

ORACLE

# How to Contribute to OpenJDK Projects

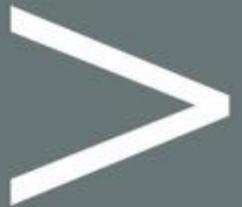
---

**Alex Buckley**

Java Platform Group

Oracle

June / August 2025



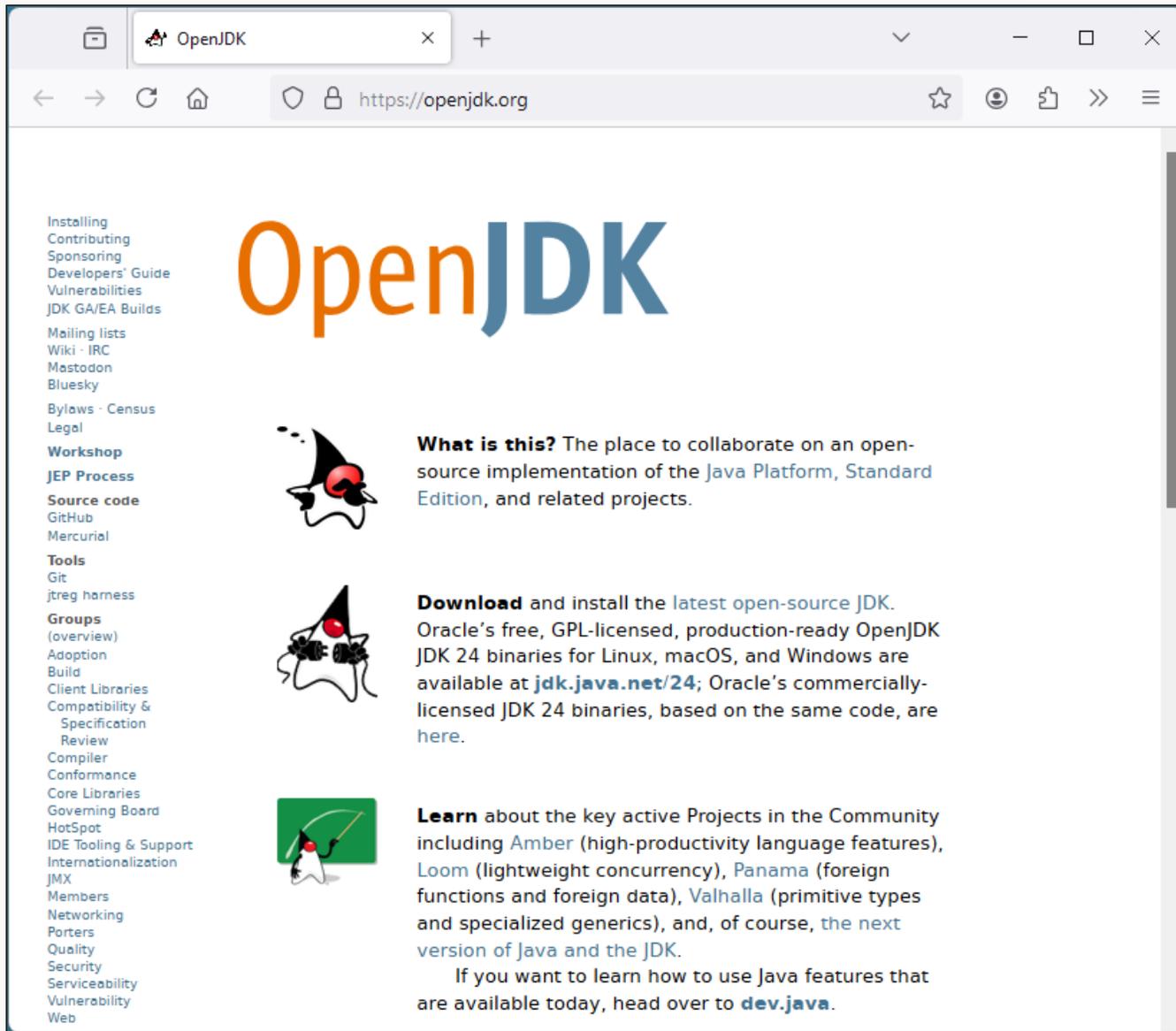
# How to Contribute to OpenJDK Projects

1. Structure of OpenJDK
2. Contributing to OpenJDK
3. Practical steps for contributing

# 1. Structure of OpenJDK

---





Founded 2006

An association of developers

Produces an open-source implementation of the Java SE Platform

Source code only, no binaries

Has a Governing Board (Roughly similar to the EC)

