JET BRAINS

Beyond Java 8

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

Upgrading Past Java 9 Sounds Scary

…and I don’t want to pay for Java


Super happy with Java 8, thanks

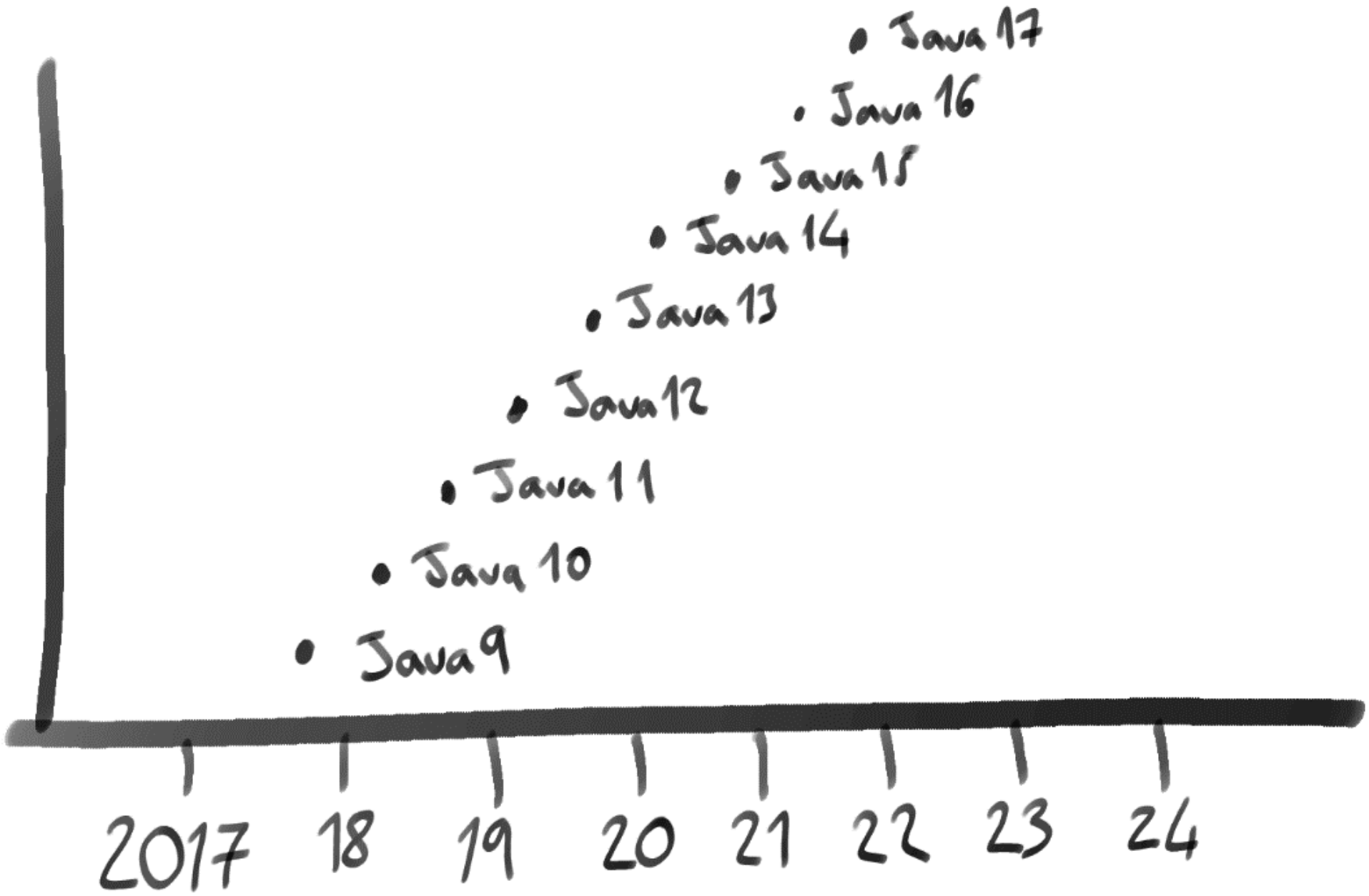
Trisha Gee
@trisha_gee

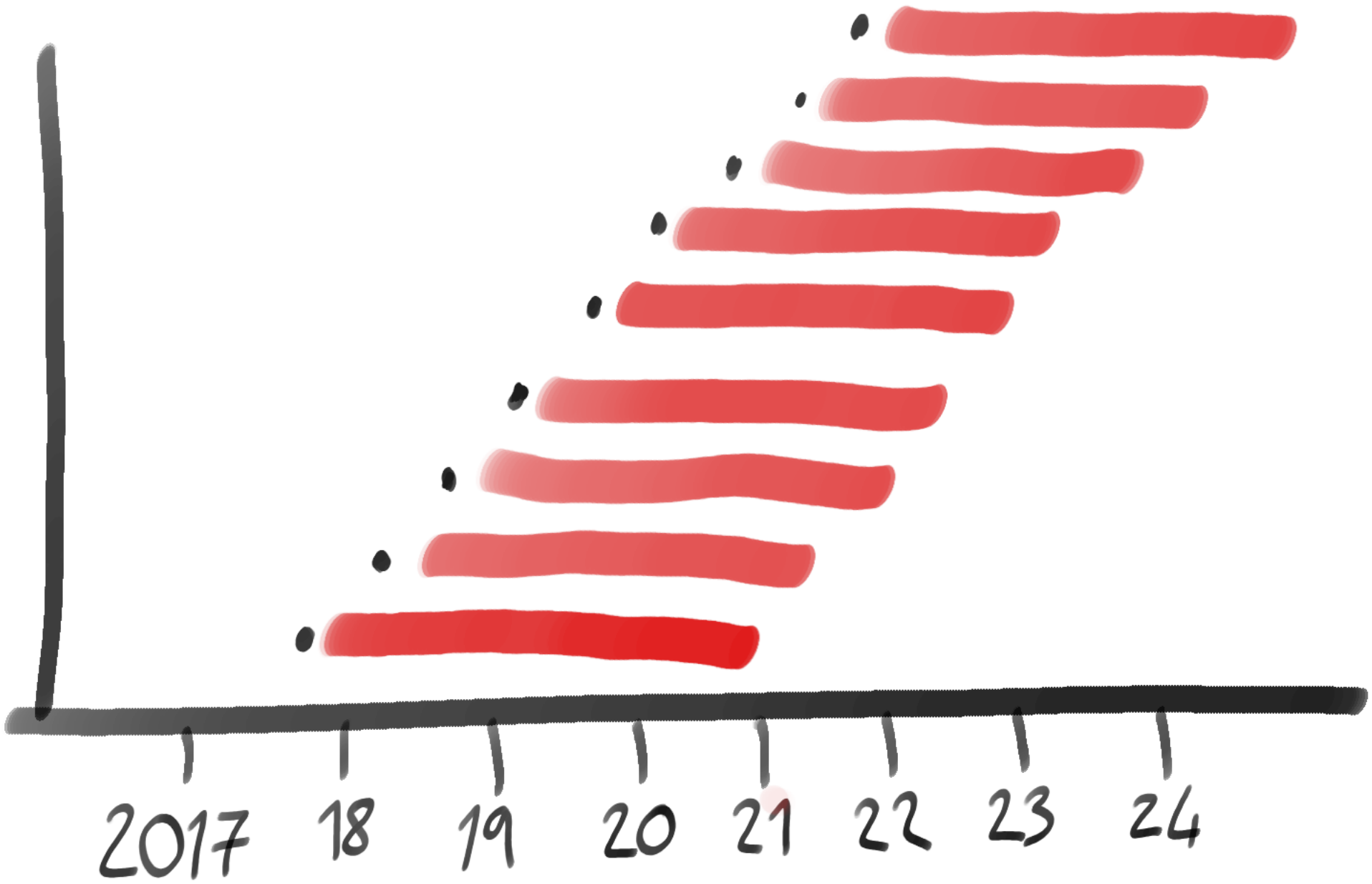
Which version of #Java are you using?

