Tip & Tail
The Release Model for Java

Alex Buckley
Java Platform Group
Oracle
June 2025

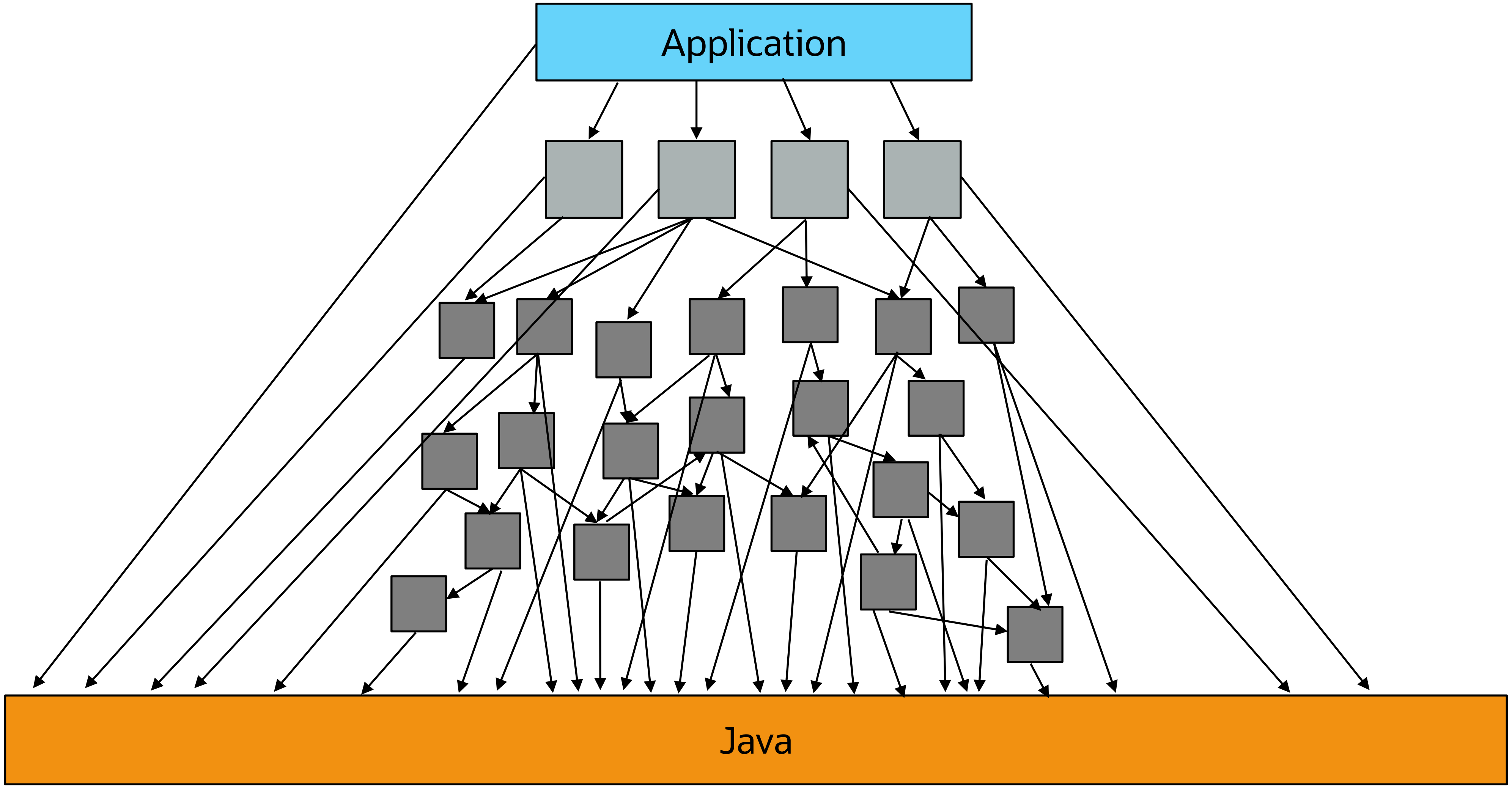
Release model:

A set of rules for evolving and publishing software

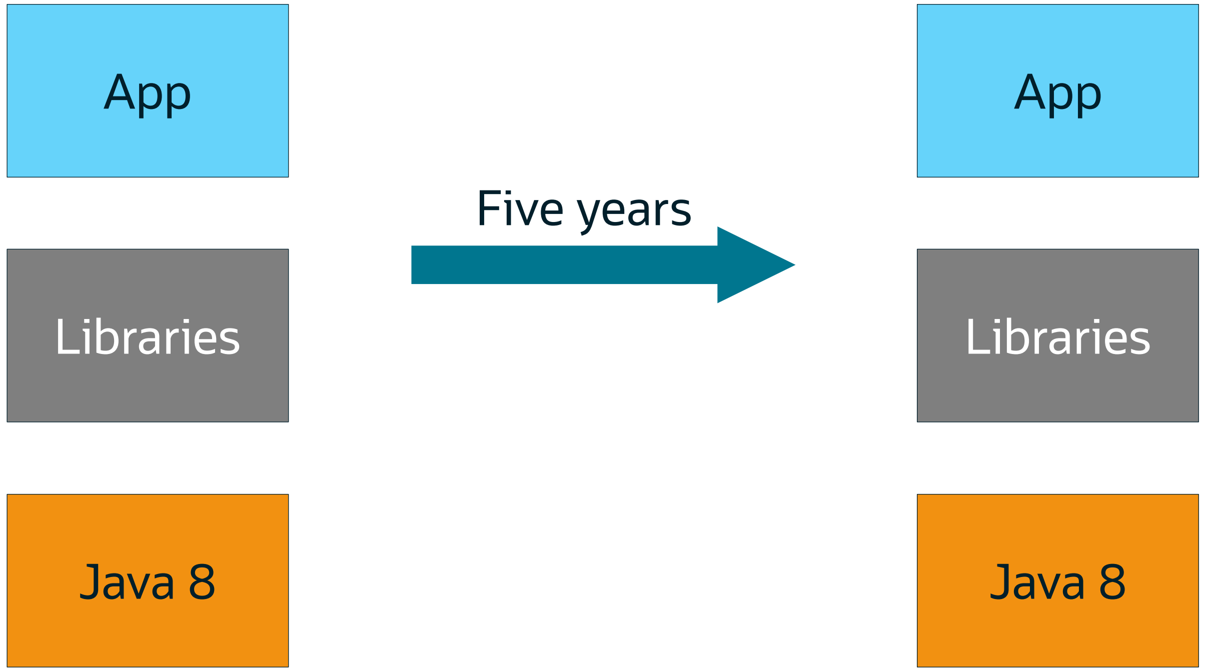
Agenda

- How do you use Java?
- The "One Size Fits All" Release Model
- The "Tip & Tail" Release Model
- Tip & Tail in the JDK
- Tip & Tail in the Java Ecosystem

How do you use Java?