"A place, not a thing"  
Work happens *in* OpenJDK  
*The* OpenJDK 21 does not exist



Organizing principle:

Groups (20) + Projects (54)

Groups sponsor Projects

Projects have roles:

Author

Committer

Reviewer

Project Lead

Projects have repositories

Most actions require public votes

**Groups**  
(overview)  
Adoption  
Build  
Client Libraries  
Compatibility &  
Specification  
Review  
Compiler  
Conformance  
Core Libraries  
Governing Board  
HotSpot  
IDE Tooling & Support  
Internationalization  
JMX  
Members  
Networking  
Porters  
Quality  
Security  
Serviceability  
Vulnerability  
Web

**Projects**  
(overview, archive)  
Amber  
Babylon  
CRaC  
Code Tools  
Coin  
Common VM  
Interface  
Developers' Guide  
Device I/O  
Duke  
Galahad  
Graal  
IcedTea  
JDK 8 Updates  
JDK 9  
JDK (..., 23, 24, 25)  
JDK Updates  
JMC  
Jigsaw  
Kona  
Kulla  
Lanai  
Leyden  
Lilliput  
Locale Enhancement  
Loom  
Memory Model  
Update  
Metropolis  
Multi-Language VM  
Nashorn  
New I/O  
OpenJFX  
Panama  
Penrose  
Port: AArch32  
Port: AArch64  
Port: BSD  
Port: Haiku  
Port: Mac OS X  
Port: MIPS  
Port: Mobile  
Port: PowerPC/AIX  
Port: RISC-V  
Port: s390x  
SCTP  
Shenandoah  
Skara  
Sumatra  
Tson  
Valhalla  
Verona  
VisualVM  
Wakefield  
Zero  
ZGC

Organizing principle:

Groups (20) + Projects (54)

Groups sponsor Projects

Projects have roles:

