ORACLE

Java SE Update

Georges Saab

Senior Vice President, Oracle Java Platform Group

Chair, OpenJDK Governing Board

April 12, 2023

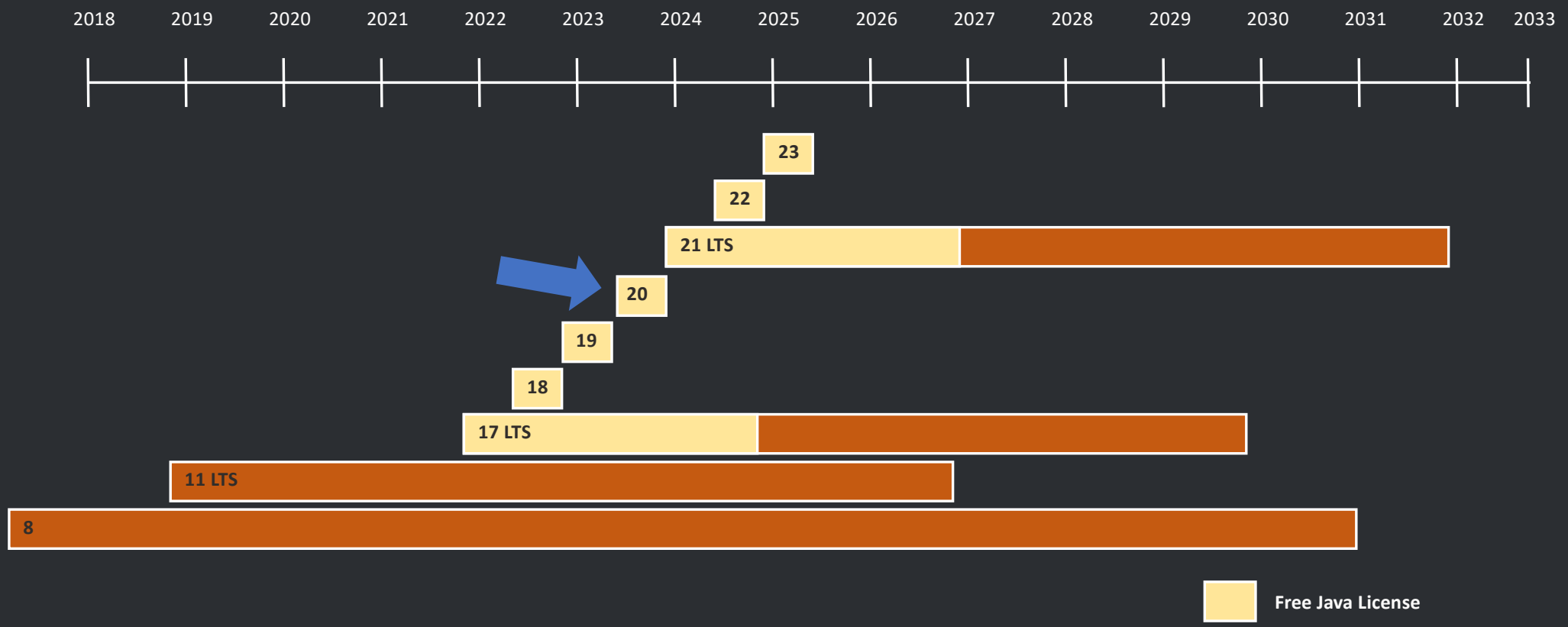
Retrospective: New Release Cadence

History

- Started in 2017/18 with JDK 10
- Have now delivered 11 of these (5 years)
- JCP EC was involved in making this possible
- Market view has moved from skepticism to enthusiasm

- What can we learn, what is next? If possible, be provocative!

Oracle JDK Releases



Copyright © 2023, Oracle and/or its affiliates. All rights reserved.
4

Why did we do these again?