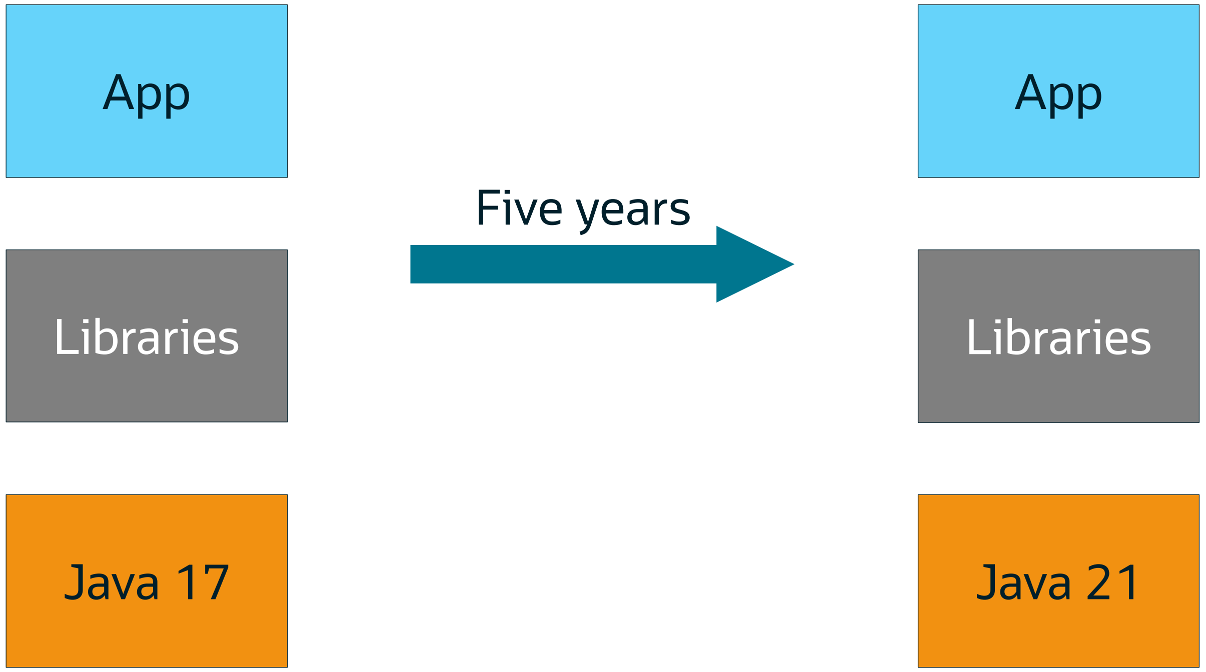

Application
Java

How do you use Java?
Scenario 1

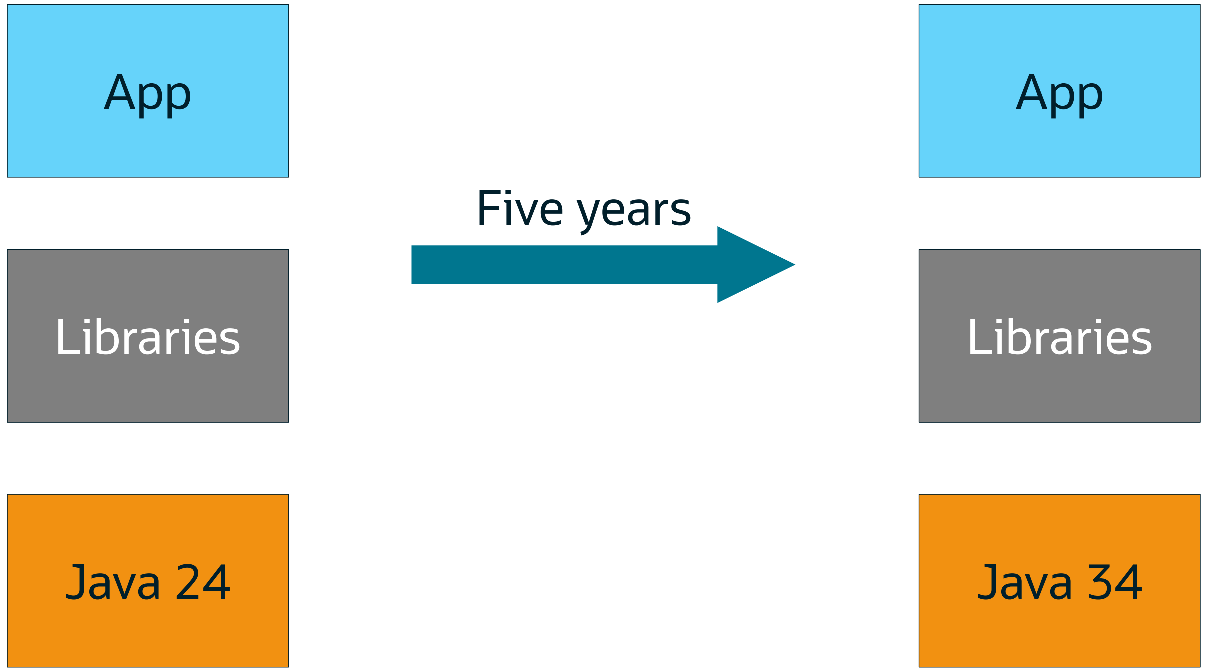


How do you use Java?

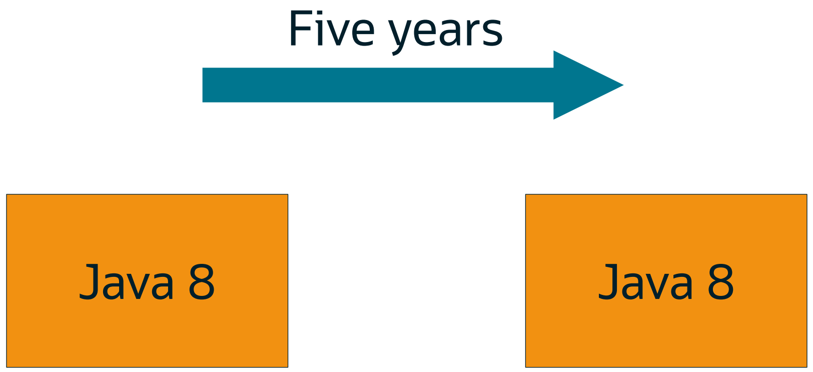
Scenario 2



How do you use Java?
Scenario 3



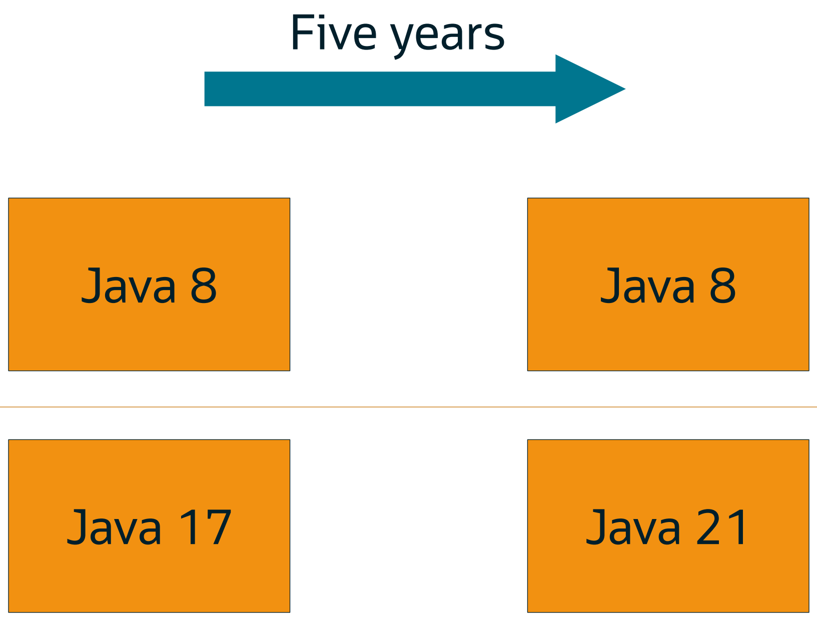
What do you value?



Stability

Upgrade as little as possible

What do you value?



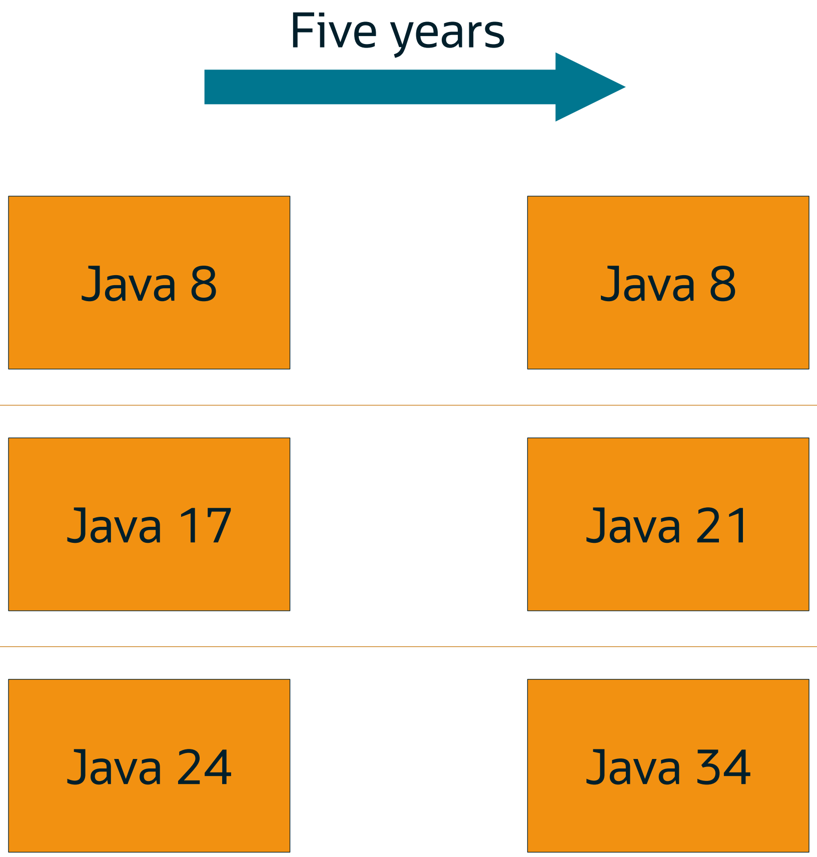
Stability

Upgrade as little as possible

Predictability

Upgrade when benefits outweigh costs

What do you value?



