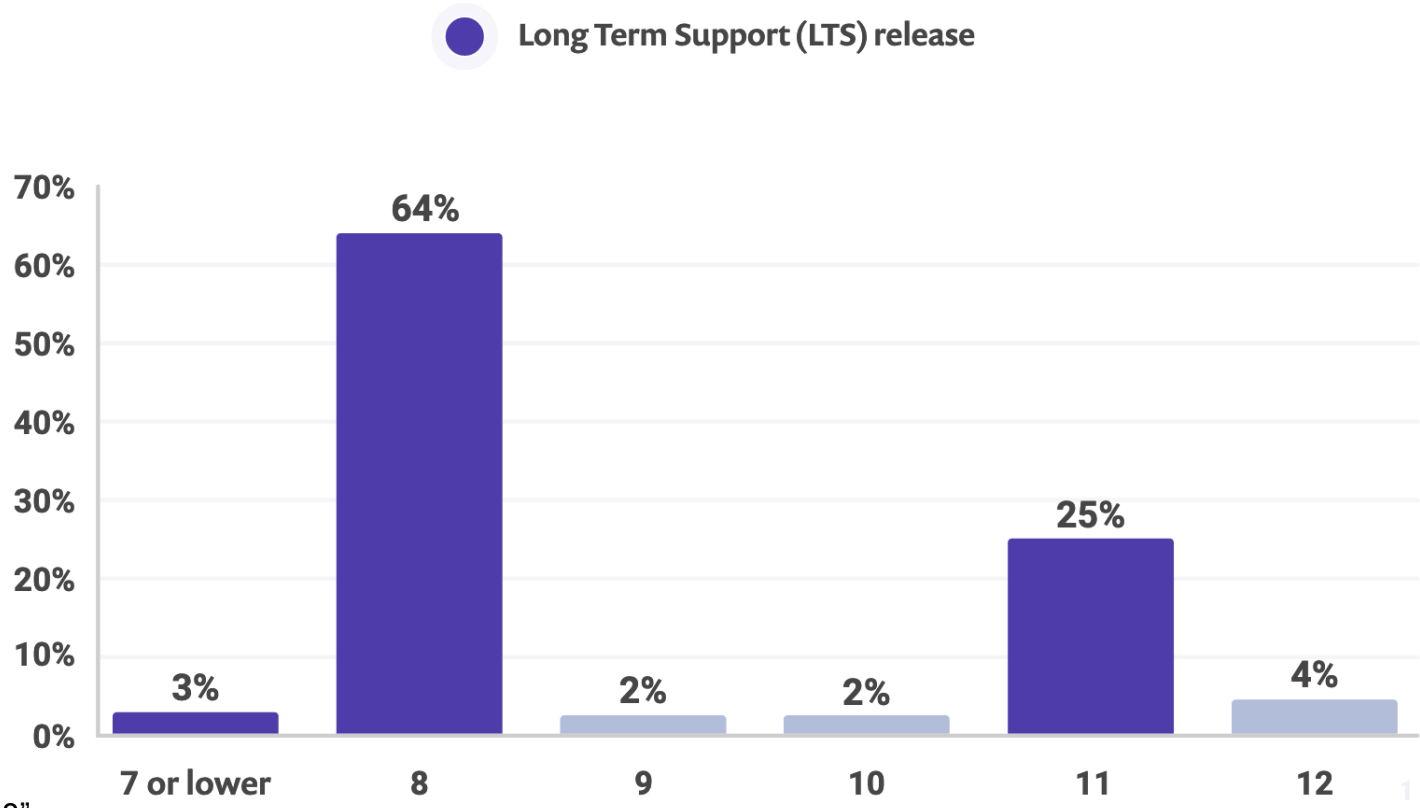
Popularity of Java versions in 2020



Java 8 remains the most popular version. It is used by 75% of professional developers who use Java as their primary language.