Author

Committer

Reviewer

Project Lead

Projects have repositories

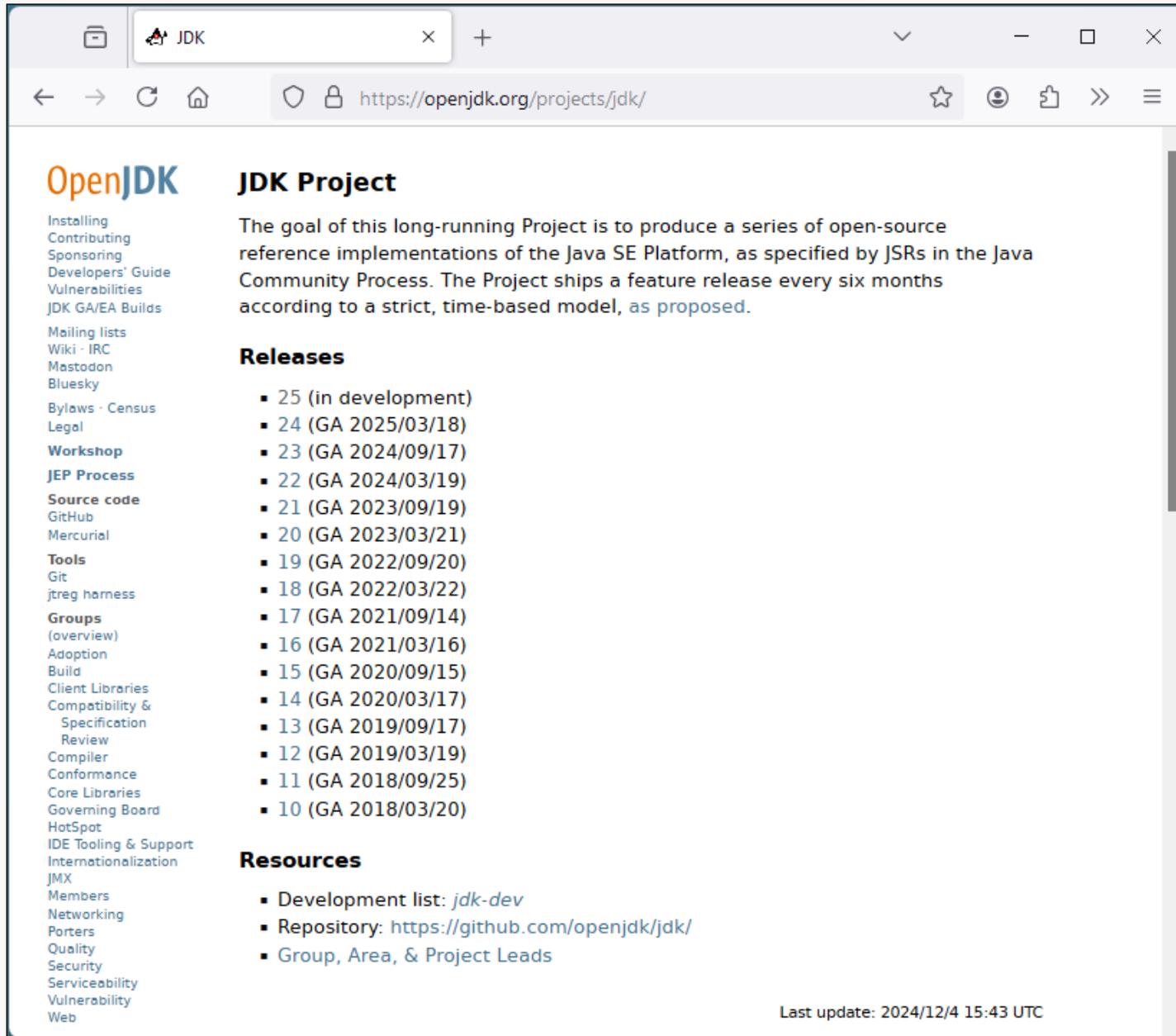
Most actions require public votes

**Groups**  
(overview)  
Adoption  
Build  
Client Libraries  
Compatibility &  
Specification  
Review  
Compiler  
Conformance  
Core Libraries  
Governing Board  
HotSpot  
IDE Tooling & Support  
Internationalization  
JMX  
Members  
Networking  
Porters  
Quality  
Security  
Serviceability  
Vulnerability  
Web



**Projects**  
(overview, archive)  
Amber  
Babylon  
CRaC  
Code Tools  
Coin  
Common VM  
Interface  
Developers' Guide  
Device I/O  
Duke  
Galahad  
Graal  
IcedTea  
JDK 8 Updates  
JDK 9  
JDK (..., 23, 24, 25)  
JDK Updates  
JMC  
Jigsaw  
Kona  
Kulla  
Lanai  
Leyden  
Lilliput  
Locale Enhancement  
Loom  
Memory Model  
Update  
Metropolis  
Multi-Language VM  
Nashorn  
New I/O  
OpenJFX  
Panama  
Penrose  
Port: AArch32  
Port: AArch64  
Port: BSD  
Port: Haiku  
Port: Mac OS X  
Port: MIPS  
Port: Mobile  
Port: PowerPC/AIX  
Port: RISC-V  
Port: s390x  
SCTP  
Shenandoah  
Skara  
Sumatra  
Tson  
Valhalla  
Verona  
VisualVM  
Wakefield  
Zero  
ZGC



A screenshot of a web browser displaying the OpenJDK Project website. The browser's address bar shows the URL https://openjdk.org/projects/jdk/. The page features a navigation sidebar on the left with links for installing, contributing, and other project-related topics. The main content area includes the OpenJDK logo, a title 'JDK Project', a paragraph describing the project's goal, a list of releases from version 10 to 25, and a list of resources. The footer indicates the last update was on 2024/12/4 at 15:43 UTC.

**OpenJDK**

**JDK Project**

The goal of this long-running Project is to produce a series of open-source reference implementations of the Java SE Platform, as specified by JSRs in the Java Community Process. The Project ships a feature release every six months according to a strict, time-based model, as proposed.