Stability

Upgrade as little as possible

Predictability

Upgrade when benefits outweigh costs

Functionality

Upgrade whenever higher productivity or better performance is available

What do you want from libraries and the JDK?

What you value most	New functionality Want?	Need?
Stability Upgrade as little as possible	✗	✗
Predictability Upgrade when benefits outweigh costs	✓	✗
Functionality Upgrade whenever higher productivity or better performance is available	✓	✓

What do you want from libraries and the JDK?

What you value most	New functionality Want?	 Need?	Bug fixes Want?	 Need?
Stability Upgrade as little as possible	✗	✗	✓	✗
Predictability Upgrade when benefits outweigh costs	✓	✗	✓	✓
Functionality Upgrade whenever higher productivity or better performance is available	✓	✓	✓	✓

What do you want from libraries and the JDK?

What you value most	New functionality		Bug fixes		Security patches	
	Want?	Need?	Want?	Need?	Want?	Need?
Stability Upgrade as little as possible	✗	✗	✓	✗	✓	✓
Predictability Upgrade when benefits outweigh costs	✓	✗	✓	✓	✓	✓
Functionality Upgrade whenever higher productivity or better performance is available	✓	✓	✓	✓	✓	✓

$$\text{Application developers} \left.\vphantom{\begin{matrix}a\\b\\c\end{matrix}}\right\} \begin{matrix}\text{value}\\\text{want}\\\text{need}\end{matrix} \quad \text{different things}$$

One Size Fits All

One-Size-Fits-All

$$\left.\begin{array}{l}\text{New features}\\ \text{Functional enhancements}\\ \text{Bug fixes}\\ \text{Security patches}\\ \text{Performance improvements}\end{array}\right\} \quad \text{In every release.}$$

Users must upgrade to the latest release to get what they want and need.

Only the latest release is "the good one".

One-Size-Fits-All involves a baseline version of Java

One-Size-Fits-All: Bad for Users who value Functionality

Modules (9)

Launch source programs with `java` (11)

`var` keyword (10)

Helpful NullPointerExceptions (14)

HTTP Client API (11)

Text Blocks (15)

Records (16)

Sealed Classes (17)

UTF-8 By Default (18)

Code Snippets in Javadoc (18)

Virtual Threads (21)

Markdown in Javadoc (23)

Sequenced Collections (21)

Stream Gatherers (24)

Foreign Function & Memory API (22)

Class-File API (24)

One-Size-Fits-All: Bad for Users who value Stability

As time goes by, the library gains new features, enhancements, bug fixes, security patches, etc.

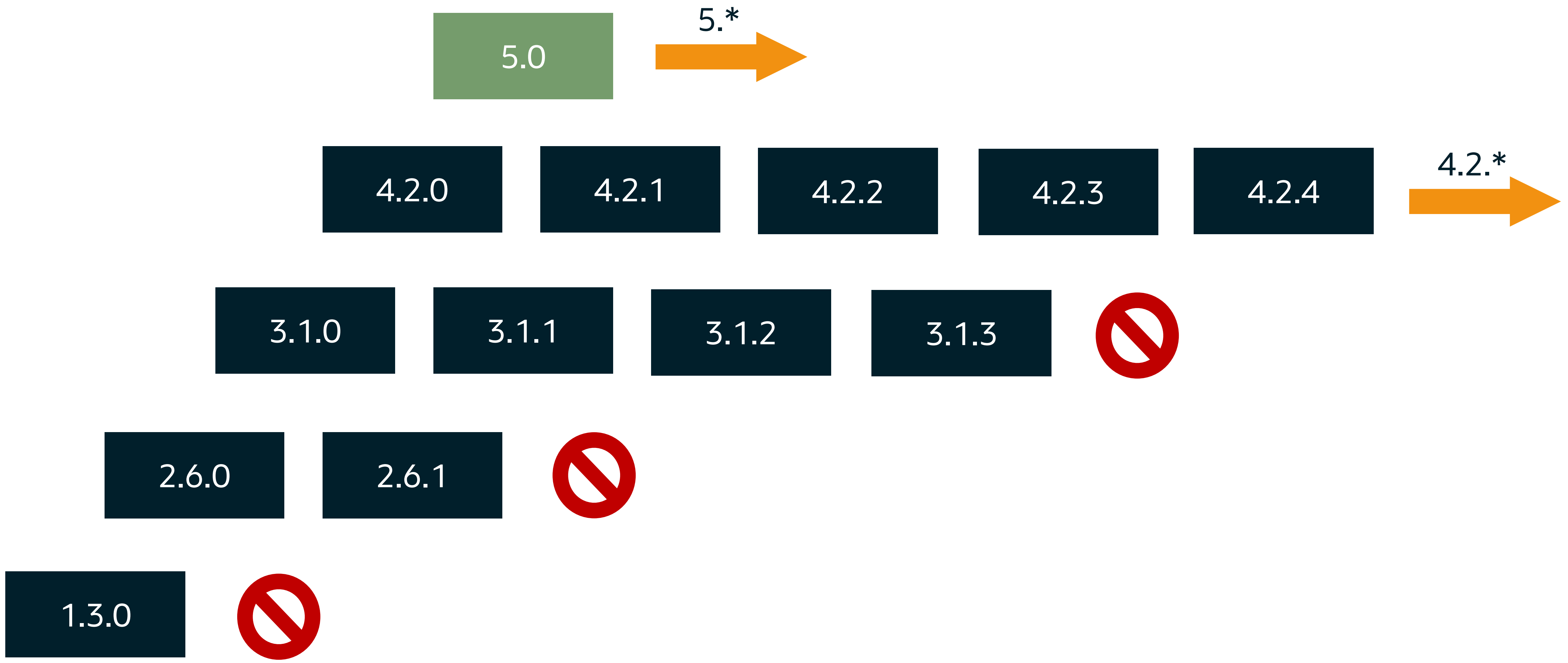
None of these are backported to older versions!

Users who value stability don't want to upgrade because they won't use new features and may rely on features that have been removed.

But *eventually*, they must upgrade to get critical bug fixes and security patches!

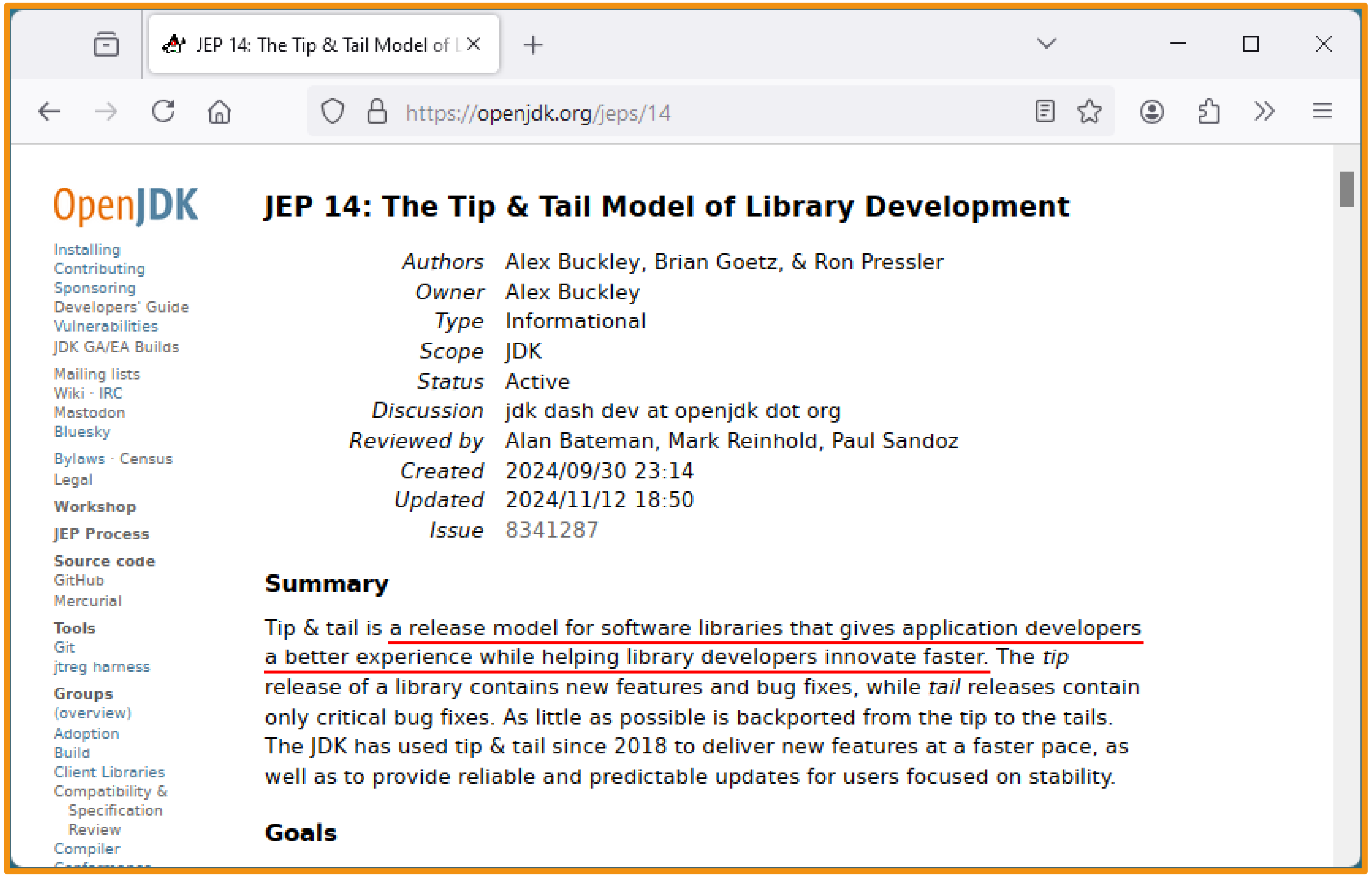
Upgrading to a new library version may also mean upgrading to a new JDK ☹