https://blog.jetbrains.com/idea/2020/09/a-picture-of-java-in-2020/



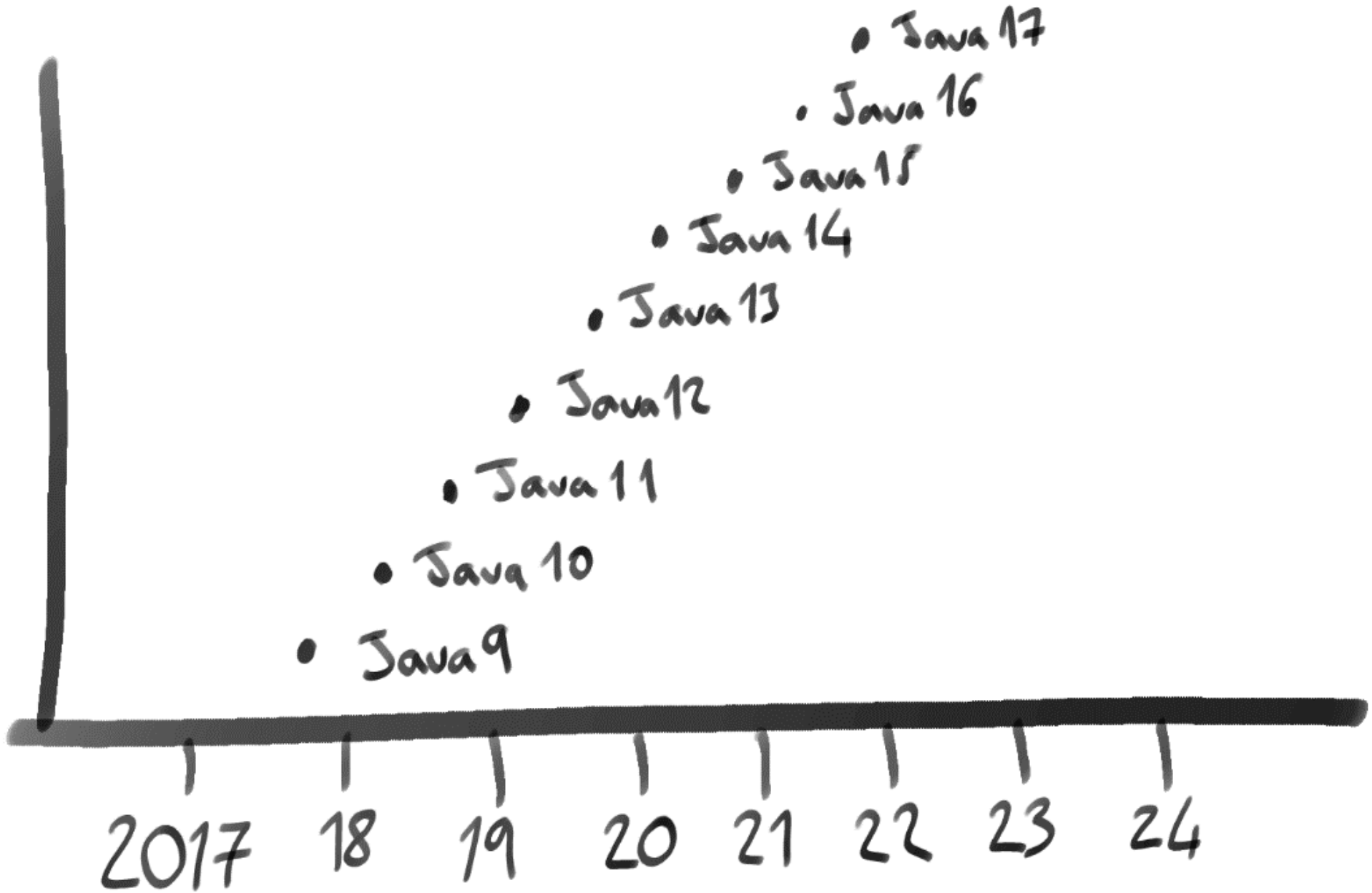


"The majority of JVMs (over 85%) are running on Java 8"

