The Future of Java is Today

Sharat Chander

Director, Java Platform Product Management

@Sharat_Chander

Oracle Openworld - Asia

26 – March, 2019

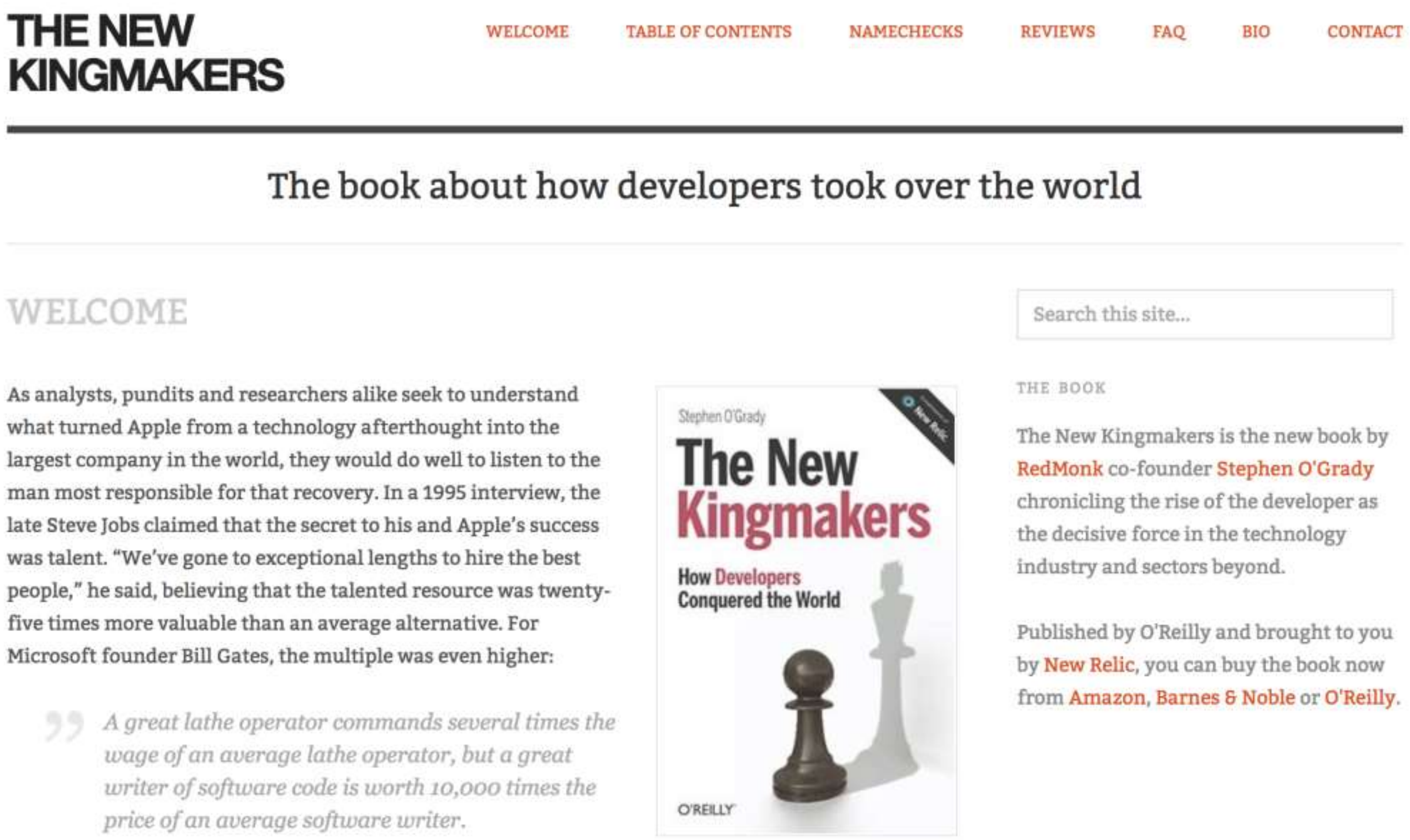
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

About me

- Part of the Java Team at Sun Microsystems
- Joined Oracle in 2010
- Team Sherpa’d the Java Evangelism Team
- Served as the JavaOne Conference chairperson
- Now part of the Java Platform product management team
- You can find me in the digital world:
  @Sharat_Chander

My preamble…



The book about how developers took over the world

WELCOME

As analysts, pundits and researchers alike seek to understand what turned Apple from a technology afterthought into the largest company in the world, they would do well to listen to the man most responsible for that recovery. In a 1995 interview, the late Steve Jobs claimed that the secret to his and Apple's success was talent. "We've gone to exceptional lengths to hire the best people," he said, believing that the talented resource was twenty-five times more valuable than an average alternative. For Microsoft founder Bill Gates, the multiple was even higher:

> *A great lathe operator commands several times the wage of an average lathe operator, but a great writer of software code is worth 10,000 times the price of an average software writer.*

THE BOOK

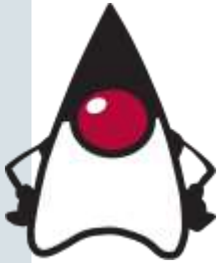
The New Kingmakers is the new book by RedMonk co-founder Stephen O'Grady chronicling the rise of the developer as the decisive force in the technology industry and sectors beyond.

Published by O'Reilly and brought to you by New Relic, you can buy the book now from Amazon, Barnes & Noble or O'Reilly.

https://thenewkingmakers.com

Is the Java ecosystem shrinking?

1995



Java 1.0

1998

Java 1.2

2002

Java 1.4

2006

Java 1.6

2014

Java 8

2018

Java 10

1997

Java 1.1

2000

Java 1.3

2004

Java 1.5

2011

Java 7

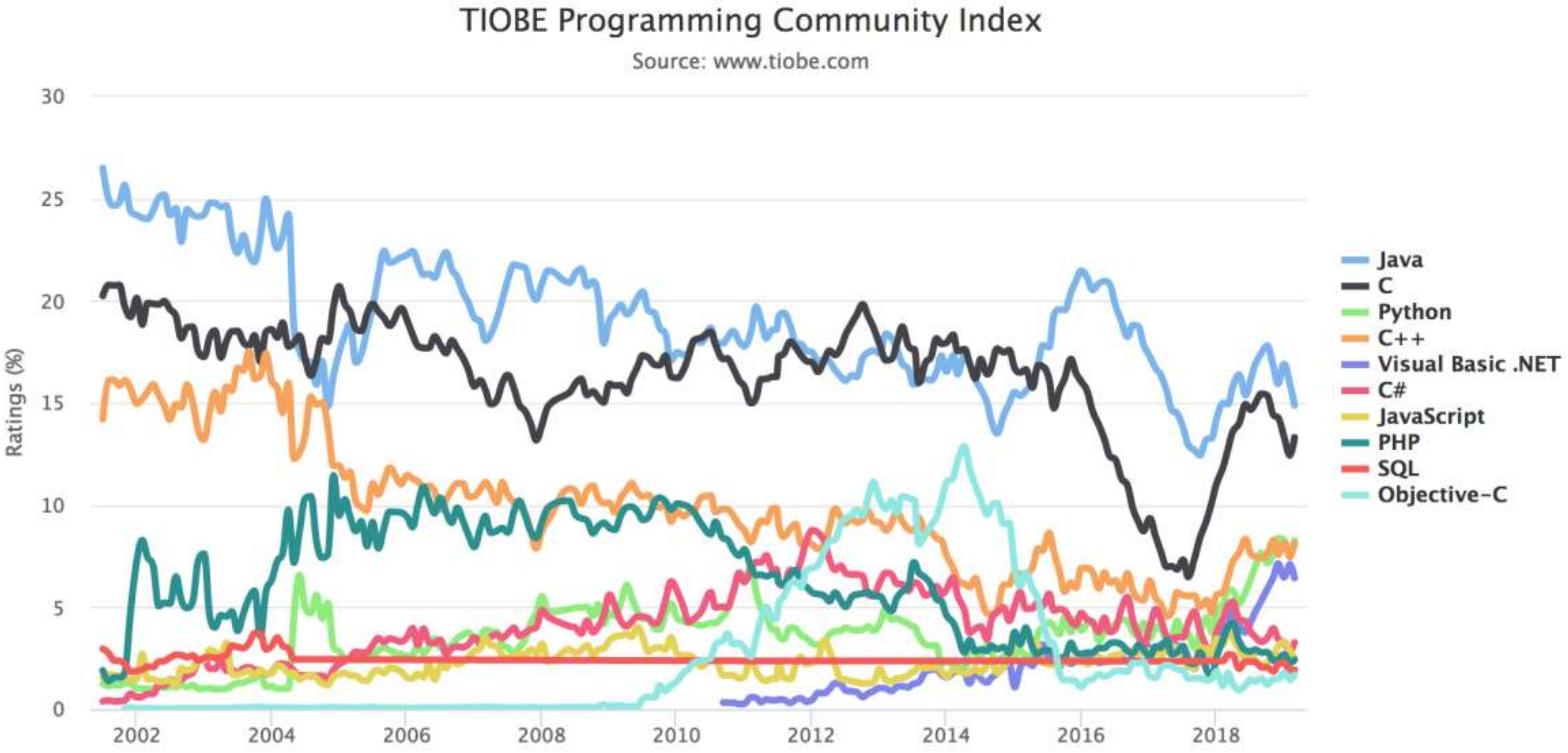
2017

Java 9

2018

Java 11

Java ORACLE


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
Python
C++
Visual Basic .NET
C#
JavaScript
PHP
SQL
Objective-C