- Developer perception of Java as having a “glacial pace of development”
  - This was largely because of latency rather than throughput
  - Developers wished they could get access to new features more quickly
- Enterprises said they preferred stability to new features – but sometimes would choose differently when it came to new projects…
  - Indicating that they placed a higher value on innovation than confessed
  - LTS offering was an important step
- We wanted better predictability
  - Internal planning and replanning cycles were very expensive
  - The track record of ‘delivering on time’ was not good when the priority of ‘ensuring we didn’t make mistakes the world would have to live with for decades’ was (and should be!) a higher priority

- There was also a business reason – that changes driving revenue once every few years tend to be bad in a global industry with “faster cycles”

A *lot* of work was needed

- Much of it was done before even proposing the cadence change
- Much of it was not ‘visible’
  - like improvements to infrastructure and testing to ensure quality of features on branches prior to integration in mainline]
  - Some rode on work being done for other reasons as well, such as the Java Module system, and Skara

- Very much of it was cultural rather than technical
  - Like helping people grok that it was preferable to take the next train if you weren’t going to make this one smoothly
  - Like moving to “tip development with LTS” rather than “express model” or “backporting all the things” (more on this in a minute!!!)
  - Sometimes unlearning decades of experience and muscle memory

Tip Development vs Express model

- JVM development in 2010 was largely done based on the express model
  - Separate development of JVM and “rest of JDK”
  - Latest JVM packaged with “latest version of each JDK under support”
  - This was done by, for instance, JRockit, J9, and yes, Hotspot
- Advantages
  - New cool shiny JVM for everyone all the time!
- Drawbacks
  - Held back JVM development
  - Increased complexity and slowed down overall progress
  - Made it super hard/slow to deliver features that spanned both areas
  - Tended to encourage a desire to “backport all the things!” which
    - Increased complexity and cost of backporting
    - Contributed to even slower pace of adoption of new versions

Tip Development vs Express model

- Key insights
  - Customers never asked for express model – in fact it meant you were delivering a new and potentially risky VM to people who had intended to choose stability
  - Effectively it was a developer conceit driven by the bad aspects of the multi-year, big bang release cadence

- So
  - If we moved from express to tip+LTS, we could move Java as a whole forward far faster
  - We should exercise restraint in doing backports
    - Security of course, stability and perf sometimes
  - For new features, we should make it easier to move forward and expect people really needing these to be willing to do so

Encouraging people to use modern versions of Java extends the viability of the technology and its ecosystem in the face of healthy competition

Initial reception was skeptical

- From everyone!
- Some wanted to view these as Betas
- Many were sure that we would lose interest after a few of these
- No one knew whether these would actually bring about better predictability
- Or how throughput of feature delivery would be affected

- Quotes at the time included “no one will ever need anything past JDK 8!!
- And no one was more skeptical than “people on the inside”!

Ian Brown
@igb

After careful deliberation, having consulted with many JVM experts, I can confidently say that the new Java release cadence is bullshit.

12:05 AM · Dec 8, 2017 from San Francisco, CA

10 Likes

Nitsan "Yak" Wakart
@nitsanw

Replying to @snazy and @normanmaurer

If a JVM is released every 6 months, but everyone is still using Java 8, did it actually release?

5:10 PM · May 9, 2018

Stephen Colebourne
@jodastephen

I want to highlight this tweet. We now have Amazon, Red Hat, IBM, Spring and more only focussed on Java LTS releases - 8, 11, then (probably) 17. As per my last blog, it looks like the 6 month release cycle will be of limited value to most Java devs: blog.joda.org/2018/10/adopt-...

> **Arun Gupta** @arungupta · Nov 14, 2018
> Replying to @jodastephen and @errcraft
> At this time, we're planning Corretto 8 and 11 only.