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

https://snyk.io/wp-content/uploads/jvm_2020.pdf

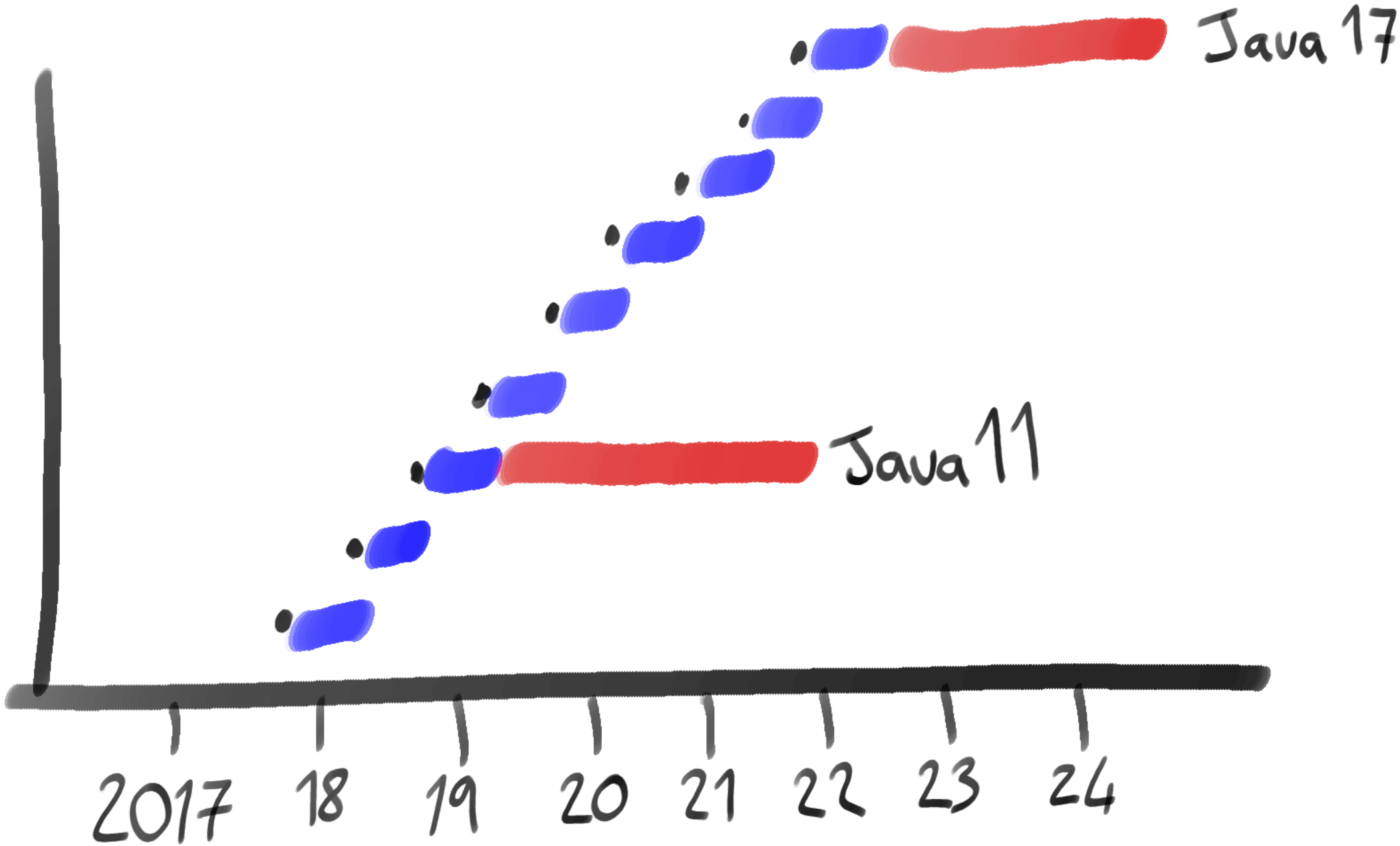