Continuing growth

12 Million developers run Java

#1 Programming language

#1 Developer choice for the cloud

30 Billion active Java Virtual Machines

21 Billion cloud connected Java Virtual Machines

Continuing support for the Java ecosystem

Communicating, collaborating, connecting

Java Magazine	Over 250,000 subscribers, and it’s FREE!
Java User Groups	Over 350 groups worldwide…join (y)our peers
Java Champions	Over 150 members to connect with and learn from
Student Outreach	Java Foundations, Java Fundamentals, Java Programming
JCP	Over 1,000 participants helping to develop standard technical specifications for Java
jDuchess Program	Global organization for women in Java technology, currently with 550 members in over 60 countries

Preserving Java’s virtues

Ensuring Java continues to be **open** and **free**

Delivering ongoing platform completeness

Continuing to invest in quality and security

Preserving open and transparent development

Investing in developer productivity and compatibility

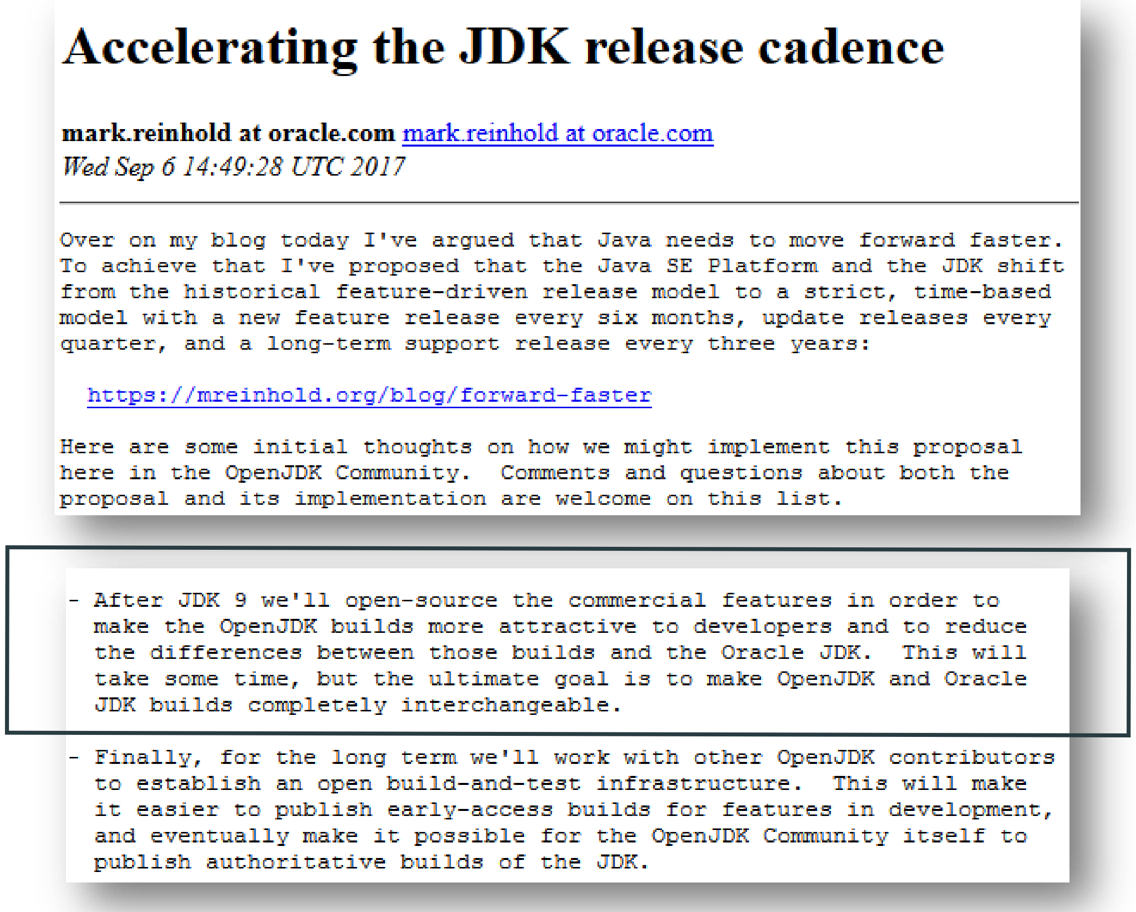
Oracle commitments for Java

- Make Java more open
- Deliver enhancements and innovation faster
- Continue support for the Java ecosystem

Is Java still free?

Moving Java Forward Faster

Keep Java Free and more open (opener?)



Accelerating the JDK release cadence

mark.reinhold at oracle.com mark.reinhold at oracle.com
Wed Sep 6 14:49:28 UTC 2017

```
Over on my blog today I've argued that Java needs to move forward faster.
To achieve that I've proposed that the Java SE Platform and the JDK shift
from the historical feature-driven release model to a strict, time-based
model with a new feature release every six months, update releases every
quarter, and a long-term support release every three years:

  https://mreinhold.org/blog/forward-faster

Here are some initial thoughts on how we might implement this proposal
here in the OpenJDK Community.  Comments and questions about both the
proposal and its implementation are welcome on this list.
```

```
  - After JDK 9 we'll open-source the commercial features in order to
    make the OpenJDK builds more attractive to developers and to reduce
    the differences between those builds and the Oracle JDK.  This will
    take some time, but the ultimate goal is to make OpenJDK and Oracle
    JDK builds completely interchangeable.

  - Finally, for the long term we'll work with other OpenJDK contributors
    to establish an open build-and-test infrastructure.  This will make
    it easier to publish early-access builds for features in development,
    and eventually make it possible for the OpenJDK Community itself to
    publish authoritative builds of the JDK.
```

- Oracle will open source commercial features
- The new OpenJDK builds will be licensed under GPL v2
  GNU General Public License Version 2 with Class Path Exception (GPL 2 with CPE)
- Oracle will now produce OpenJDK builds
- New Java feature release will be made every 6 months
- Oracle will work with other OpenJDK contributors to make the community infrastructure complete, modern and accessible

URL: http://mail.openjdk.java.net/pipermail/discuss/2017-September/004281.html

Making Java more open

What has been open-sourced

- **Application Class Data Sharing (OpenJDK 10)**
  - Enables you to place classes from the standard extensions directories and the application class path in the shared archive
- **Project ZGC (OpenJDK 11)**
  - Low latency garbage collector to support multi-terabyte heaps
- **Flight Recorder (OpenJDK 11)**
  - Collects diagnostic and profiling data about a running Java application
- **Mission Control (OpenJDK 11)**
  - Monitor and manage Java applications with minimal performance overhead

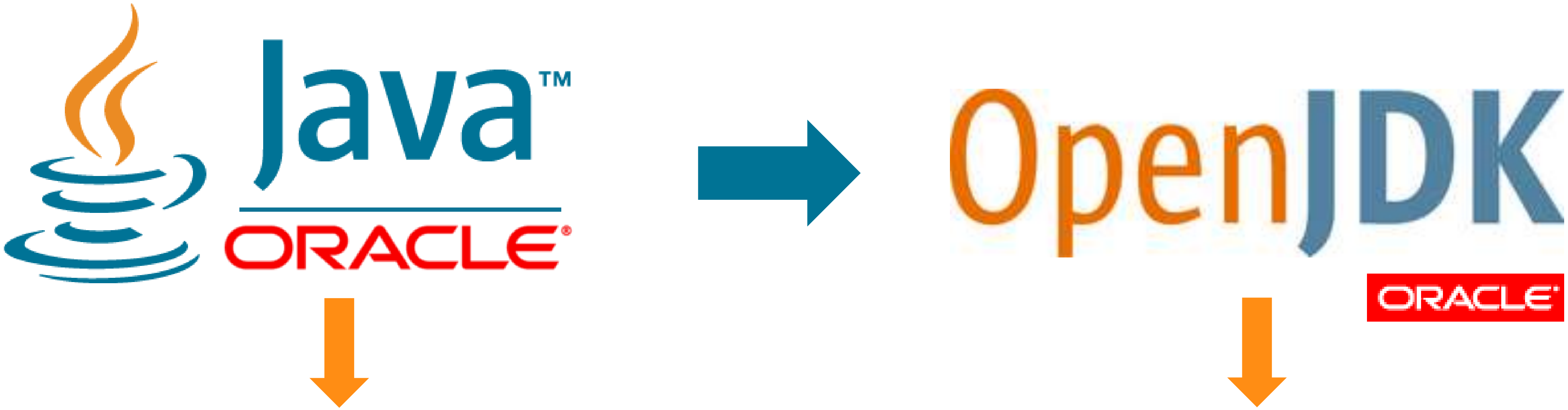
Ensuring Java is available for free

- **Previously**
  - OpenJDK: free as source
  - Oracle JDK: free binaries under BCL
  - Java SE Advanced: fee-based under commercial license

- **Currently**
  - OpenJDK: free binaries under GPLv2+CPE
  - Oracle JDK: free binaries under commercial license