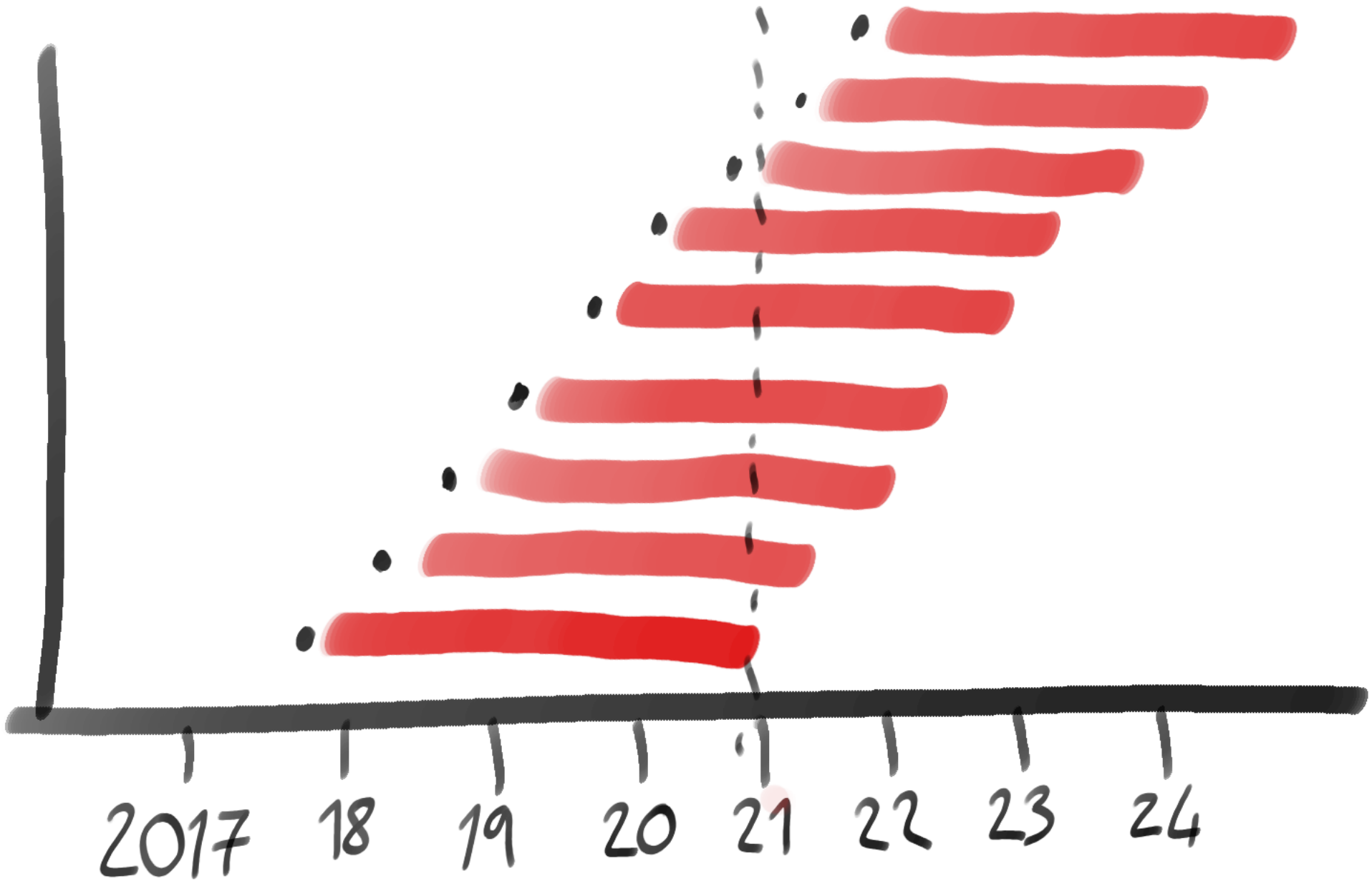
78%	Java 8
2%	Java 9
2%	Java 10
18%	Java 11