Releases, Updates, Licensing & Support

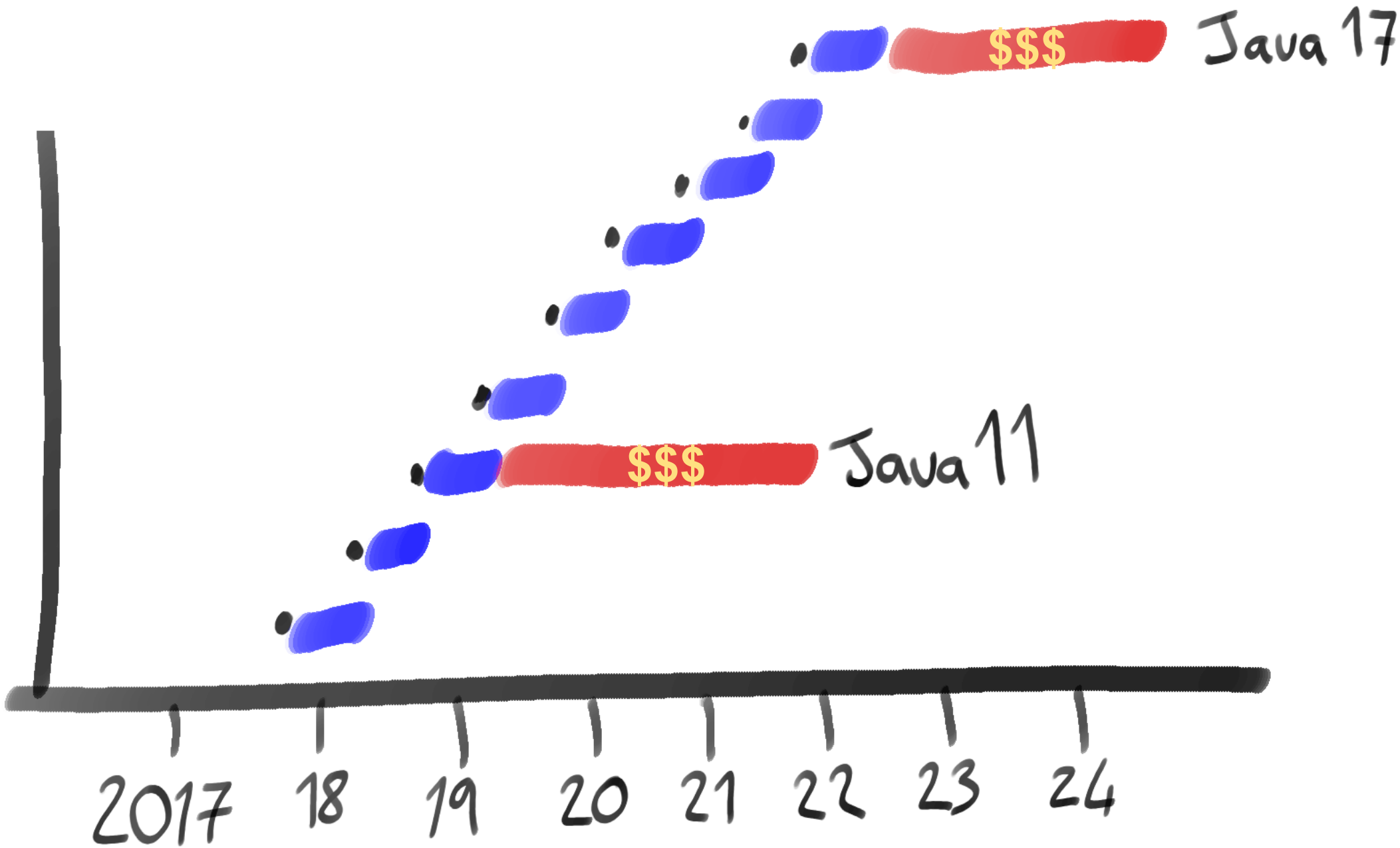

Java 17
Java 16
Java 15
Java 14
Java 13
Java 12
Java 11
Java 10
Java 9
2017
18
19
20
21
22
23
24