Making Java more open

From Oracle JDK to Oracle OpenJDK



oracle.com/javadownload

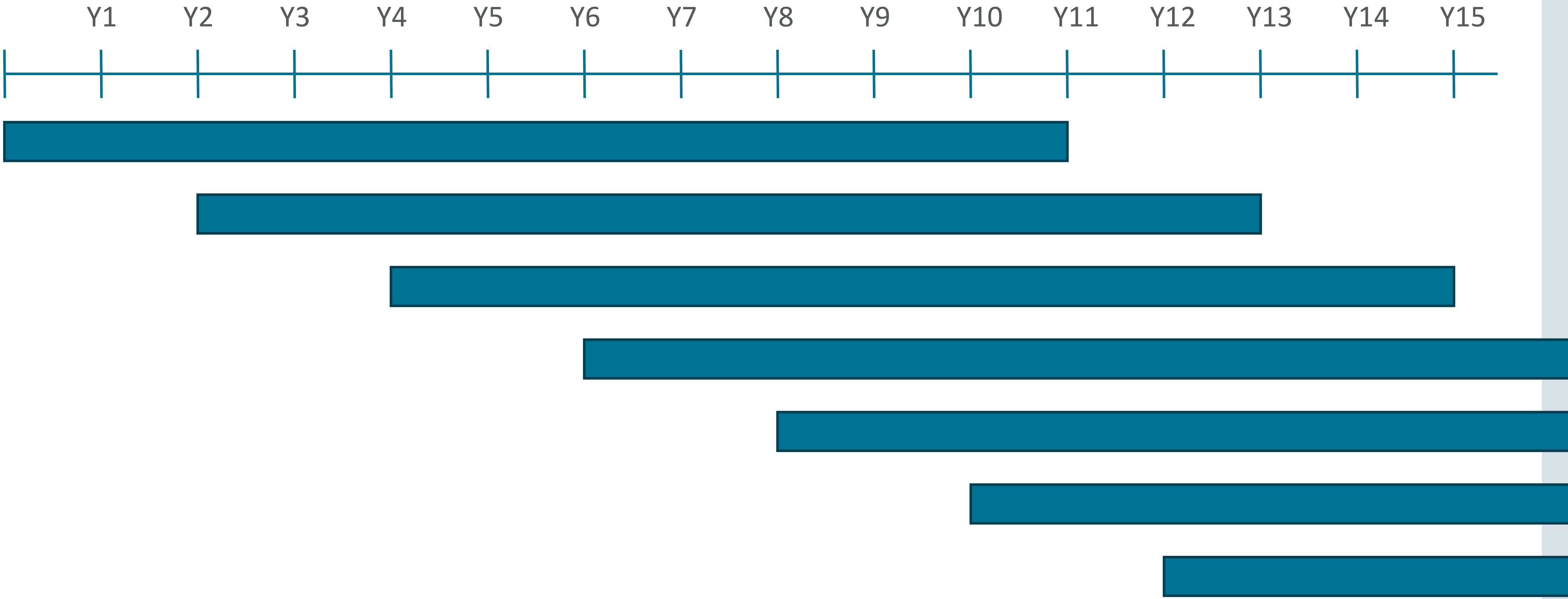
openjdk.java.net

Is Java moving slower or faster?

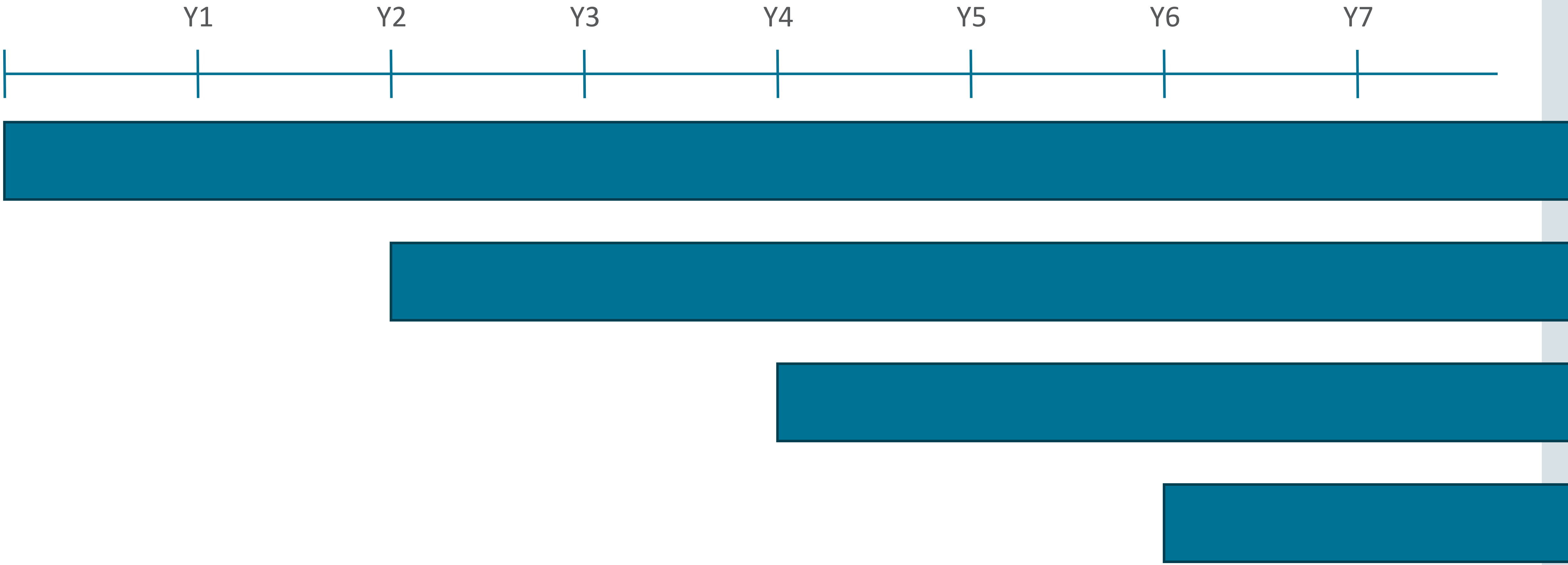
The Release Model

No more limousines, think trains!