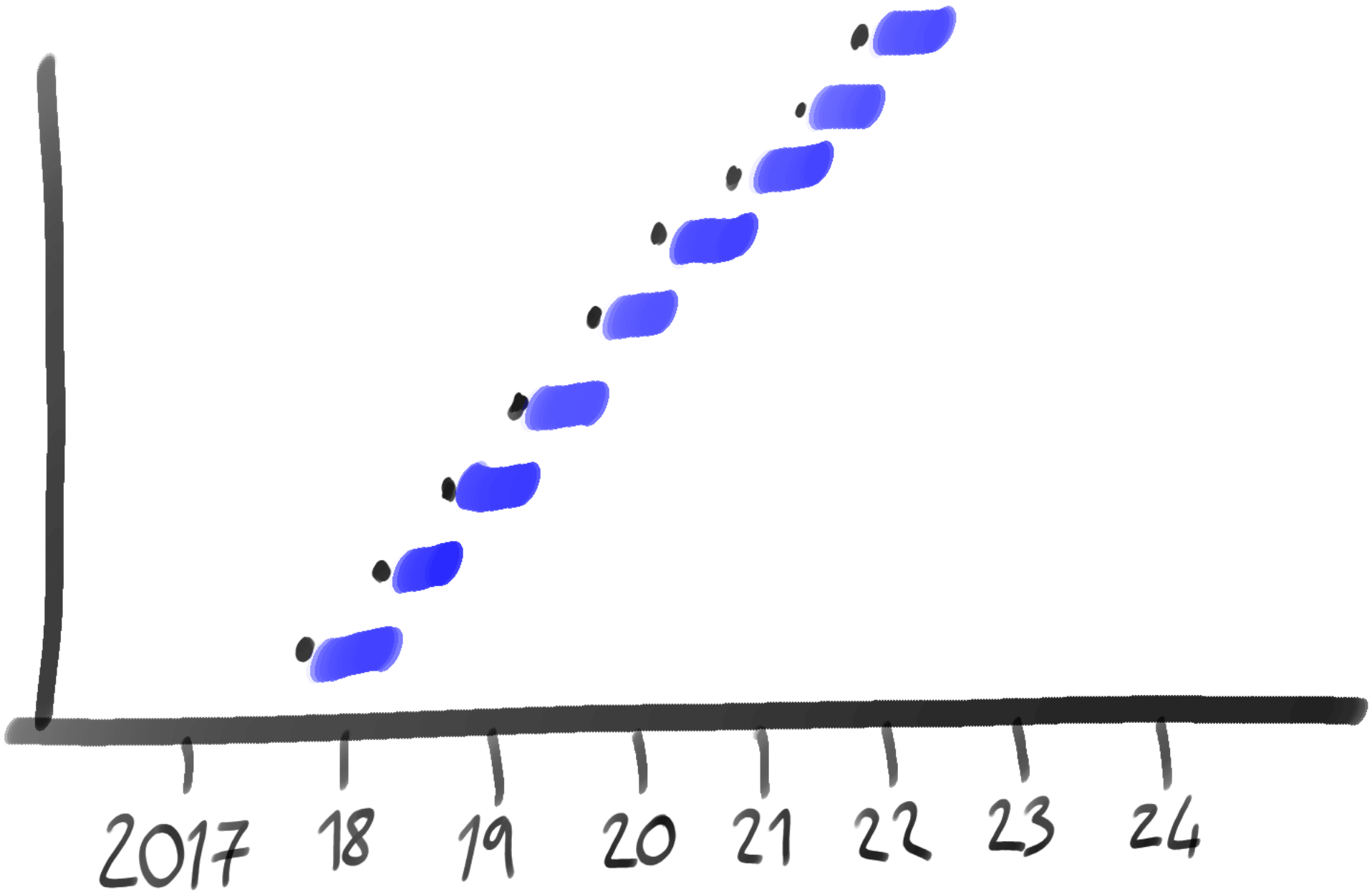
1,371 votes • Final results

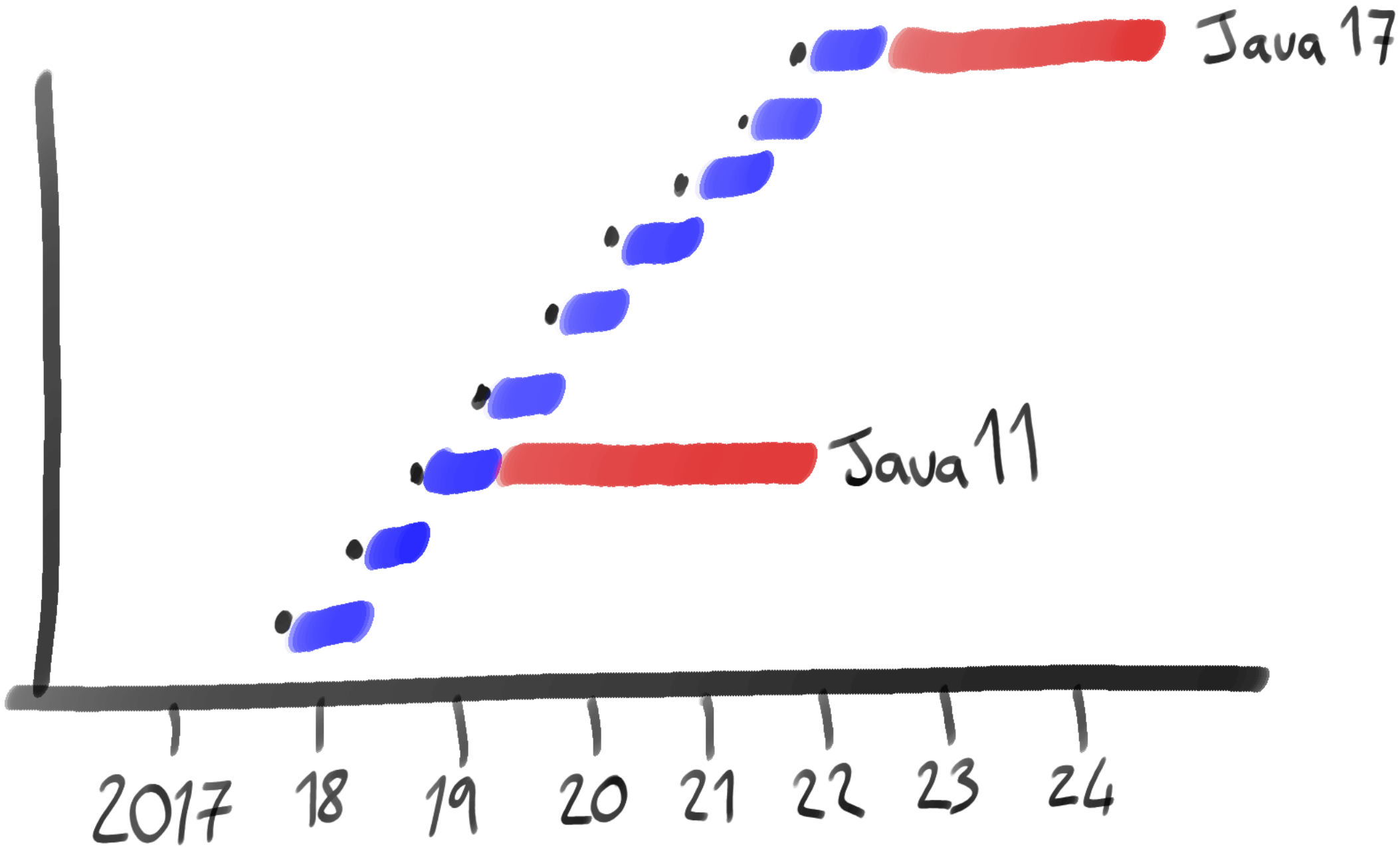
Releases, Updates, Licensing & Support