9:21 AM · Nov 15, 2018

Mario Fusco @mariofusco@jvm.social
@mariofusco

Java's 6 months release cadence is

1. Very good, is exciting to see Java moving fast

2. Good in theory, useless in practice. Pro users are on Java 11 at best, all libraries has to support Java 8 at least

3. Harmful for enterprises

4. Just a desperate attempt to monetize Java

1. Very good	30.8%
2. Useless in practice	**49.3%**
3. Harmful	6.3%
4. Just to make money	13.7%

2,315 votes · Final results

10:02 AM · Jan 19, 2020

Gradual acceptance over time

As time went by, the mood started to shift…

- 11 LTS saw relatively modest uptake
- I was quoted as saying “JDK 11 is the new JDK 7”

- We kept plugging away, delivering these one after another
- We found that the infrastructure changes made things smooth
- We found that the regularity and time-boundedness of the releases vastly reduced the complexity of our planning cycles
- We found that the cultural shift of “not running down the icy stairs for the train” vasty reduced our stabilization cycles
- We found that the smaller increments allowed broader ‘introduction testing’ via incubator and preview
- The new cadence helped fuel shifts in the business model which resulted in more customers than ever before
- More vendors saw an opportunity to produce their own downstream distros and support offerings

- Developer confidence in the predictability and quality of 6 month releases grew
- Developers started liking having “always something new at the Java corral”

TechTarget | TheServerSide
Your Enterprise Java Community

Videos Definitions Webinars More

Core Java DevOps Development Frameworks

FEATURE

No doubt now about Java release cycle

At the Oracle Code One keynote, the recent track record of on-time releases and feature enhancements is a topic to boast about.

By **Cameron McKenzie,** TechTarget Published: **17 Sep 2019**

SAN FRANCISCO -- Given a five-year wait for the release of Java 11, and three-year spans between the releases of Java 8 and Java 9, it's easy to understand why the Java community raised a collective brow when Oracle announced a switch to a six-month Java release cycle.

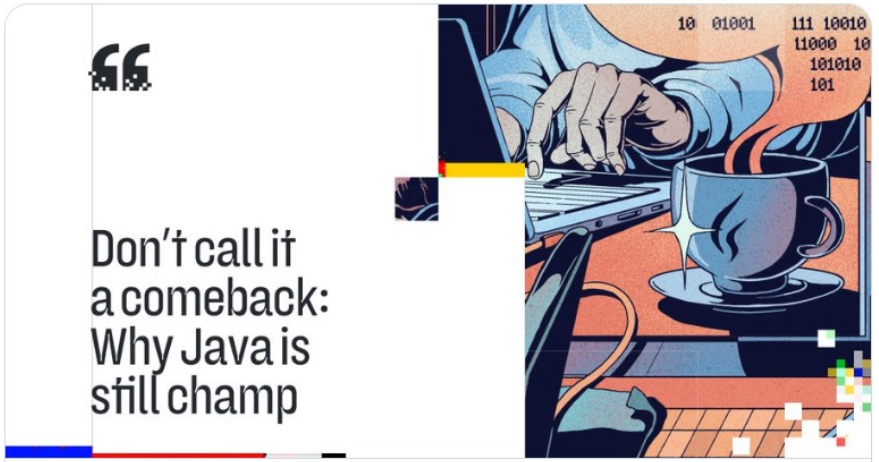
Gunnar Morling
@gunnarmorling

Looking at the planned release contents of #Java 19 (e.g. preview of virtual threads + foreign function/memory API), I think the 6 months release cadence is the best thing that ever has happened to Java. The constant influx of fundamental improvements is nothing but impressive.

5:51 PM · May 24, 2022

Dan Vega
@therealdanvega

"Java really hit its stride with Java 9 and the increased release cadence around 2018," he says. "The smaller but much more frequent releases are really working well." #java



github.com
Don't call it a comeback: Why Java is still champ
Java has been declared dead many times—and yet, it's still going strong, undergoing what some might call a renaissance from both a technology and ...

9:41 PM · Aug 9, 2022

Current view

When Java 20 came out last month, I had a quick look around social media etc to gauge customer temperature (as usual).

By new the mood has completely flipped from skepticism to enthusiasm:
The six month releases are the best thing that ever happened to Java
Java is firing on all cylinders
New Java renaissance, Java not perceived as having a ‘glacial pace’.
In many respects ahead of others, where not, catching up and overtaking quickly
Etc Etc.