From One-Size-Fits-All to Multiple Release Trains



Tip & Tail

Tip & Tail is a release model that gives application developers a better experience while helping library developers innovate faster



JEP 14: The Tip & Tail Model of L

https://openjdk.org/jeps/14

OpenJDK

Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds

Mailing lists
Wiki · IRC
Mastodon
Bluesky

Bylaws · Census
Legal

Workshop

JEP Process

Source code
GitHub
Mercurial

Tools
Git
jtreg harness

Groups
(overview)
Adoption
Build
Client Libraries
Compatibility & Specification Review
Compiler

JEP 14: The Tip & Tail Model of Library Development

Authors	Alex Buckley, Brian Goetz, & Ron Pressler
Owner	Alex Buckley
Type	Informational
Scope	JDK
Status	Active
Discussion	jdk dash dev at openjdk dot org
Reviewed by	Alan Bateman, Mark Reinhold, Paul Sandoz
Created	2024/09/30 23:14
Updated	2024/11/12 18:50
Issue	8341287

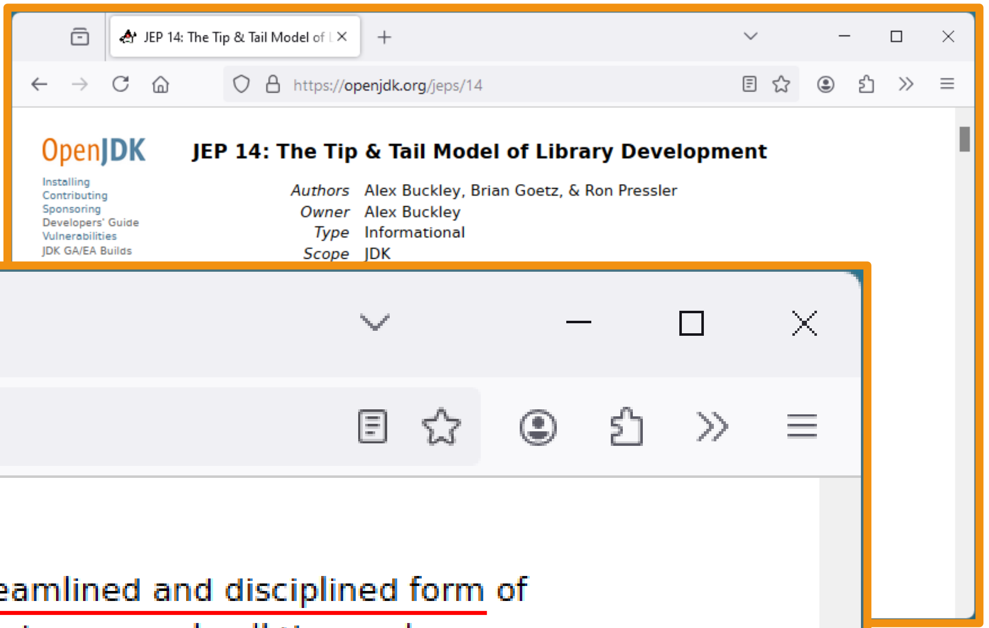
Summary

Tip & tail is a release model for software libraries that gives application developers a better experience while helping library developers innovate faster. The *tip* release of a library contains new features and bug fixes, while *tail* releases contain only critical bug fixes. As little as possible is backported from the tip to the tails. The JDK has used tip & tail since 2018 to deliver new features at a faster pace, as well as to provide reliable and predictable updates for users focused on stability.

Goals

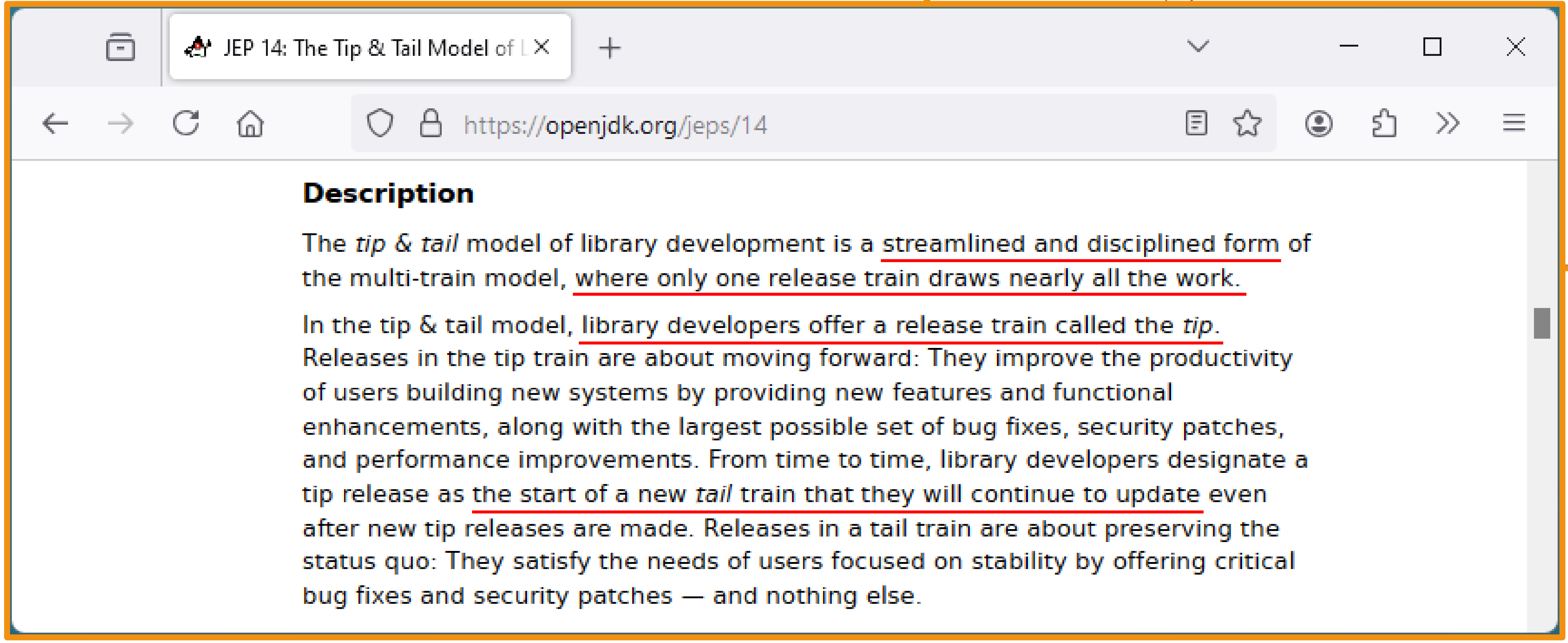
https://openjdk.org/jeps/14

Tip & Tail



JEP 14: The Tip & Tail Model of Library Development

Authors	Alex Buckley, Brian Goetz, & Ron Pressler
Owner	Alex Buckley
Type	Informational
Scope	JDK