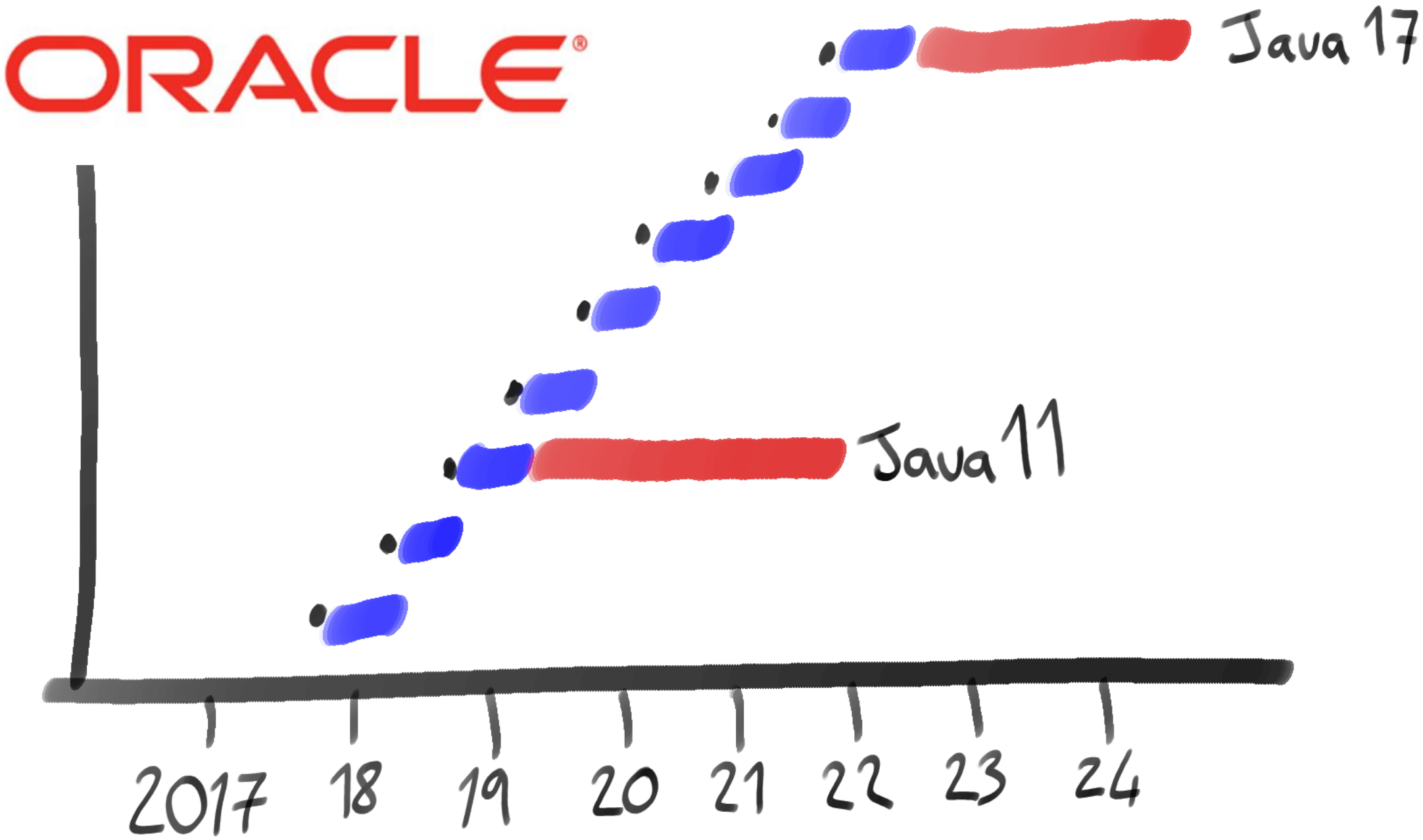

Java 17
Java 16
Java 15
Java 14
Java 13
Java 12
Java 11
Java 10
Java 9
2017
18
19
20
21
22
23
24