**Releases**

- 25 (in development)
- 24 (GA 2025/03/18)
- 23 (GA 2024/09/17)
- 22 (GA 2024/03/19)
- 21 (GA 2023/09/19)
- 20 (GA 2023/03/21)
- 19 (GA 2022/09/20)
- 18 (GA 2022/03/22)
- 17 (GA 2021/09/14)
- 16 (GA 2021/03/16)
- 15 (GA 2020/09/15)
- 14 (GA 2020/03/17)
- 13 (GA 2019/09/17)
- 12 (GA 2019/03/19)
- 11 (GA 2018/09/25)
- 10 (GA 2018/03/20)

**Resources**

- Development list: [jdk-dev](#)
- Repository: <https://github.com/openjdk/jdk/>
- [Group, Area, & Project Leads](#)

Last update: 2024/12/4 15:43 UTC

The screenshot shows the OpenJDK 25 project page. A red circle highlights the 'Schedule' table, which lists key milestones for the release. The table includes dates and descriptions for Rampdown Phase One, Rampdown Phase Two, Initial Release Candidate, Final Release Candidate, and General Availability.

Date	Event
2025/06/05	Rampdown Phase One (branch from main line)
2025/07/17	Rampdown Phase Two
2025/08/07	Initial Release Candidate
2025/08/21	Final Release Candidate
2025/09/16	General Availability



OpenJDK **JDK 25**

This release will be the Reference Implementation Platform, as specified by JSR 400 in the Java Code

**Schedule**

2025/06/05	Rampdown Phase One (bra
2025/07/17	Rampdown Phase Two
2025/08/07	Initial Release Candidate
2025/08/21	Final Release Candidate
2025/09/16	General Availability

**Status**

The main line branch is open for bug fixes, sma proposed and tracked via the JEP Process.

**Features**

**JEPs proposed to target JDK 25**

- 509: JFR CPU-Time Profiling (Experimental)

**JEPs targeted to JDK 25, so far**

- 470: PEM Encodings of Cryptographic Objects
- 502: Stable Values (Preview)
- 503: Remove the 32-bit x86 Port
- 505: Structured Concurrency (Fifth Preview)
- 506: Scoped Values
- 507: Primitive Types in Patterns, instanceof, and switch (Third Preview)
- 508: Vector API (Tenth Incubator)
- 510: Key Derivation Function API
- 511: Module Import Declarations
- 512: Compact Source Files and Instance Main Methods
- 513: Flexible Constructor Bodies

OpenJDK **JEP 3: JDK Release Process**

**Owner** Mark Reinhold  
**Type** Process  
**Scope** JDK  
**Status** Active  
**Discussion** jdk dash dev at openjdk dot java dot net  
**Created** 2018/06/19 16:58  
**Updated** 2024/12/02 21:41  
**Issue** 8205352

**Summary**

Define the process by which Contributors in the OpenJDK Community produce time-based, rapid-cadence JDK feature releases.

**Quick reference**

This table is provided here for easy access; the terminology it uses is defined below.

	Candidates	Fix	Drop	Defer	Enhance?
<b>RDP 1</b>	Current P1-P3 Targeted P1-P3	Current P1-P3 Targeted P1-P3 if time P1-P5 doc/test changes	All P4-P5 Targeted P1-P3	Current P1-P2, with approval	With approval
<b>RDP 2</b>	Current P1-P2 Targeted P1-P2	Current P1-P2, with approval P1-P5 doc/test changes	All P3-P5 Targeted P1-P2	Current P1-P2, with approval	With approval
<b>RC</b>	Current P1 Targeted P1	Current P1, with approval	All P2-P5 Targeted P1	Current P1, with approval	No



OpenJDK **JDK 25**

This release will be the Reference Implementation of version 25 of the Java SE Platform, as specified by JSR 400 in the Java Community Process.

**Schedule**

2025/06/05	Rampdown Phase One (branch from main line)
2025/07/17	Rampdown Phase Two
2025/08/07	Initial Release Candidate
2025/08/21	Final Release Candidate
2025/09/16	General Availability

**Status**

The main line branch is open for bug fixes, small enhancements, and JEPs as proposed and tracked via the JEP Process.