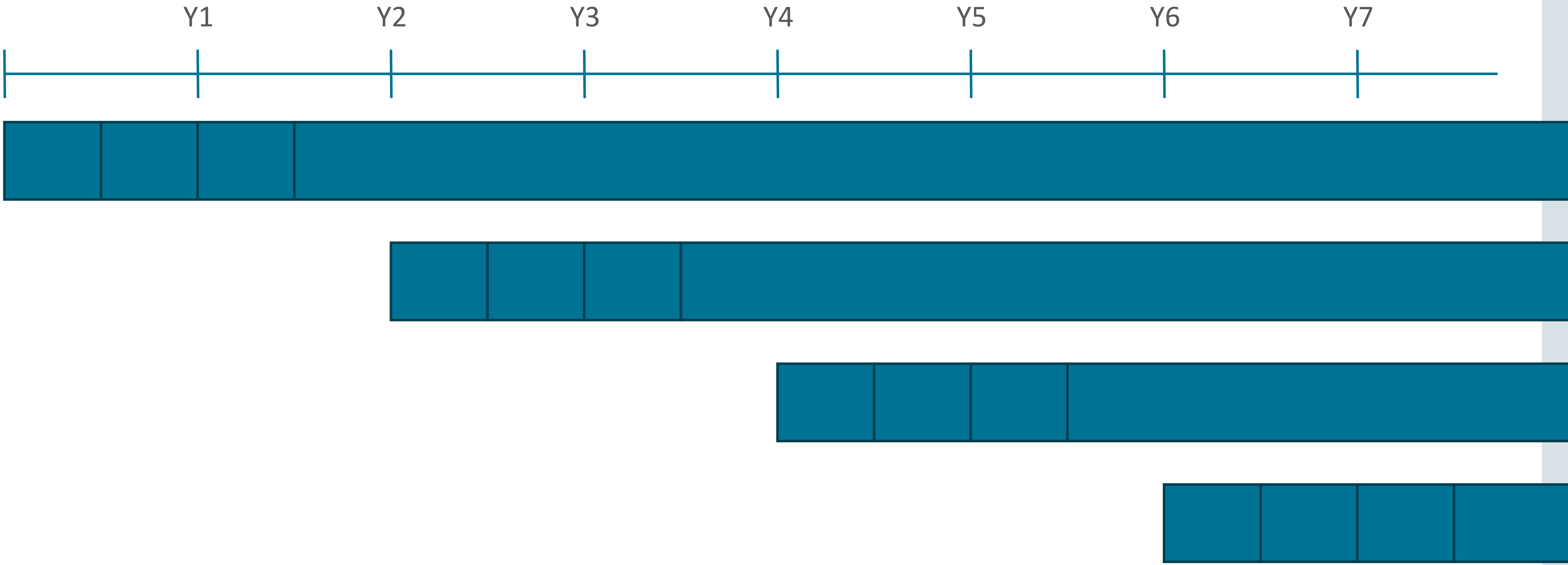
Previous JDK Release Model



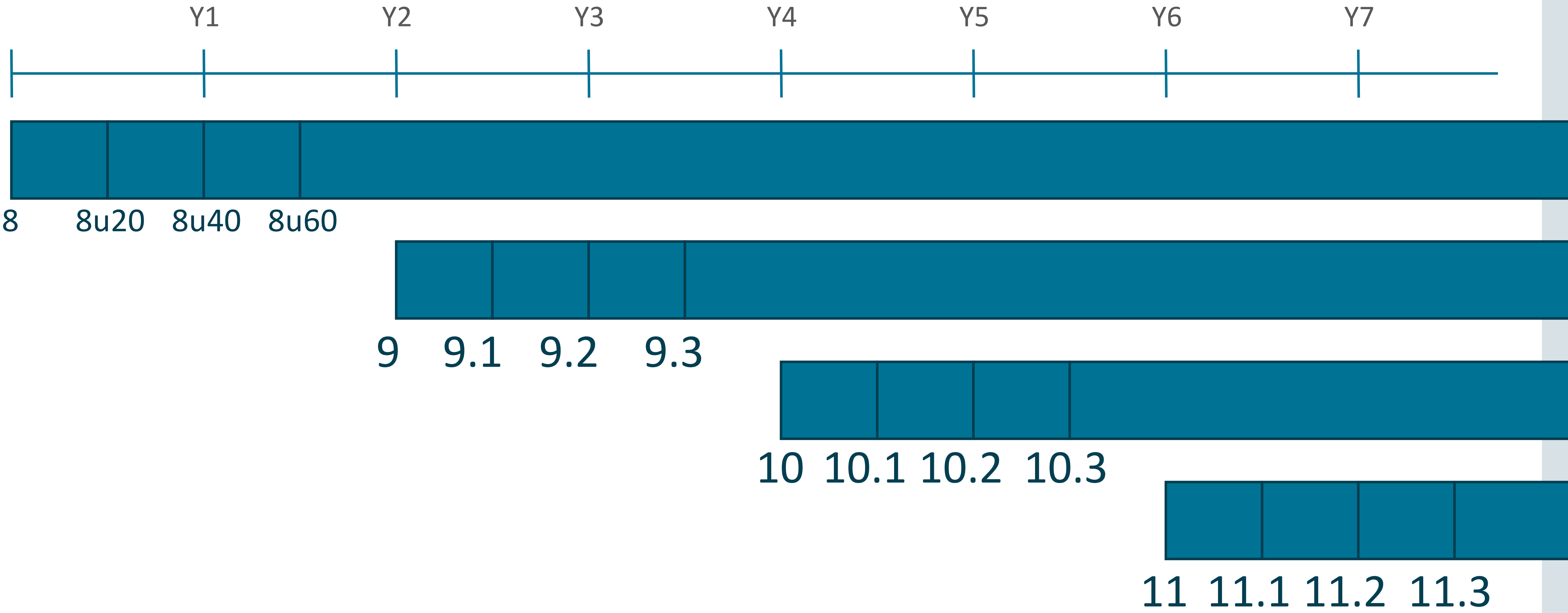
Previous JDK Release Model



Previous JDK Release Model

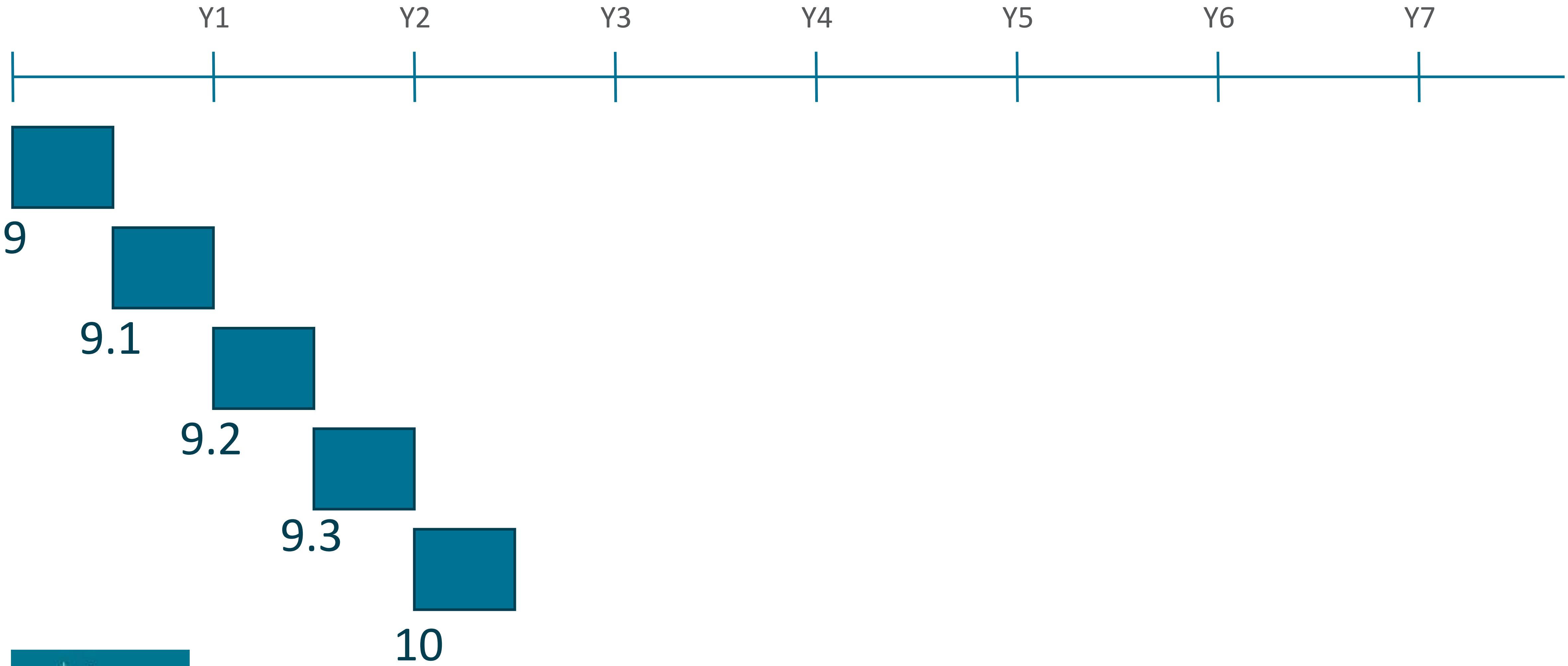


Previous JDK Release Model

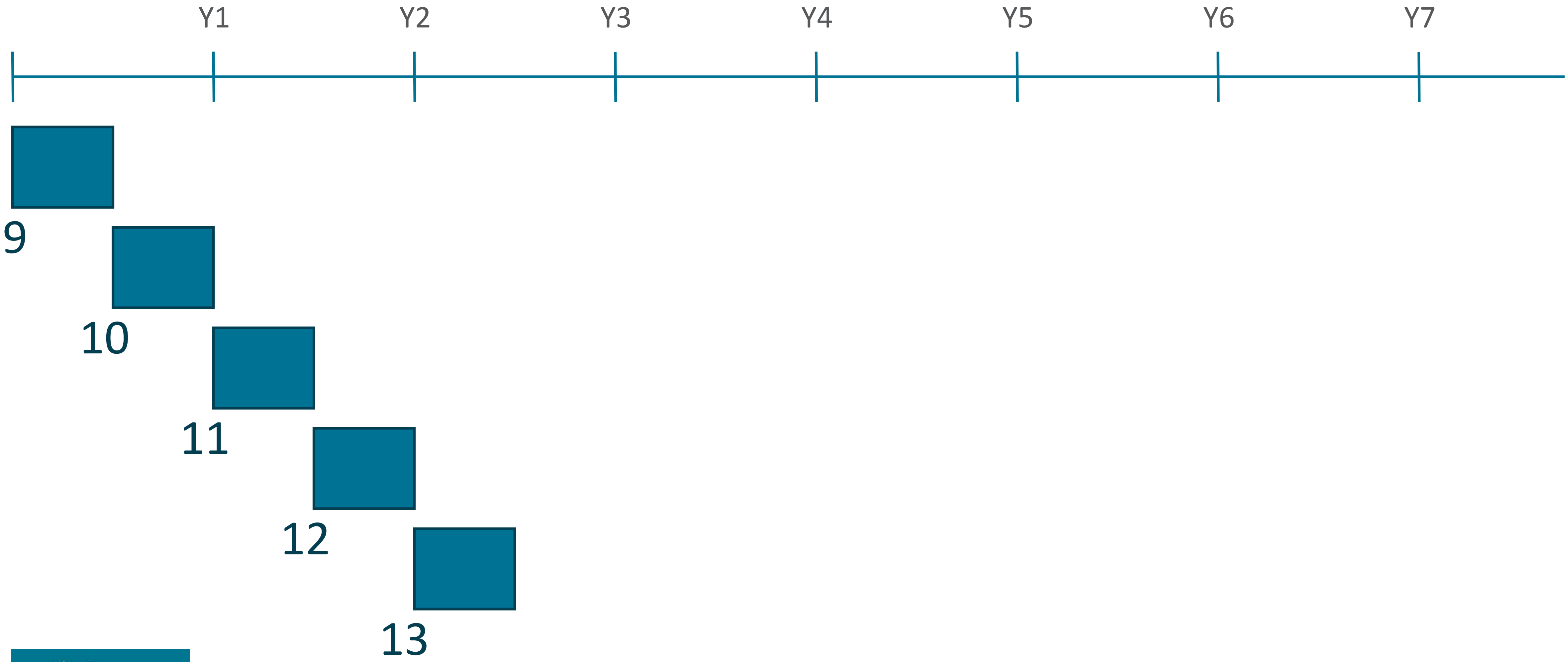


New JDK Release Model

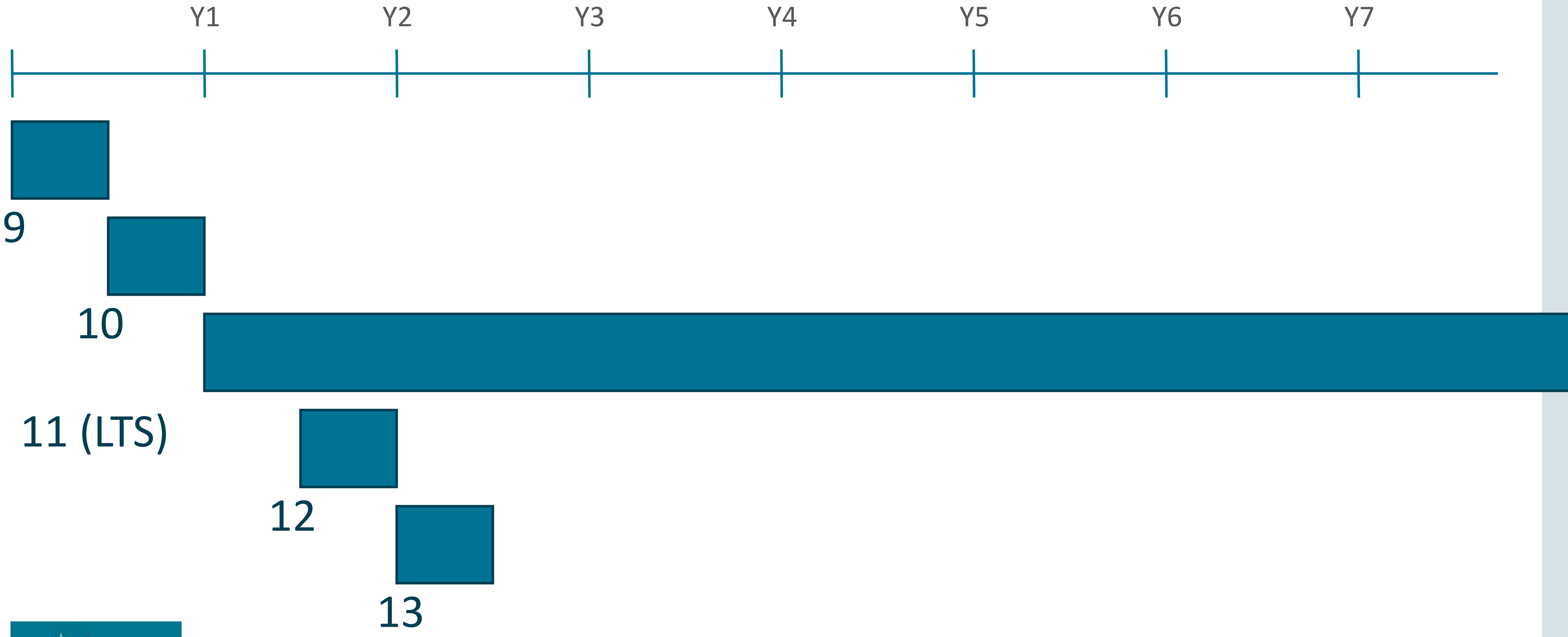
Feature releases every 6 months



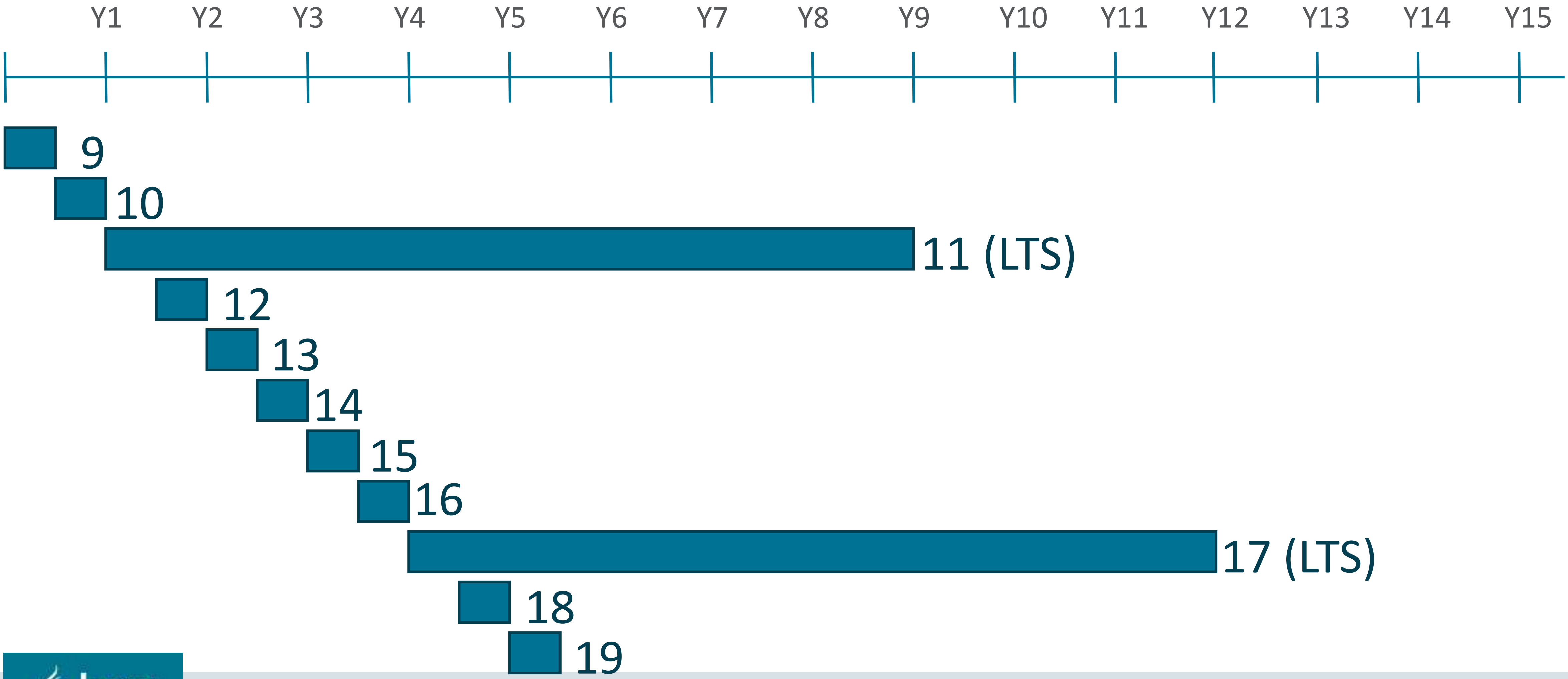
New JDK Release Model



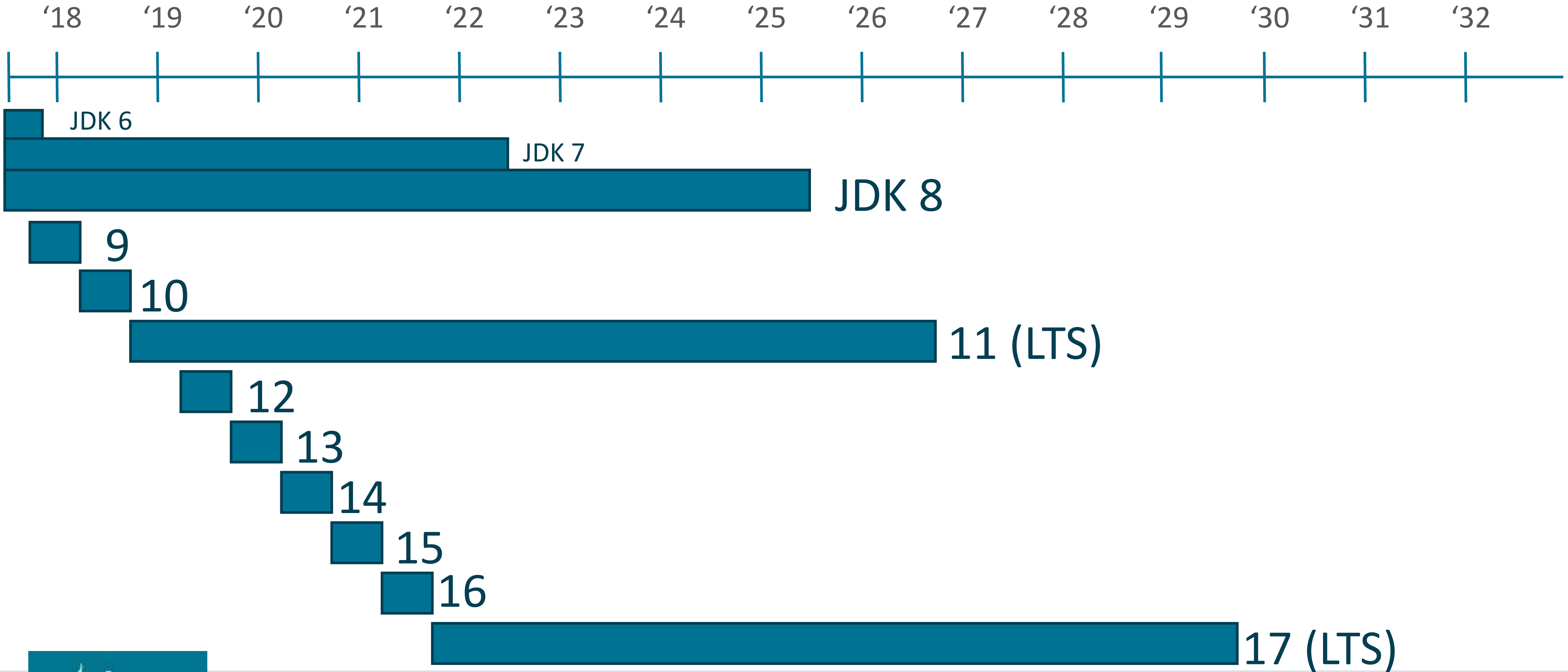
New JDK Release Model - LTS Releases



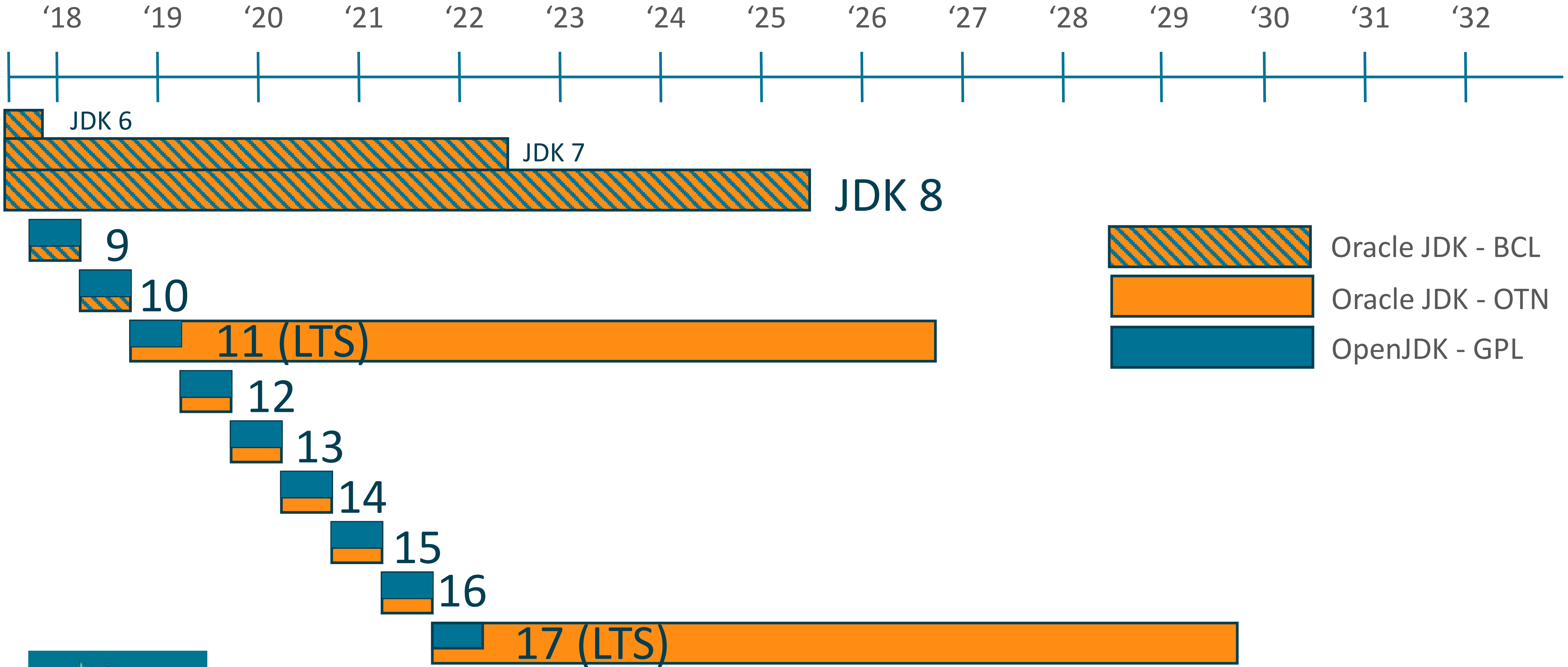