2017
18
19
20
21
22
23
24


2017
18
19
20
21
22
23
24


2017
18
19
20
21
22
23
24


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


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

Starting with Java 11, Oracle will provide JDK releases under the open source GNU General Public License v2, with the Classpath Exception (GPLv2+CPE), and under a commercial license for those using the Oracle JDK as part of an Oracle product or service, or who do not wish to use open source software. This… replaces the historical "BCL" license, which had a combination of free and paid commercial terms.

https://blogs.oracle.com/java-platform-group/oracle-jdk-releases-for-java-11-and-later

Donald Smith, Sr. Director of Product Management (June 2018)

That sounds frightening, and overly complicated. Talk us through it.

The Lego Batman Movie

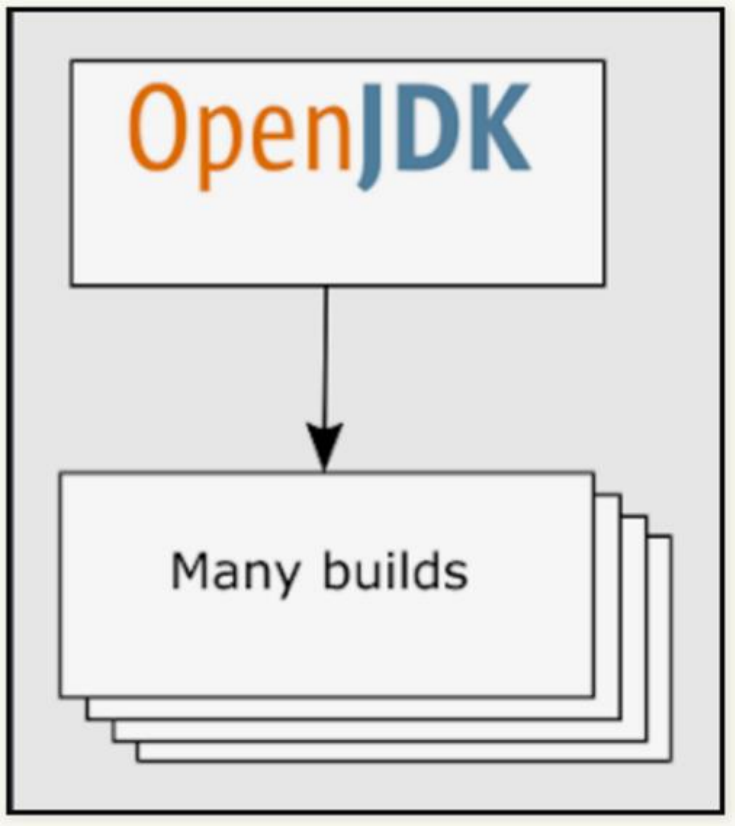

That sounds frightening, and overly complicated. Talk us through it.
The Lego Batman Movie

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

From Java 11 forward, Oracle JDK builds and OpenJDK builds will be essentially identical.

https://blogs.oracle.com/java-platform-group/oracle-jdk-releases-for-java-11-and-later

Donald Smith, Sr. Director of Product Management (Sept 2018)



https://blog.joda.org/2018/09/time-to-look-beyond-oracles-jdk.html

redhat.

ORACLE®

aws

IBM®

AdoptOpenJDK

AdoptOpenJDK

1. Choose a Version

- (•) OpenJDK 8 (LTS)
- ( ) OpenJDK 9
- ( ) OpenJDK 10
- ( ) OpenJDK 11 (LTS)

2. Choose a JVM

- (•) HotSpot
- ( ) OpenJ9

Select a platform

Docker Official Image	Linux x64 jdk8u202-b08	Windows x32 jdk8u202-b08
Windows x64 jdk8u202-b08	macOS x64 jdk8u202-b08	Linux s390x jdk8u202-b08

https://adoptopenjdk.net

In addition, every three years one feature release will be designated as the Long Term Supported (LTS) release. We will produce LTS releases for at least four years. This assurance will allow you to stay on a well-defined code stream, and give you time to migrate to the next, new, stable, LTS release when it becomes available.