JPG and OpenJDK developers love it and can’t imagine going back to how things were done before [Our sales/finance/executives love it too!]
Other vendors/distros love it too
People who develop in Java love it – huge sense of optimism in the roadmap
Enterprises simply choose their vendor, use the LTS, and plan their upgrades when convenient. They also are choosing Java more than ever for new projects

So….. What’s next?

Amazing-Cicada5536 · vor 21 Tagen

The new release model is probably the best thing that happened to Java on the management side — instead of the usual churn of "this feature has to be done till this deadline", which might work for that boring CRUD feature but is absolutely terrible for such a complex program like the JVM, it allows for each new development to take as mucl time as necessary. If it didn't get ready for release N, just continue to work on it and 6 months later it can be delivered. It is especially important as many of these features have non-trivial interactions with each others.

StoneOfTriumph · vor 21 Tagen

What some people forget is that the JDK is advancing at a faster pace with smaller deliverables more frequency than before, so while it seems that they released a buttload of versions, the releases themselves include a smaller delta, so the jump to 8 to 17 isn't that bad in terms of breaking changes, not near as difficult to perform as 6 to 8.

When you take the time to actually look at the changelog, when you take the time to actually upgrade your project's java version source/targets to a higher version as part of a proof of concept/spike/timebox, there's much less than you may think.

Joram2 · vor 21 Tagen

Java 20 is basically a patch release on Java 19. But it's important + necessary.

TheCountRushmore · vor 21 Tagen

This contains lots of other bug fixes and improvements. It's a full production quality release.

henk53 · vor 21 Tagen

Why don't we backport *everything* to the supported LTS releases? Then those supported LTS releases will be virtually identical to Java 20 each ;)

1 Antworten Teilen •••

JakeWharton · vor 21 Tagen

That's just called updating to the latest. The best LTS release remains and will remain simply tracking the latest version.

MyFavouriteNick · vor 20 Tagen

For all the in-house projects that I maintain the upgrade from OpenJDK 17 to 19 (and now 20) has been a simple version bump too.

In my opinion, 12/13/14/15/16 can and should be used in production. But you don't use just 12 or 16 and leave it there, you always depend on the latest release, and plan accordingly. 6 months of support is absolutely fine if an adequately set up CI is in place and you have good end-to-end test coverage.

CubsThisYear · vor 20 Tagen

Meh - wake me up when we get to Valhalla.

THENEWSTACK

PODCASTS EBOOKS EVENTS NEWSLETTER

ARCHITECTURE ENGINEERING OPERATIONS

LTS Release Cadence

Meanwhile, Saab said Oracle has proposed that the next LTS release should be Java 21 and made available in September 2023, which will change the ongoing LTS release cadence from three years to two years.

"I think this is a pretty big step for Oracle and the Java community at large because in moving LTS from three to two years, they're saving enterprises from having to delay their adoption of new features," said Brad Shimmin, an analyst at Omdia. "This move in conjunction with their decision to include an offer to run Oracle JDK for free without support with one year as an overlap with the next release, lets users explore new features and build those into their long-term deployment plans with greater freedom and frequency."

Where should we go from here

We've gained a lot with the shift in delivery model for Java.
If there is one thing I hear people cite as holding them back from even more eager use of new versions at this point, it is "3rd party library support".

And what I hear from the authors of third party libraries is "it is too difficult/expensive to make our newest version support all these different Java versions!!

- This is exactly the "tip vs express" cultural change which I described before.

Ok, I promised to be provocative. Here goes!!

> "The world is ready for the ecosystem of Java libraries, frameworks, and tools to embrace a delivery model similar to that of the JDK. Tip development, with LTS offerings. By making this shift, library vendors can realize exactly the same kind of benefits that we have achieved for the Java platform itself. What's more, this will further strength and extend the viability of Java overall."

Discuss!!