**Features**

<b>JEPs proposed to target JDK 25</b>	<i>review ends</i>
509: JFR CPU-Time Profiling (Experimental)	2025/06/04
<b>JEPs targeted to JDK 25, so far</b>	
470: PEM Encodings of Cryptographic Objects (Preview)	
502: Stable Values (Preview)	
503: Remove the 32-bit x86 Port	
505: Structured Concurrency (Fifth Preview)	
506: Scoped Values	
507: Primitive Types in Patterns, instanceof, and switch (Third Preview)	
508: Vector API (Tenth Incubator)	
510: Key Derivation Function API	
511: Module Import Declarations	
512: Compact Source Files and Instance Main Methods	
513: Flexible Constructor Bodies	



**OpenJDK** **JEP 520: JFR Method Timing & Tracing**

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  
jreg harness  
**Groups**  
(overview)  
Adoption  
Build  
Client Libraries  
Compatibility & Specification Review  
Compiler  
Conformance  
Core Libraries  
Governing Board  
HotSpot  
IDE Tooling & Support  
Internationalization  
JMX  
Members  
Networking  
Porters  
Quality  
Security  
Serviceability

<i>Owner</i>	Erik Gahlin
<i>Type</i>	Feature
<i>Scope</i>	JDK
<i>Status</i>	Integrated
<i>Release</i>	25
<i>Component</i>	hotspot/jfr
<i>Discussion</i>	hotspot dash jfr dash dev at openjdk dot org
<i>Effort</i>	S
<i>Duration</i>	S
<i>Reviewed by</i>	Markus Grönlund, Vladimir Kozlov
<i>Endorsed by</i>	Vladimir Kozlov
<i>Created</i>	2024/03/20 14:10
<i>Updated</i>	2025/05/29 17:53
<i>Issue</i>	8328610

**Summary**

Extend the [JDK Flight Recorder \(JFR\)](#) with facilities for method timing and tracing via [bytecode instrumentation](#).

**Goals**

- For method invocations, record complete and exact statistics rather than incomplete and inexact sample-based statistics.
- Allow execution times and stack traces to be recorded for specific methods without requiring source code modifications.
- Allow methods to be selected via command-line arguments, configuration files, the `jcmd` tool, and over the network via the [Java Management Extensions API \(JMX\)](#).

**Future Work**

Sometimes, a class to be timed or traced implements a known interface, but the name of the class itself is unknown. In such cases it would be convenient to be able to specify the interface in a filter, causing every class implementing that interface to be timed or traced. We could add such functionality in the future, without altering this design.

**Alternatives**

- JFR could record method arguments and non-static fields when timing or tracing. That would, however, enable it to be used as an attack vector for exfiltrating sensitive information, either programmatically by an unvetted third-party library or through a maliciously crafted configuration file.
- The filter grammar could allow specific method-name overloads to be specified. That would, however, complicate the syntax since comma-separated parameter names would conflict with the commas that separate JFR options.
- The filter grammar could accept wildcards, but that could lead to excessive numbers of instrumented methods, bringing the application to a halt.
- To prevent excessive numbers of instrumented methods, JFR could limit the number of methods that can be instrumented. There are, however, scenarios where that can be acceptable. For example, static initializers are only invoked once, so asking to time or trace all of them is not unreasonable.

**Risks and Assumptions**

Specifying a filter that includes JDK methods used by the injected instrumentation bytecode can lead to infinite recursion. JFR attempts to avoid this, but its mechanism for doing so is fragile. If you observe such recursion then please submit a bug report; in the mean time, you can avoid the recursion by removing JDK methods from your filter.

## 2. Contributing to OpenJDK

---



# Different kinds of contribution



**Reporting your experience  
is the most valuable contribution  
you can make**

**Contributing to an OpenJDK Project**

Contributing to OpenJDK can take many forms. Writing code and providing patches is just one of them. A big part of developing a feature or a bugfix is testing and code review. Anything you can do to help out in these areas will be recognized as a contribution. Join the [mailing lists](#) to engage in design discussions and reviews, and download the latest EA builds or Project repositories to try out new features and give feedback. If you see some misbehavior, or if you see somebody mention some misbehavior on some internet forum, try to track it down. Good bug reports with reproducible test cases are extremely valuable and make excellent contributions.

Anything you can do to spread the word about Java, new features, and your experiences using the JDK will be helpful for the community and to the OpenJDK developers. Trying out a new feature and reporting your experiences is also a contribution. Whether you find that the new feature improves your application, or if you find some area that needs to be improved, your feedback is valuable to the developers of that feature.

If you have a success story where Java solved your problem, or if you successfully upgraded to a more recent version of the JDK and noticed some improvements, spreading this story through a blog, news article, or some other channel is also a contribution.

If you're in a position to choose what programming language to use in a project, in a tutorial, or in a class, you have the power to enlarge the Java community in a very direct way, and your colleagues or students will get an opportunity to learn one of the most used programming languages in the world.