Based upon this roadmap, and starting with Java 8:

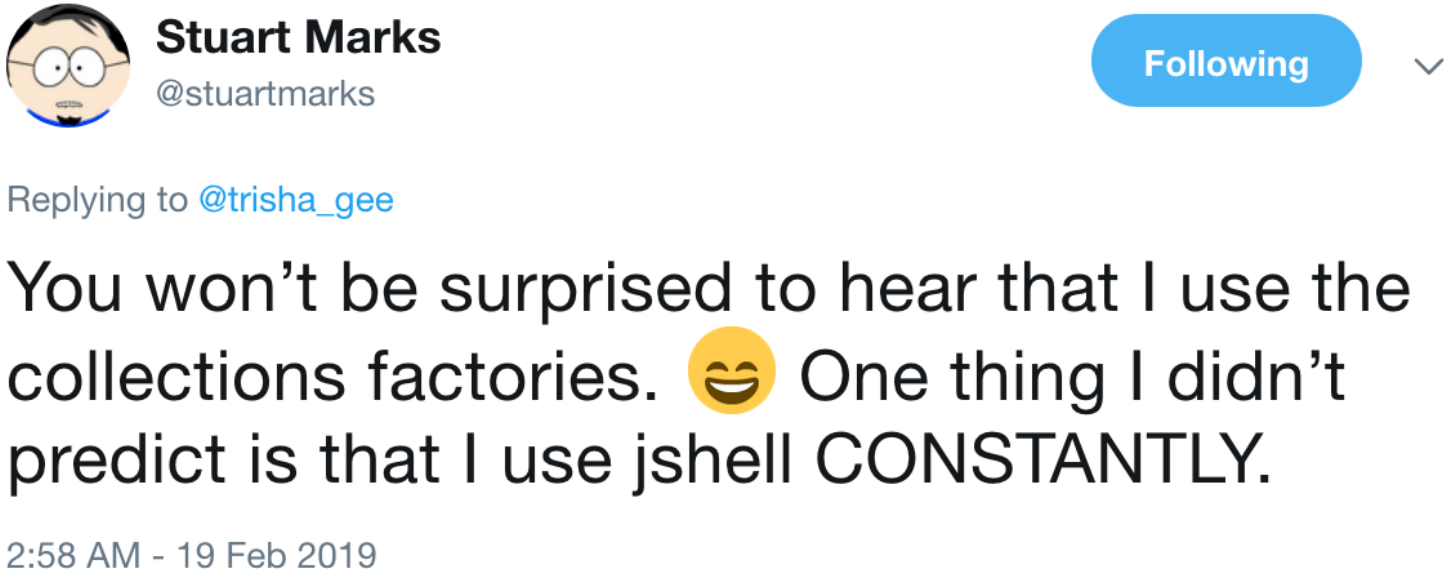
	First Availability	End of Availability [1]
Java 8 (LTS)	March 2014	At Least Sep 2023
Java 9	Sept 2017	March 2018
Java 10	March 2018	Sept 2018
Java 11 (LTS)	Sept 2018	At Least Sept 2022

https://adoptopenjdk.net/support.html

Why bother?

Language features

JShell


Stuart Marks
@stuartmarks
Following
Replying to @trisha_gee
You won't be surprised to hear that I use the collections factories. 😄 One thing I didn't predict is that I use jshell CONSTANTLY.
2:58 AM - 19 Feb 2019

JShell

Shaggy Dude @akubot · Feb 18

Replying to @craftingjava @trisha_gee

v9+ has jshell built in, so no need to install another utility or run Scala REPL instead

JShell

Christian Stein @sormuras · Feb 18

Replying to @trisha_gee

modules - !

mrjar - #junit5's platform ships a one, since day 1.

jshell and java Bach.java (JEP330) - use #java to build #java.

Also var, List/Set.of(), String.isBlank(), String.lines(), String.repeat()

var

Christian Stein @sormuras · Feb 18

Replying to @trisha_gee

modules - !
mrjar - #junit5's platform ships a one, since day 1.
jshell and java Bach.java (JEP330) - use #java to build #java.
Also var, List/Set.of(), String.isBlank(), String.lines(), String.repeat()

var

Tudor Prodan @tudor_prodan · Feb 18

Replying to @trisha_gee

var

var

Davy Steegen @dsteegen · Feb 18

Replying to @trisha_gee

I start to enjoy using the var keyword and the better memory management in Docker containers.

var

Laszlo Csontos @craftingjava · Feb 18

Replying to @trisha_gee

Compact strings, List.of(), Var, Optional.ifPresentOrElse(), standardized JVM options and log, free flight recorder (11+)

var

Marcel Overdijk @marceloverdijk · Feb 18

Replying to @lukaseder @trisha_gee

var is great, but val in @kotlin is even nicer 👌

var

Adeyemo Adesegun @alexcrownus · Feb 18

Replying to @trisha_gee

Java 11 + Lombok kind of gets me closer to Kotlin-like productivity for existing codebase. I might want to use Kotlin for new projects though.