New JDK Release Model - LTS Every 3 years



New JDK Release Model – Starting with JDK 9

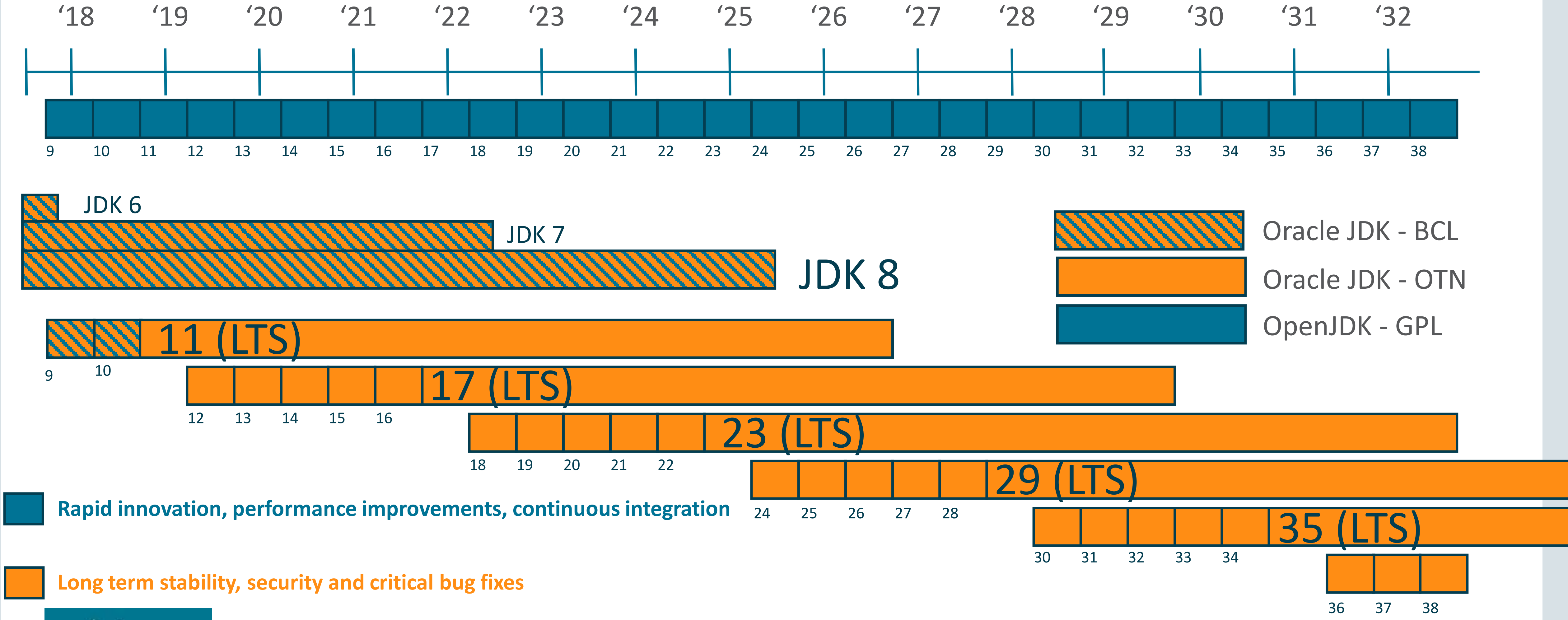


Oracle JDK & Oracle OpenJDK



New JDK Release model

6-Month Releases & LTS Every 3 Years



🟦 **Rapid innovation, performance improvements, continuous integration**

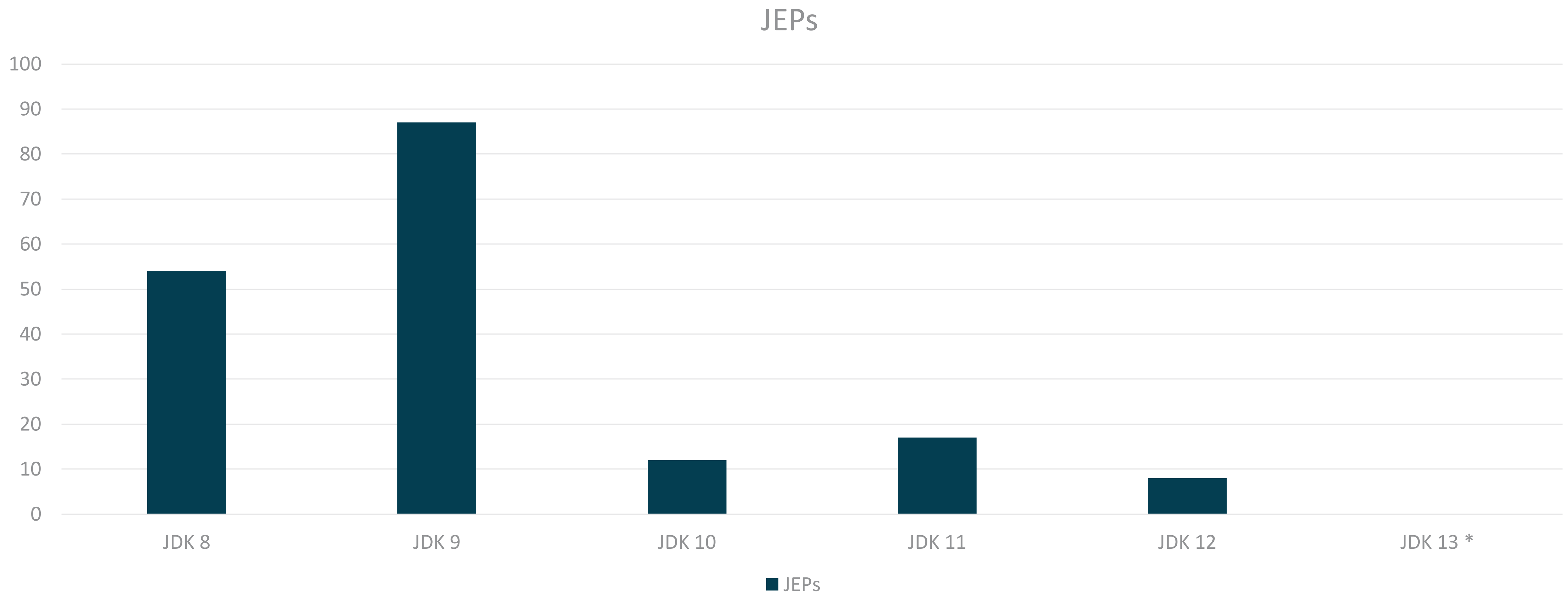
🟧 **Long term stability, security and critical bug fixes**

Comparing Legacy “Major” and “Minor” with new “Feature” Cadence

Comparing Legacy “Major” and “Minor” with new “Feature” Cadence

Don’t

Targeted JEPs per JDK Release



The Pessimist

“Each feature release will be as disruptive as the old “major” releases. Nobody will be able to keep up!”

Not really…

- Legacy major releases usually had revolutionary changes like Generics, Lambdas or JPMS, plus hundreds of smaller features and bug fixes. A major release represented the culmination of 3+ years of changes
- Feature releases have more digestible feature changes. It represents “the last 6 months” of changes

The Optimist

Each "feature" release will be trivial to update.
It will be exactly like the old "minor" updates
which most people didn't even notice.