We have two types of releases

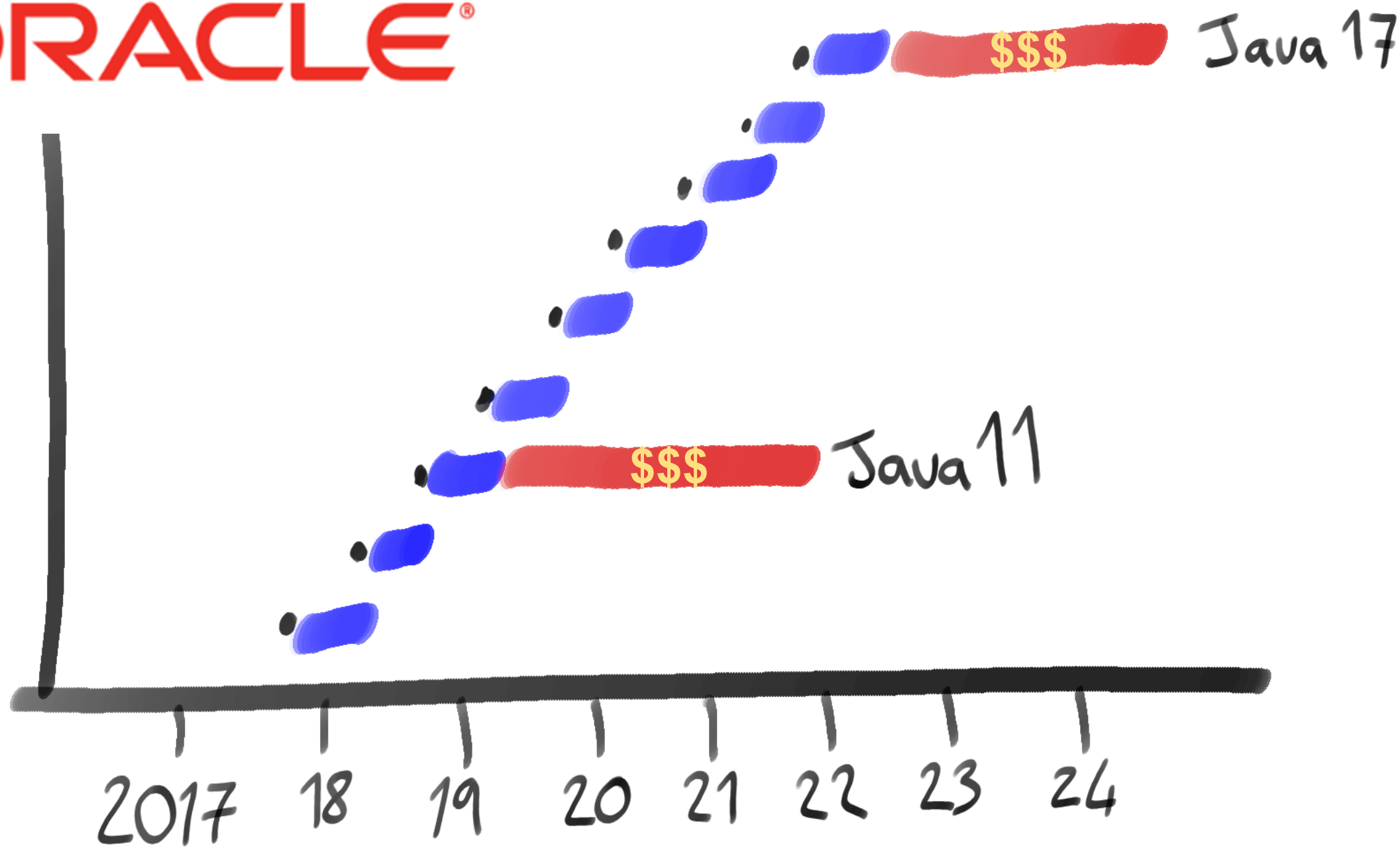
We have two types of releases

Releases and LTS (Long Term Support) releases


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


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

ORACLE



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

Java 11

Long Term Support (LTS) Release; ~3 years support

Java 15

Current release; be prepared to upgrade in March 2021

https://jdk.dev

redhat.