JEP 14: The Tip & Tail Model of L
https://openjdk.org/jeps/14

Description

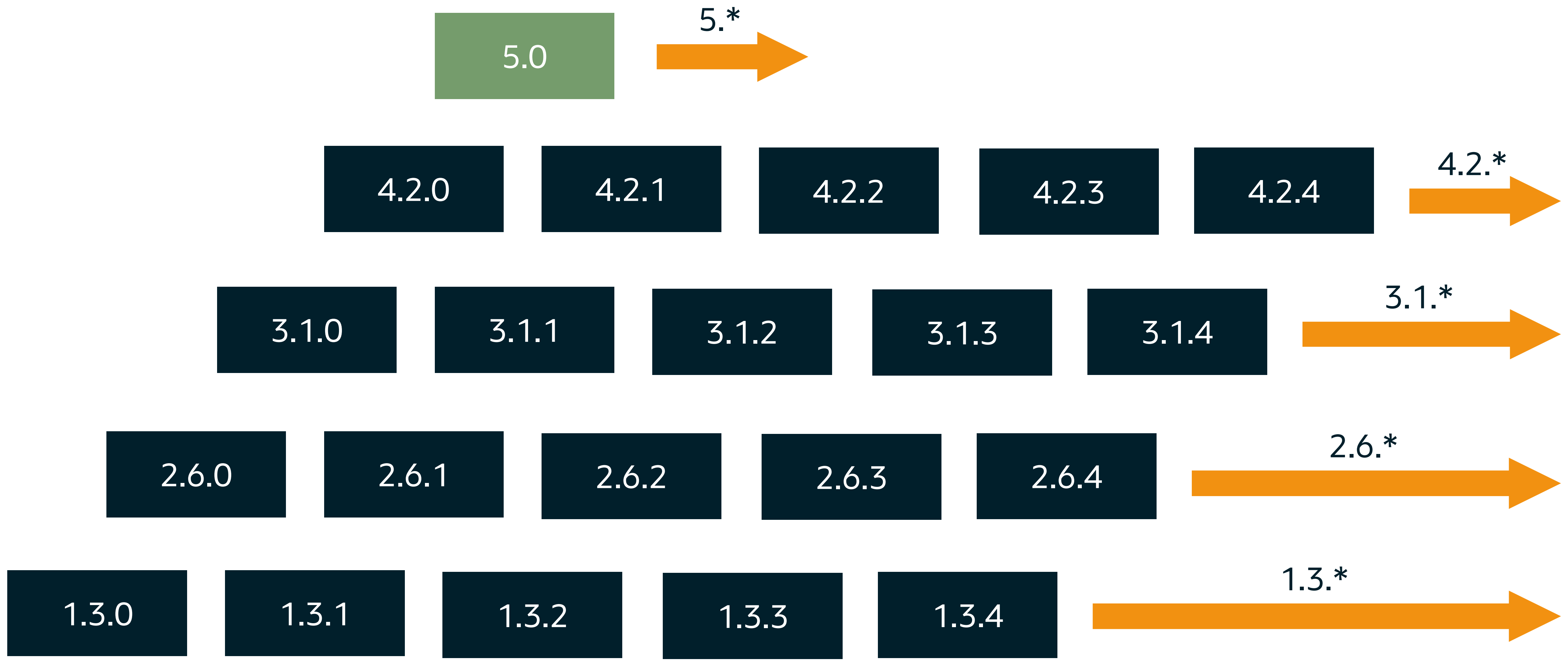
The *tip & tail* model of library development is a streamlined and disciplined form of the multi-train model, where only one release train draws nearly all the work.

In the tip & tail model, library developers offer a release train called the *tip*. Releases in the tip train are about moving forward: They improve the productivity of users building new systems by providing new features and functional enhancements, along with the largest possible set of bug fixes, security patches, and performance improvements. From time to time, library developers designate a tip release as the start of a new *tail* train that they will continue to update even after new tip releases are made. Releases in a tail train are about preserving the status quo: They satisfy the needs of users focused on stability by offering critical bug fixes and security patches — and nothing else.

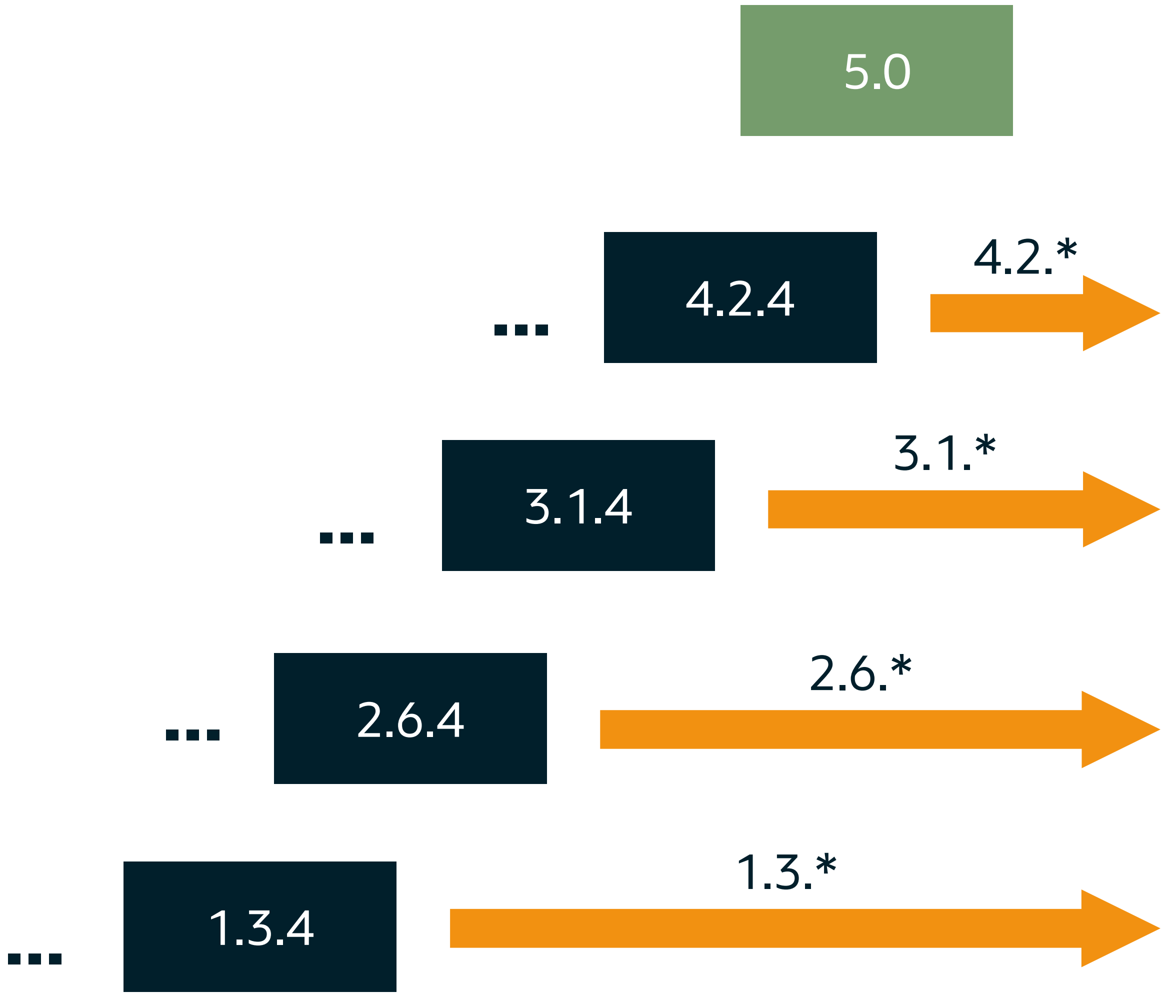
Tip release: Everything

Tail releases: Only critical bug fixes & security patches

Tip & Tail ❤️ Multiple Release Trains



Tip & Tail ❤️ Multiple Release Trains



Functionality

Upgrade whenever higher productivity or better performance is available

Predictability

Upgrade when benefits outweigh costs

Stability

Upgrade as little as possible

Tip & Tail:

Gives you what you need

Doesn't give you what you don't need

Tip & Tail
for Library Developers

Library developers:

Add new features only in the tip, not in the tails

Backport as little as possible from tip to tails

Backport as little as possible

Fixes for critical bugs
Patches for security vulnerabilities
Changes to externally-sourced data sets

Reduces the churn in tail trains.
✓ *Good for users who value stability*, since updates are lower risk.