Also not correct…

- Minor releases did break applications from time to time….
- Feature releases can (and do) deprecate/remove features, support different OS/Archs, introduce new functionality.
- Feature releases have spec changing features which may require tool chain adoption.

Realist

"Most feature releases will be easy to uptake, but this is relatively new and there's still some risks."

- Less pressure to rush a feature into a release -> more stable releases
  - Eg, Raw String Literals
- More impactful features can be phased in
  - Eg, Switch Expressions preview in JDK 12 -> Pattern Matching in JDK ??
- Once in a while a feature release <u>will</u> require that you update code
  - But it won't be one feature release disrupting ALL programs
- We have introduced new tools (jdeps, jdeprscan) and processes (enhanced deprecation) to minimize surprises

Delivering enhancements and innovation more rapidly

Predictable 6-month release cadence

Incremental improvement

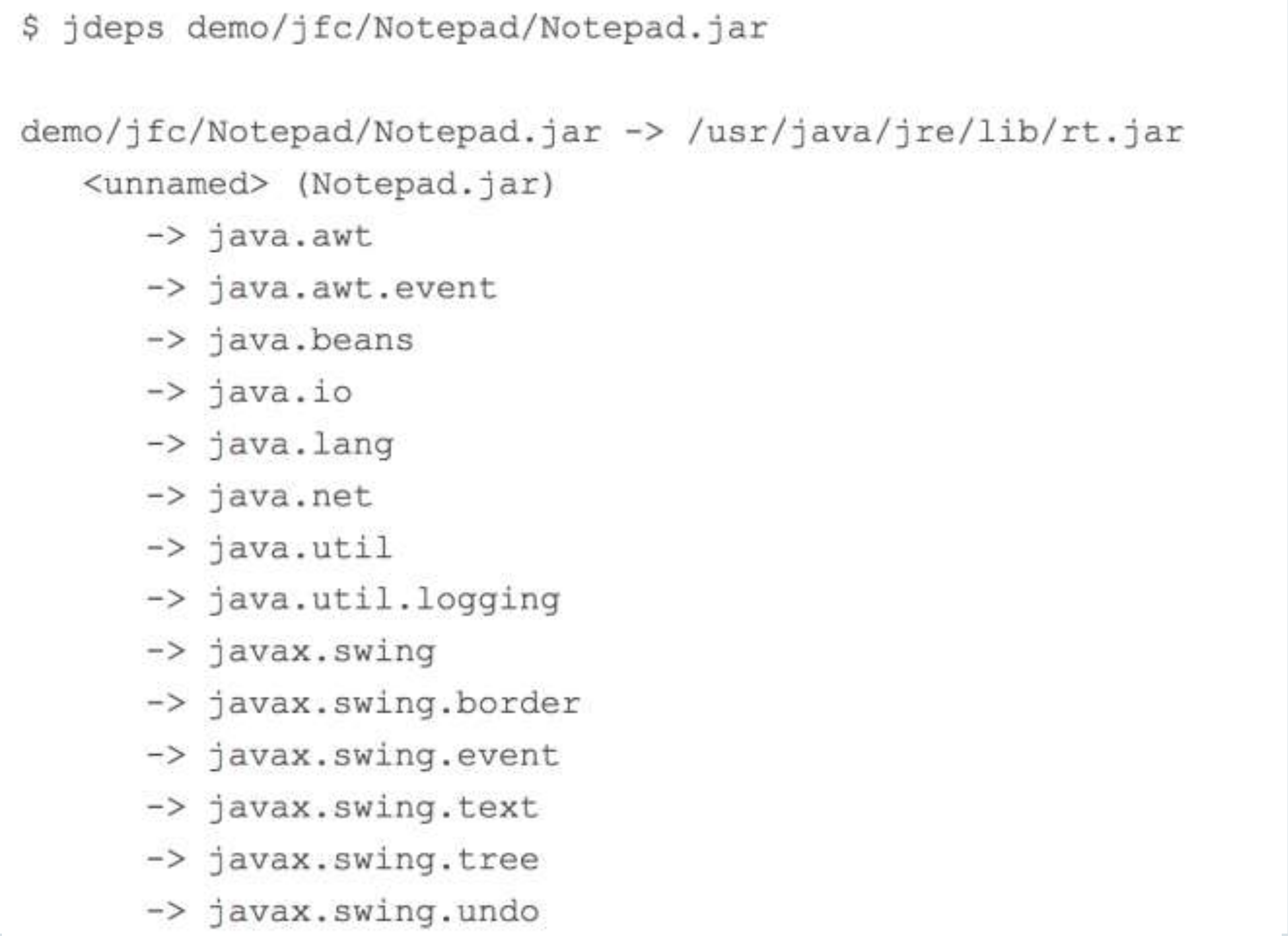
Get access to new features sooner

No more disruptive major release

New tools for new process

JDK 8: jdeps Java class dependency analyzer

- Shows the package-level or class-level dependencies of Java class files



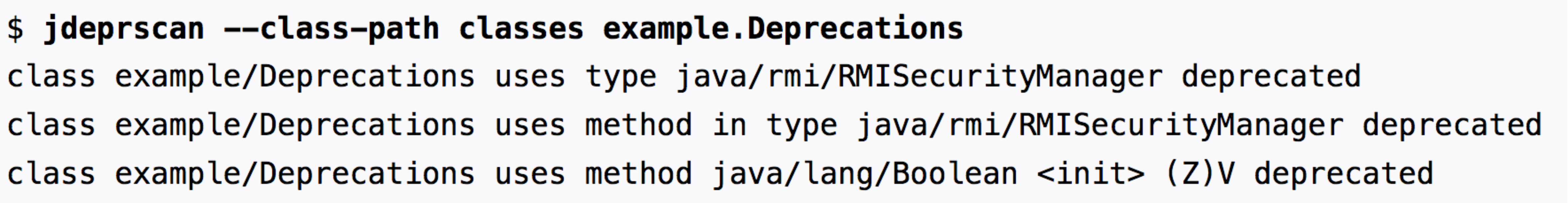
```
$ jdeps demo/jfc/Notepad/Notepad.jar

demo/jfc/Notepad/Notepad.jar -> /usr/java/jre/lib/rt.jar
   <unnamed> (Notepad.jar)
      -> java.awt
      -> java.awt.event
      -> java.beans
      -> java.io
      -> java.lang
      -> java.net
      -> java.util
      -> java.util.logging
      -> javax.swing
      -> javax.swing.border
      -> javax.swing.event
      -> javax.swing.text
      -> javax.swing.tree
      -> javax.swing.undo
```

New tools for new process

JDK 9: jdeprscan

- scans a jar file for uses of deprecated API elements
- ***–for-removal*** Limits scanning or listing to APIs that are deprecated for removal



```
$ jdeprscan --class-path classes example.Deprecations
class example/Deprecations uses type java/rmi/RMISecurityManager deprecated
class example/Deprecations uses method in type java/rmi/RMISecurityManager deprecated
class example/Deprecations uses method java/lang/Boolean <init> (Z)V deprecated
```

New tools for new process

JEP 11: Incubator Modules (introduced with JDK 9)

- Incubator modules are a means of putting non-final APIs in the hands of developers while the APIs progress towards either finalization or removal in a future release

New tools for new process

JEP 12: Preview Language and VM Features

- A preview language or VM feature is a new feature of the Java SE Platform that is fully specified, fully implemented, and yet impermanent

- JDK 12 shipped the first preview feature

```
--enable-preview
--release $N
```

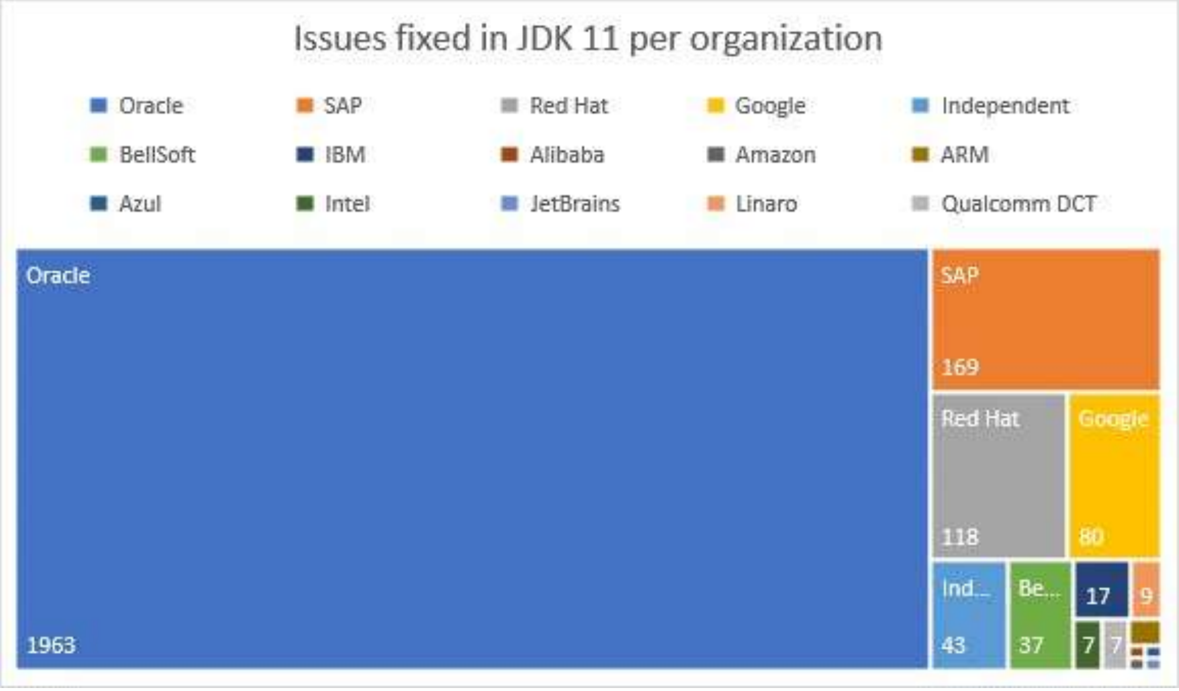
Is Oracle the only Java contributor?