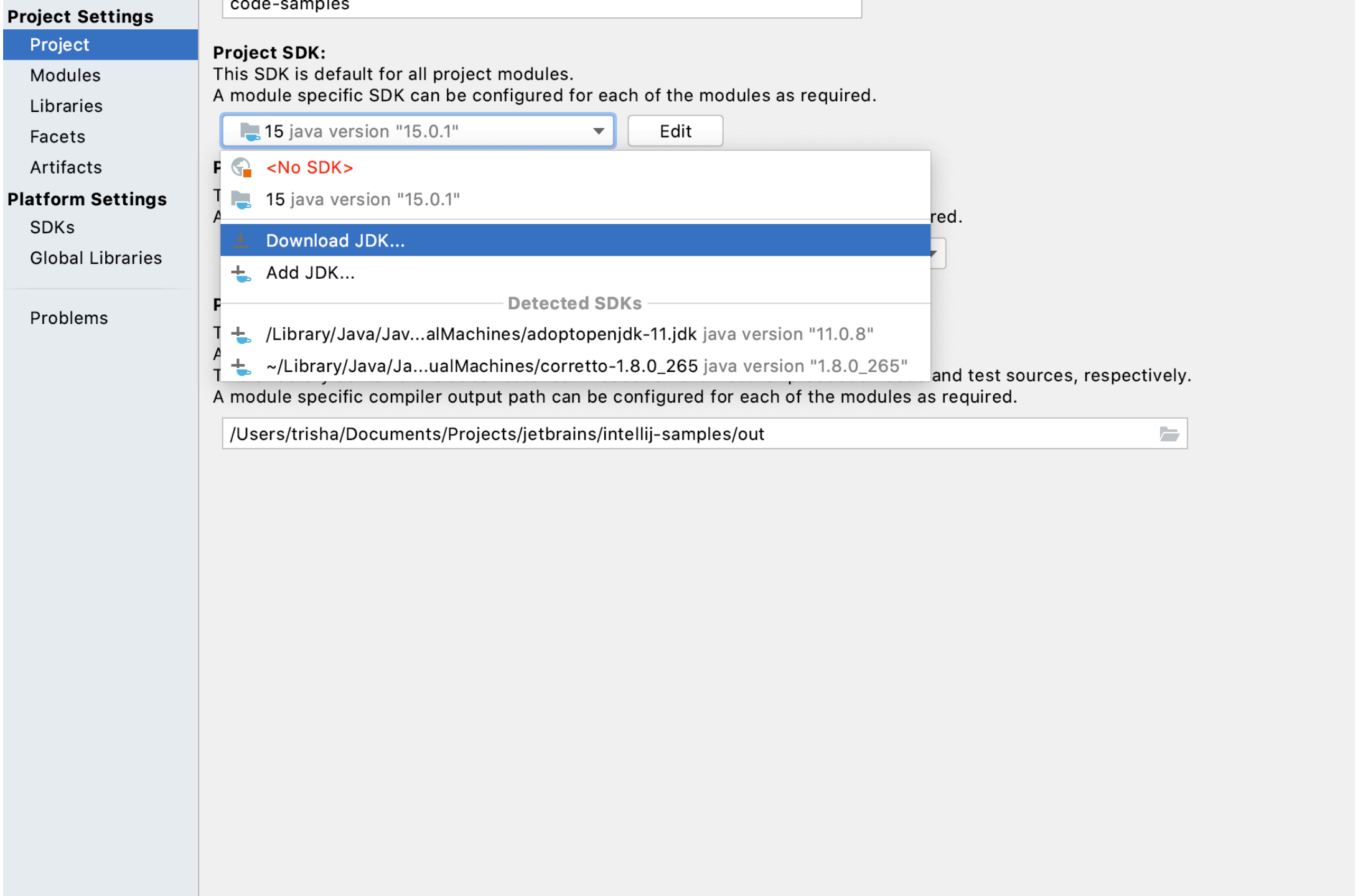
ORACLE

aws

IBM

AdoptOpenJDK

IntelliJ IDEA can help you


Project Settings
Project
Modules
Libraries
Facets
Artifacts
Platform Settings
SDKs
Global Libraries
Problems
code-samples
Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
15 java version "15.0.1"
Edit
<No SDK>
15 java version "15.0.1"
Download JDK...
Add JDK...
Detected SDKs
/Library/Java/Jav...alMachines/adoptopenjdk-11.jdk java version "11.0.8"
~/Library/Java/Ja...ualMachines/corretto-1.8.0_265 java version "1.8.0_265"
and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.
/Users/trisha/Documents/Projects/jetbrains/intellij-samples/out

Why bother?

Java 11

Long Term Support (LTS) 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

Java 15

Current Release

Switch Expressions

Switch Expressions

https://jb.gg/ij-jdk12

Text Blocks

Text Blocks

https://jb.gg/ij-jdk13

Hidden classes

Hidden classes

JEP 371

A Scalable Low-Latency Garbage Collector

A Scalable Low-Latency Garbage Collector

JEP 377

Preview Features

Pattern Matching for instanceof

Pattern Matching for instanceof

https://jb.gg/ij-jdk14

Records

Records

https://jb.gg/ij-jdk14

Sealed Types

Sealed Types

https://jb.gg/ij-jdk15

https://openjdk.java.net/projects/jdk/15/

The Future

Java 16

- JEP 302: Lambda Leftovers (including underscore for param)
- JEP 360: Sealed Types (Preview)
- JEP draft: Pattern matching for switch (Preview)
- Project Amber
- Valhalla
- Loom
- Lots of Garbage Collection improvements

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 15)

- Java 12: More Updates to G1
- Java 14: Deprecate the ParallelScavenge + SerialOld GC Combination
- Java 15: ZGC (Production)
- Java 15: Shenandoah (Production)

Cost

Cost

£$€

Tips for Migration

Run on updated JDK

Run on updated JDK

It might “just work”

Address compiler warnings

Address compiler warnings

…they are there for a reason

Update your dependencies

Update your dependencies

And add new ones

Update your build tool

Update your build tool

Updated Maven and Gradle required

Compile against updated JDK

Compile against updated JDK

…and start using the shiny new features

https://bit.ly/8-to-11

In Summary

Modern Java Can Help You

Modern Java Can Help You

Performance, cost, maintenance…

There are two upgrade options

There are two upgrade options

To Java 11 (LTS) or to Java 15 (upgrade every 6 months)

Upgrade Now And Reduce Future Pain

Upgrade Now And Reduce Future Pain

…and keep upgrading, at least in CI

https://bit.ly/love-beyond-8

@trisha_gee