Increases the number of tail trains, since the cost of maintaining each tail is low.
✓ *Good for users who value predictability*, since they can move forward at their own pace.

Reduces the time invested in tail trains so that more time is available to work on the tip train.
✓ *Good for users who value functionality*, who get more new features.

Tip & Tail leaves plenty to the library developer

Does not specify when or why tail trains are created, nor when or why they are discontinued.

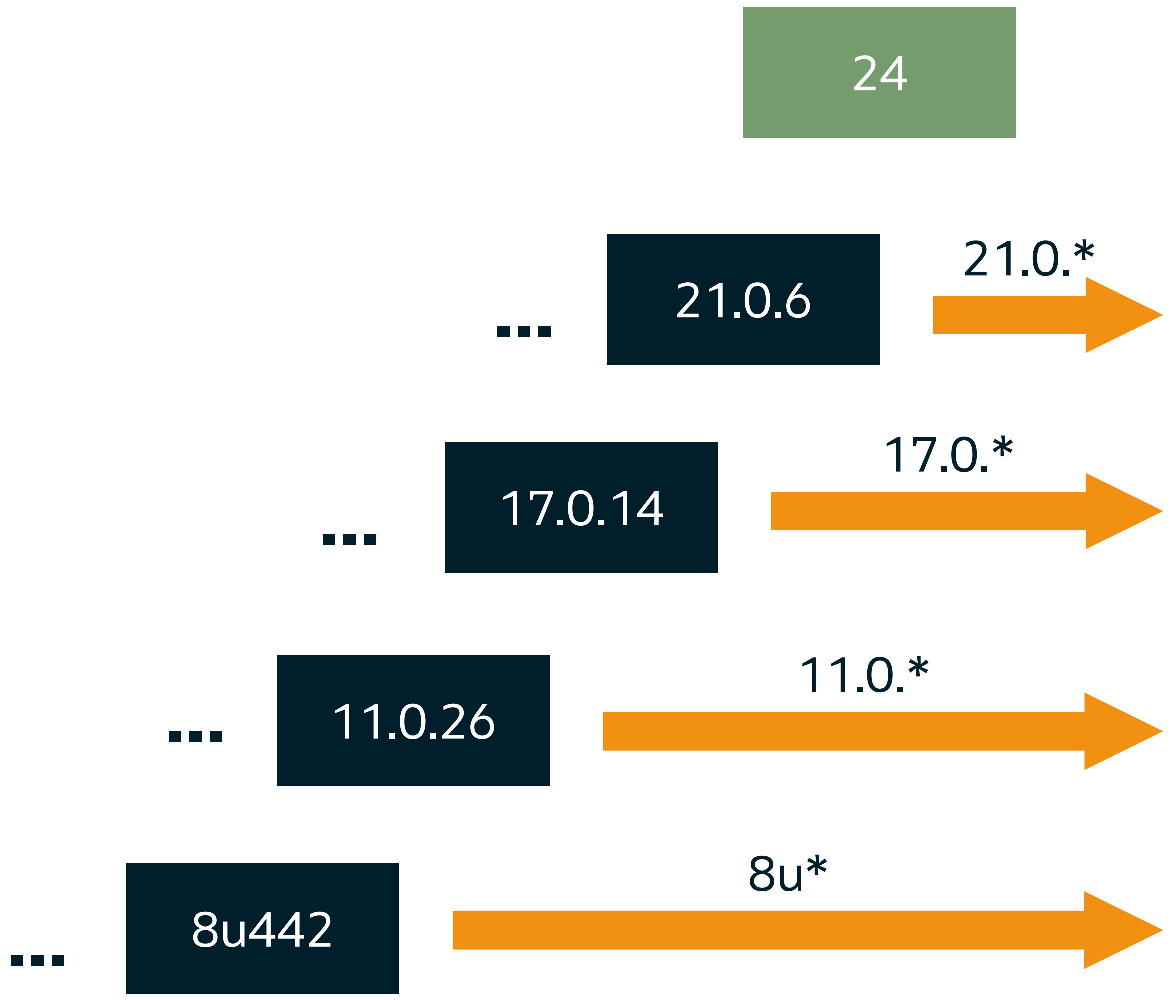
Does not specify how releases are versioned, or licensed.

Does not require the tip release to be baselined on the latest JDK.

- e.g., The tip train could target JDK 21 while tail trains target JDK 8 and 17
- e.g., Every train could target JDK 17