Convenience Factory Methods for Collections

Laszlo Csontos @craftingjava · Feb 18

Replying to @trisha_gee

Compact strings, List.of(), Var, Optional.ifPresentOrElse(), standardized JVM options and log, free flight recorder (11+)

Convenience Factory Methods for Collections

redestad @cl4es · Feb 18

Replying to @trisha_gee

I work on @OpenJDK so I'm surely biased, but my daily #java usage has become so much more enjoyable with jshell, var, Map/Set/List.of and overall better performance, snappier startup etc..

Convenience Factory Methods for Collections

Christian Stein @sormuras · Feb 18

Replying to @trisha_gee

modules - !

mrjar - #junit5's platform ships a one, since day 1.

jshell and java Bach.java (JEP330) - use #java to build #java.

Also var, List/Set.of(), String.isBlank(), String.lines(), String.repeat()

Convenience Factory Methods for Collections

Arash Shahkar @ArashShahkar · Feb 19

Replying to @trisha_gee

11, and lots of reasons. The ones I find useful every single day: CGroup-friendliness, collection factories, Optional enhancements.

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()`

Daniel Kraus @beatngu1101 · Feb 18

Replying to @trisha_gee

Since it hasn't been mentioned so far: There's finally `Predicate::not` in Java 11.
🎉

New Methods on `Optional`

Laszlo Csontos @craftingjava · Feb 18

Replying to @trisha_gee

Compact strings, List.of(), Var, Optional.ifPresentOrElse(), standardized JVM options and log, free flight recorder (11+)

Arash Shahkar @ArashShahkar · Feb 19

Replying to @trisha_gee

11, and lots of reasons. The ones I find useful every single day: CGroup-friendliness, collection factories, Optional enhancements.

Http Client

Colin Vipurs I am eagerly awaiting moving to 11 (it's imminent), purely for the HTTP client. We've moved fully to kotlin now, so the language features don't interest me, but having a really good built in HTTP client will solve other headaches.

Like · Reply · 2d

Trisha Gee That is really useful to know

Like · Reply · 2d

Colin Vipurs Proper non-blocking, reactive-streams support, HTTP1.1 and 2. It's got everything we need. Every other client ends up with some kind of dependency hell so excited to have a built-in one

@tddmonkey

Multi Release Jar Files

Christian Stein @sormuras · Feb 18

Replying to @trisha_gee

modules - !

mrjar - #junit5's platform ships a one, since day 1.

jshell and java Bach.java (JEP330) - use #java to build #java.

Also var, List/Set.of(), String.isBlank(), String.lines(), String.repeat()

Jigsaw

Java Module System

Java Module System

Christian Stein @sormuras · Feb 18

Replying to @trisha_gee

modules - !
mrjar - #junit5's platform ships a one, since day 1.
jshell and java Bach.java (JEP330) - use #java to build #java.
Also var, List/Set.of(), String.isBlank(), String.lines(), String.repeat()

Eugene Krylov @ekrylov_us · Feb 18

Replying to @trisha_gee

Waiting for weblogic to get certified past java 8 for main app. Modules would be one reason (we have some structuring to do in our monolith).
Another reason would be to stay current. Whenever upgrade jump gets to be big, chances of doing it get to next to 0.

JLink

Gonzalo Ortiz @gortizja · Feb 18

Replying to @trisha_gee

jlink to create (very) small docker images

And in the future?

Java 12: Switch expressions

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

And in the future?

- JEP 302: Lambda Leftovers (including underscore for param)
- JEP 305: Pattern Matching (Preview)
- Data Classes for Java
- Project Amber
- Valhalla
- Loom

The Business Doesn’t Care About Language Features

Performance

Memory Usage

Garbage Collectors

- 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 Collectors

- 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)
- Java 12: JEP 189: Shenandoah (Experimental)
- Java 12: More Updates to G1
- Java 12: More Improvements to ZGC

Cost

£$€

Catch the 6 month release train

If It Hurts, Do It More Frequently, and Bring the Pain Forward

Continuous Delivery – Jez Humble & Dave Farley

An idea from Extreme Programming (XP)

Pain?

Modularity?

Missing classes / methods

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

Maven and Gradle work with Java 12

Compile against updated JDK

…and start using the shiny new features

In Summary

Java Is Changing

…fast

Modern Java Can Help You

Performance, cost, maintenance…

Upgrade Now And Reduce Future Pain

…and keep upgrading, at least in CI

http://bit.ly/beyond-j8

@trisha_gee