Building JDK 11 together

Thank you to the many contributions in the OpenJDK Community

2,468 JIRA issues marked as fixed in JDK 11

Over 500 were contributed by individuals and developers working for other organizations



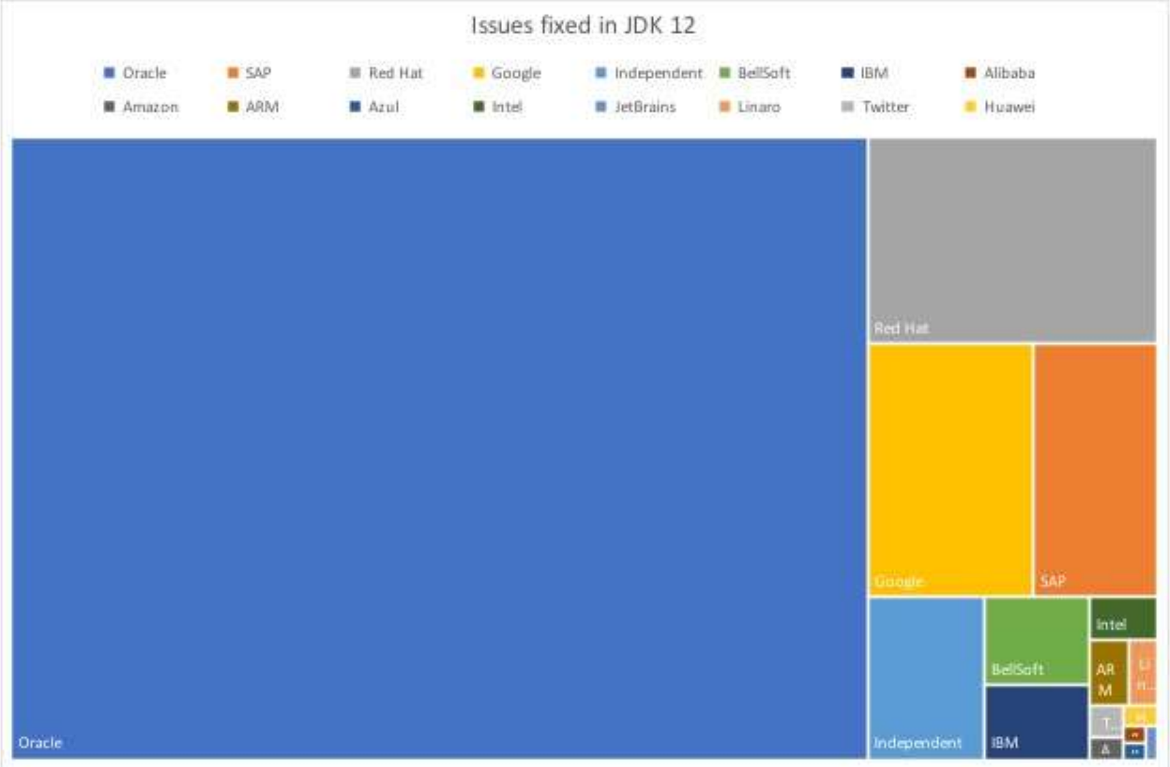
blogs.oracle.com/java-platform-group/building-jdk-11-together

Building JDK 12 together

Thank you to the many contributions in the OpenJDK Community

1,919 JIRA issues marked as fixed in JDK 12

Over 450 were contributed by individuals and developers working for other organizations



https://blogs.oracle.com/java-platform-group/the-arrival-of-java-12

What if I need commercial support?

Enterprise long-term support

New simple Oracle JDK monthly subscription support offering for desktop, server and cloud deployments (pay as you use)

Low-cost, predictable pricing

- $25.00 (USD) per month per processor (server) **or LESS**

- $2.50 (USD) per month per user (desktop) **or LESS**

- Significant discount available for large deployment support requirements (i.e. 50% discount for 10k-20K processor units)

You have a choice:

- Oracle support for specific releases
- Additional support options from other vendors

oracle.com/java/java-se-subscription

What are some new opportunities for Java?

Valhalla: https://wiki.openjdk.java.net/display/valhalla/Main

Panama: https://openjdk.java.net/projects/panama/

Portola: https://wiki.openjdk.java.net/display/portola/Main

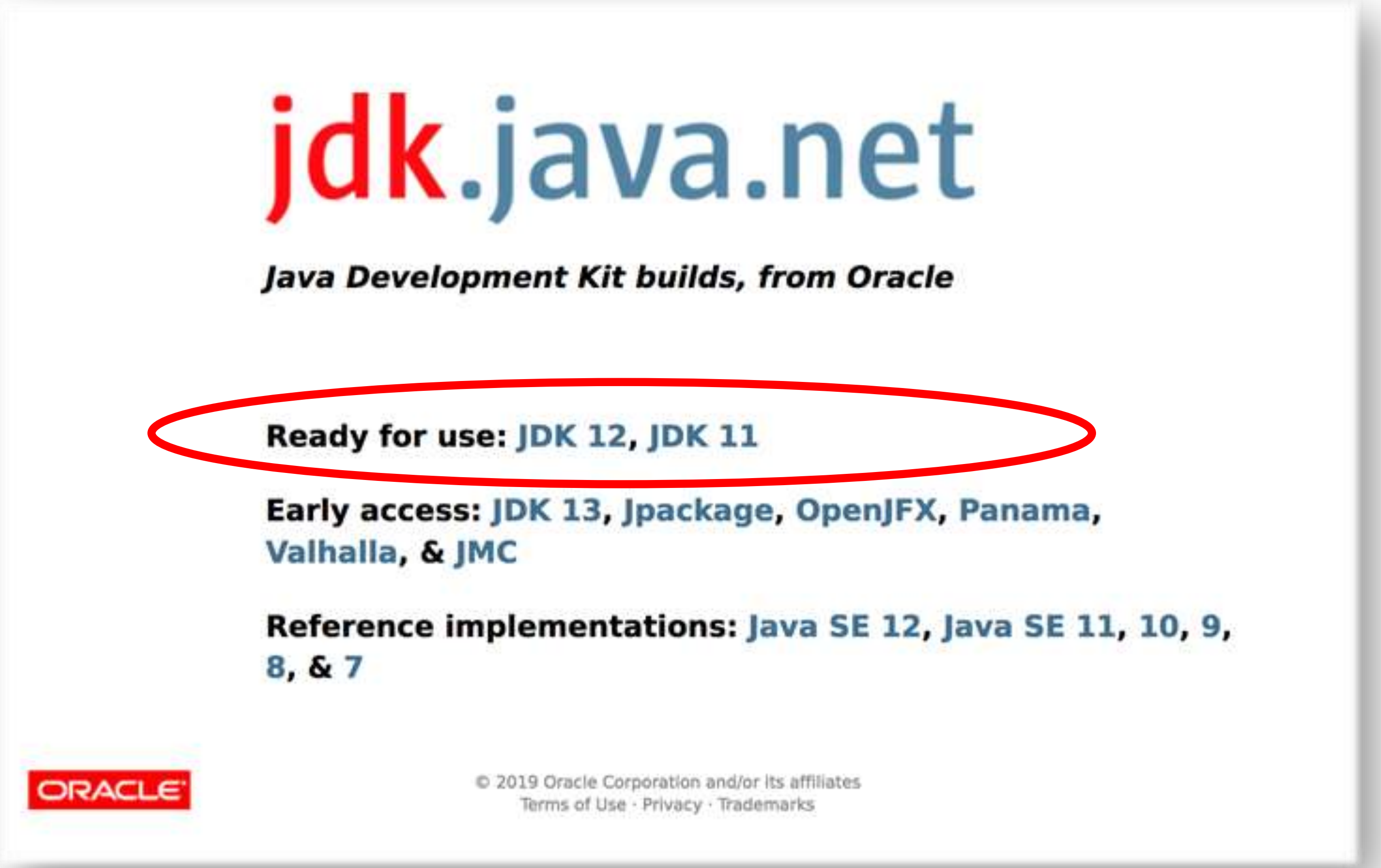
Amber: https://wiki.openjdk.java.net/display/amber/Main

Loom: https://wiki.openjdk.java.net/display/loom/Main

Skara: https://openjdk.java.net/projects/skara/

OpenJDK EA releases

- Upcoming Feature Releases
- Special Projects EAs



jdk.java.net

Java Development Kit builds, from Oracle

Ready for use: JDK 12, JDK 11

Early access: JDK 13, Jpackage, OpenJFX, Panama, Valhalla, & JMC

Reference implementations: Java SE 12, Java SE 11, 10, 9, 8, & 7

Summary

- The Java platform development on OpenJDK is becoming **more open**
- Oracle is simplifying Java licensing by transitioning out of the BCL "dual purpose" license to GPLv2+CPE
- Oracle introduced a new Java subscription offering to make Java support affordable and predictable for everyone
- Beyond Java 12, we have a solid technical roadmap

JOIN and become an **OpenJDK CONTRIBUTOR**

https://openjdk.java.net

STAY INFORMED

https://blogs.oracle.com/java-platform-group

CONNECT WITH US

@OpenJDK

@gsaab @mreinhold @BrianGoetz @MikaelVidstedt

@BTraTra @DonaldOJDK @Sharat_Chander

@robilad @aureliog @SimmsUpNorth