Tip & Tail in the JDK

JDK ❤ Tip & Tail



Functionality

Upgrade whenever higher productivity or better performance is available

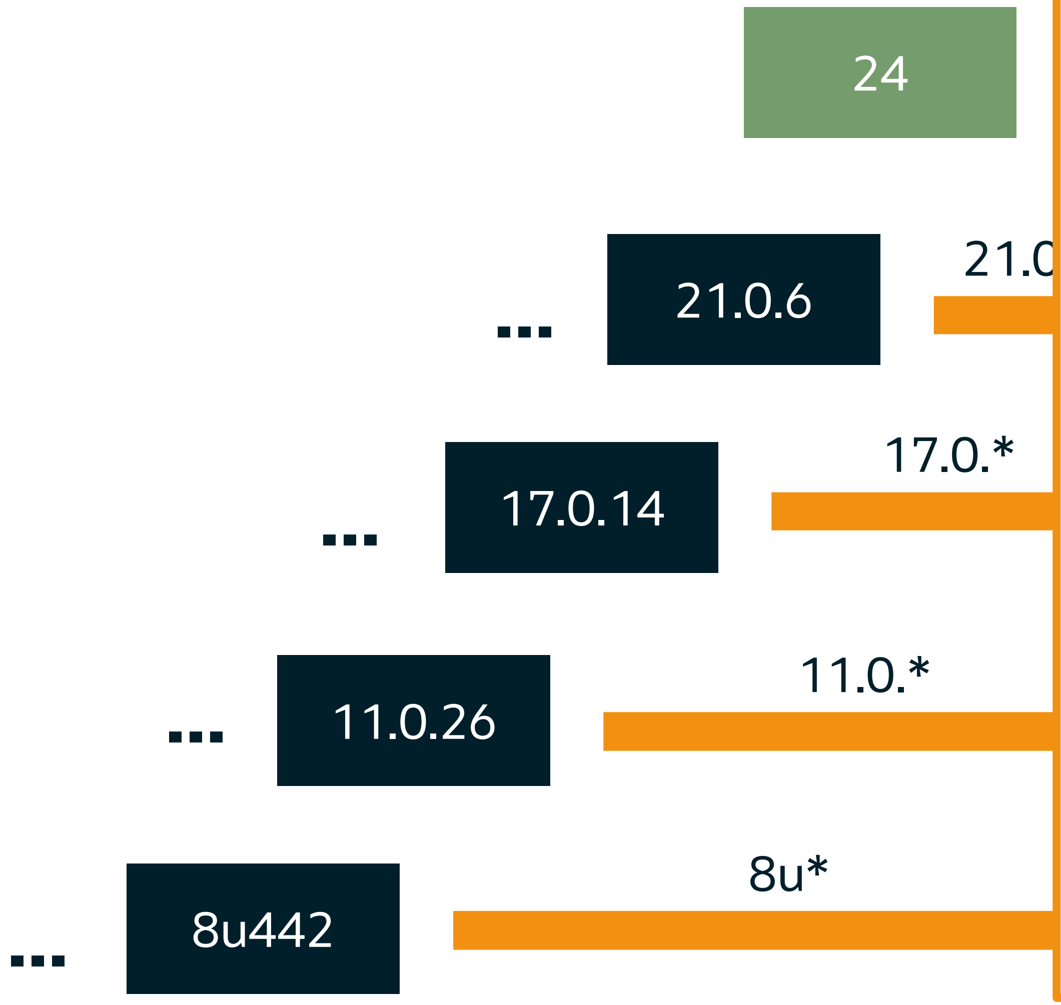
Predictability

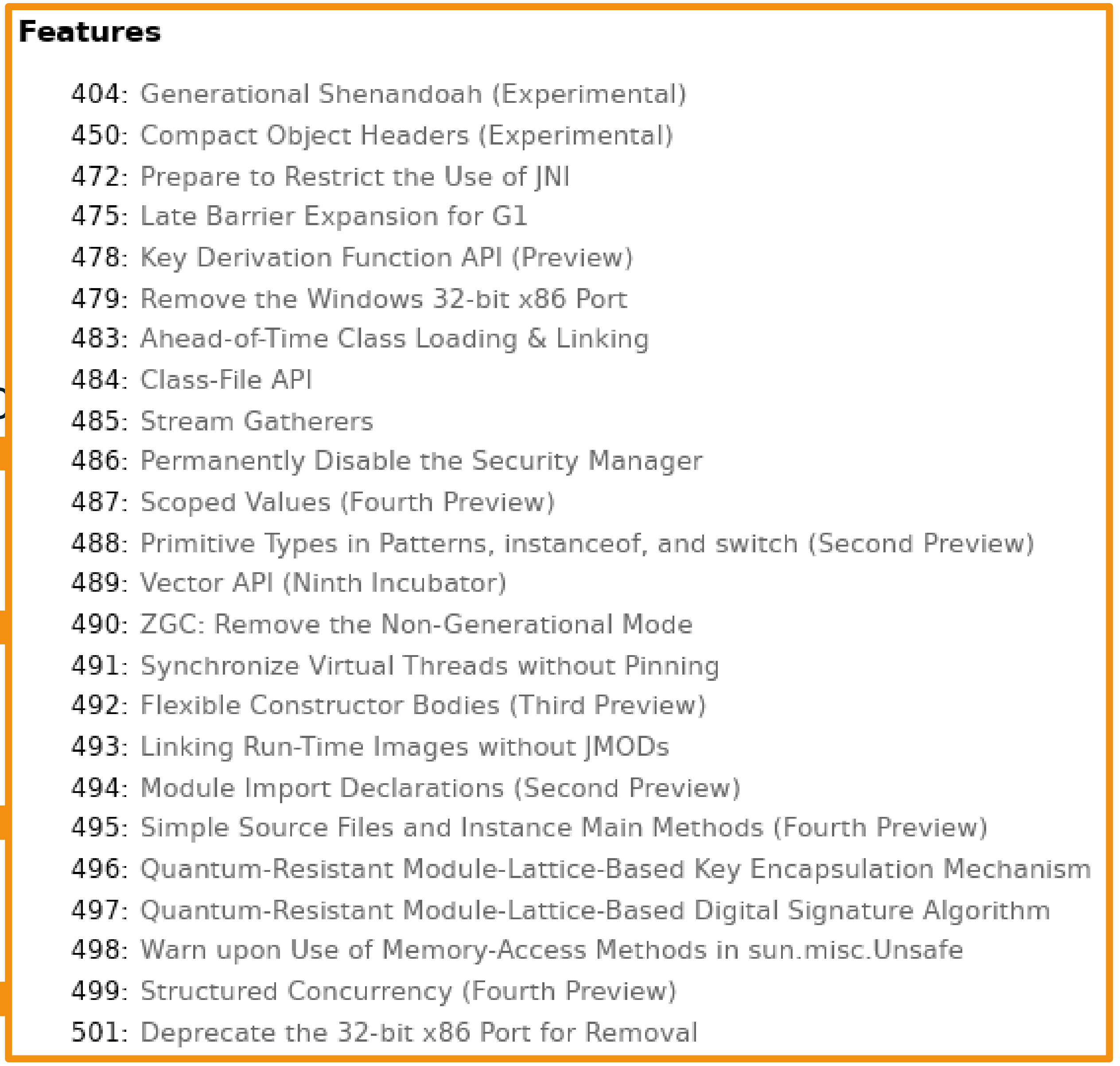
Upgrade when benefits outweigh costs

Stability

Upgrade as little as possible

JDK ❤ Tip & Tail

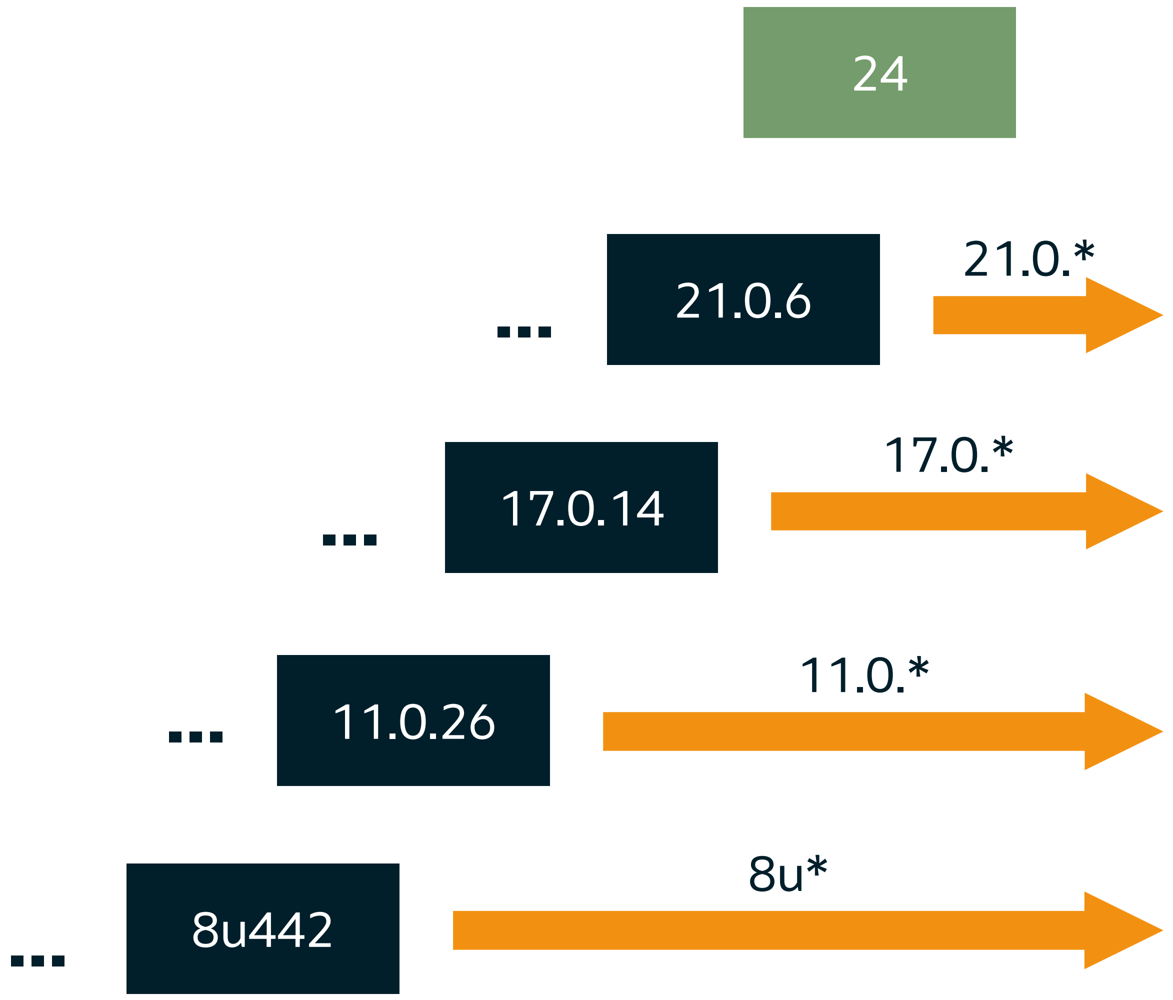




Features

404: Generational Shenandoah (Experimental)
450: Compact Object Headers (Experimental)
472: Prepare to Restrict the Use of JNI
475: Late Barrier Expansion for G1
478: Key Derivation Function API (Preview)
479: Remove the Windows 32-bit x86 Port
483: Ahead-of-Time Class Loading & Linking
484: Class-File API
485: Stream Gatherers
486: Permanently Disable the Security Manager
487: Scoped Values (Fourth Preview)
488: Primitive Types in Patterns, instanceof, and switch (Second Preview)
489: Vector API (Ninth Incubator)
490: ZGC: Remove the Non-Generational Mode
491: Synchronize Virtual Threads without Pinning
492: Flexible Constructor Bodies (Third Preview)
493: Linking Run-Time Images without JMODs
494: Module Import Declarations (Second Preview)
495: Simple Source Files and Instance Main Methods (Fourth Preview)
496: Quantum-Resistant Module-Lattice-Based Key Encapsulation Mechanism
497: Quantum-Resistant Module-Lattice-Based Digital Signature Algorithm
498: Warn upon Use of Memory-Access Methods in sun.misc.Unsafe
499: Structured Concurrency (Fourth Preview)
501: Deprecate the 32-bit x86 Port for Removal

JDK ❤ Tip & Tail



Functionality

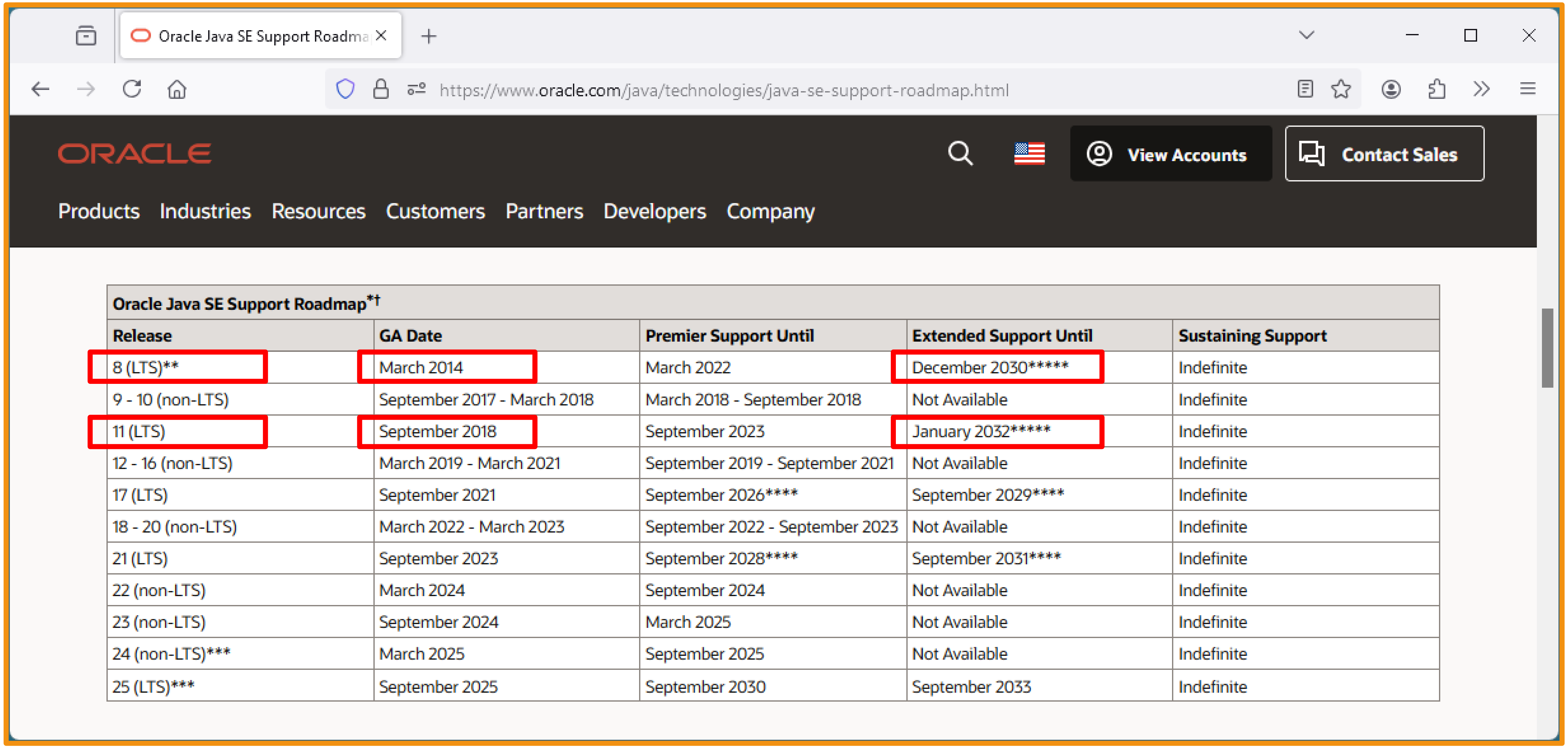
Upgrade whenever higher productivity or better performance is available

Predictability

Upgrade when benefits outweigh costs

Stability

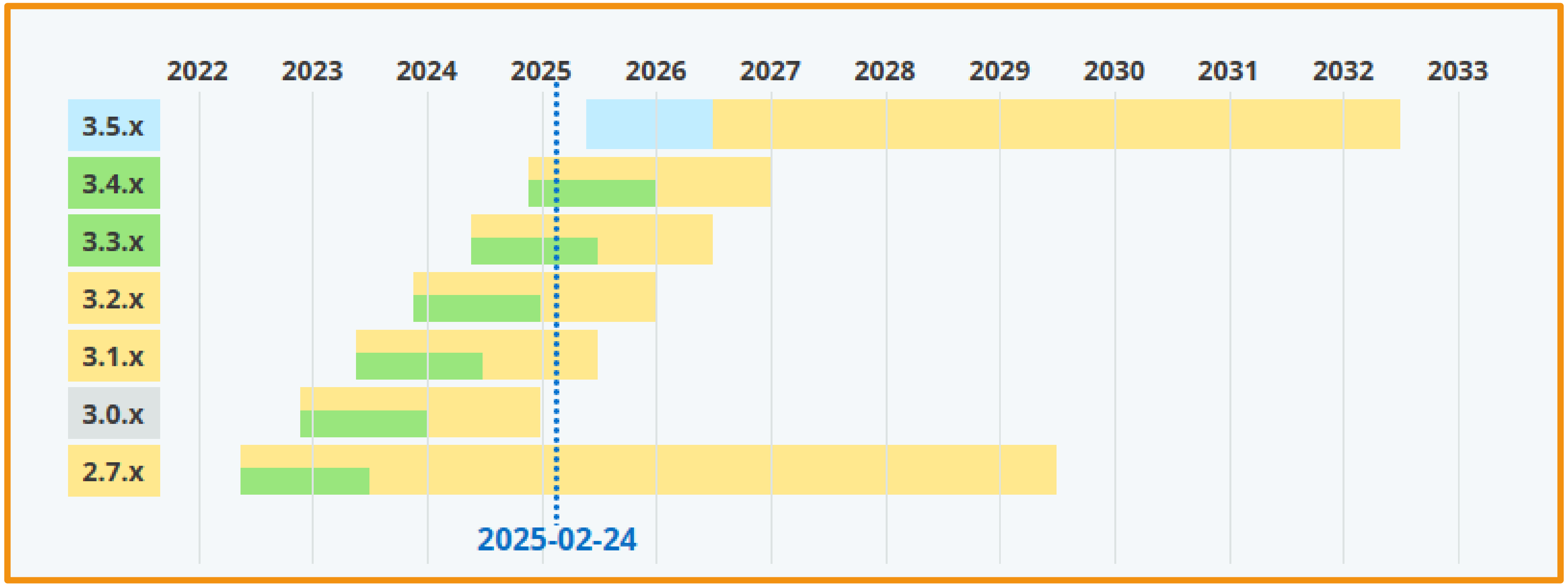
Upgrade as little as possible


Oracle Java SE Support Roadma
https://www.oracle.com/java/technologies/java-se-support-roadmap.html
ORACLE
View Accounts
Contact Sales
Products
Industries
Resources
Customers
Partners
Developers
Company

Oracle Java SE Support Roadmap*†

Release	GA Date	Premier Support Until	Extended Support Until	Sustaining Support
8 (LTS)**	March 2014	March 2022	December 2030*****	Indefinite
9 - 10 (non-LTS)	September 2017 - March 2018	March 2018 - September 2018	Not Available	Indefinite
11 (LTS)	September 2018	September 2023	January 2032*****	Indefinite
12 - 16 (non-LTS)	March 2019 - March 2021	September 2019 - September 2021	Not Available	Indefinite
17 (LTS)	September 2021	September 2026****	September 2029****	Indefinite
18 - 20 (non-LTS)	March 2022 - March 2023	September 2022 - September 2023	Not Available	Indefinite
21 (LTS)	September 2023	September 2028****	September 2031****	Indefinite
22 (non-LTS)	March 2024	September 2024	Not Available	Indefinite
23 (non-LTS)	September 2024	March 2025	Not Available	Indefinite
24 (non-LTS)***	March 2025	September 2025	Not Available	Indefinite
25 (LTS)***	September 2025	September 2030	September 2033	Indefinite

Tip & Tail in Spring Boot



https://spring.io/projects/spring-boot#support

In Closing

Tip & Tail in a Nutshell

Tip & Tail is a streamlined and disciplined form of the multi-train release model for libraries.

It gives users exactly what they need, whether their focus is functionality, predictability, or stability.

The JDK adopted Tip & Tail to provide a balance between rapid innovation and long-term support.

As more libraries adopt Tip & Tail, the Java ecosystem will become even more attractive for new applications, and even more reliable for existing applications.

Thank you