# What experiences should you report?

Bug report

Constructive critiques about usability

Test case

Build problem

Performance/benchmarking

Ease/difficulty of migrating your code

Error/typo in javadoc

Discrepancy between JEP and JDK

Suggestions for new examples

Impact on your OSS library

Announcing a blog entry



Is there any way I can help cont: X

https://old.reddit.com/r/java/comments/1kbzu6v/is\_there\_any\_way\_i\_can\_help\_contribute\_to\_valha

### Is there any way I can help contribute to Valhalla? (self.java)

36 submitted 1 month ago by valorzard

Hello!

Project Valhalla interests me, and I'd love to help it along somehow. Is there any way I can contribute pull requests or something to fix bugs to make it arrive faster?

15 comments share save hide report

all 15 comments  
sorted by: **best**

Want to add to the discussion?  
Post a comment!

[CREATE AN ACCOUNT](#)

**[ - ] pron98** 61 points 1 month ago

For all JDK features and projects, the best way to contribute and make them arrive faster is to try out early access and report on the experience. We have no shortage of experts writing code. We do have shortage in people trying out features "in the field" and providing useful feedback before the feature is released.

Useful feedback is not "this is what I think the feature should be," but "I've tried this feature, as currently implemented, in this program; this is what worked well, this is what didn't."

permalink embed save report reply

**[ - ] ThaJedi** 2 points 1 month ago

What's the best way to report?

permalink embed save parent report reply

**[ - ] pron98** 3 points 29 days ago

On the relevant [mailing list](#). For Valhalla that would be [valhalla-dev](#).

permalink embed save parent report reply

SEEK PROGRAMMING HELP

this post was submitted on 01 May 2025

**36 POINTS** (83% upvoted)

shortlink: <https://redd.it/1kbzu6v>

**reddit premium**

Get an ad-free experience with special benefits, and directly support Reddit.

[Get Reddit Premium](#)

[JOIN](#) 365,897 READERS  
76 USERS HERE NOW

**News, Technical discussions, research papers and assorted things of interest related to the Java programming language**

**NO programming help, NO learning Java related questions, NO installing or downloading Java questions, NO JVM languages - Exclusively Java**

These have separate subreddits - see below.

**PLEASE SEEK HELP WITH JAVA PROGRAMMING IN /R/ JAVAHELP!**



File Edit View Go Message Events and Tasks Tools Help

Inbox - Exchange Using CDS and AOT with the Eclipse X

From Daniel Schmid <daniel@wwwmaster.at> 

To leyden-dev@openjdk.org 

Subject Using CDS and AOT with the Eclipse IDE

List-ID Technical discussion related to Project Leyden <leyden-dev.openjdk.org>

5:47 AM

Reply Reply List Forward Archive Junk Delete More

Hi there,

I made a few experiments with CDS and AOT archives on the Eclipse IDE and wanted to share my results here.

I ran 4 Eclipse installations (my personal main Eclipse installation is in a comment further down) with AppCDS and -XX:AOTCache and I have written down my results here: <https://github.com/eclipse-platform/eclipse.platform/discussions/2060>

I think the Eclipse IDE is an interesting application to test as it is a fairly big codebase with many classes loaded at runtime (so it doesn't make it easy for Leyden's improvements) via OSGi.

I have used the latest EA build of JDK 26 (26-ea+6-582). Should I use a different JDK build to test it?

Essentially my findings (which are relevant to this mailing list) are the following:

- There has been an error when creating the AOT archive but it was still created and it was usable. I want to make sure that the people here are aware of that error. To be honest, I did not expect -XX:AOTCache to work that well with Eclipse.
- While CDS seems to come with a noticeable improvement for Eclipse startup time, -XX:AOTCache seems to have similar startup times as CDS (not faster). This might be because of Eclipse's class loading shenanigans.
- It worked with an agent attached The Lombok tooling for Eclipse attaches an agent into the IDE, I think that's necessary to inject itself into the Eclipse Java Compiler).

→ | (0) Today Pane



Inside Java  
News and views from members of the Java team at Oracle

Newscast | Podcast | JEP Café | Sip of Java dev.java | Newsletter | About | Jobs

Sort by: Date | Author | Tag

**Just Be Lazy**  
Per-Ake Minborg on July 29, 2025 JDK 25 Performance

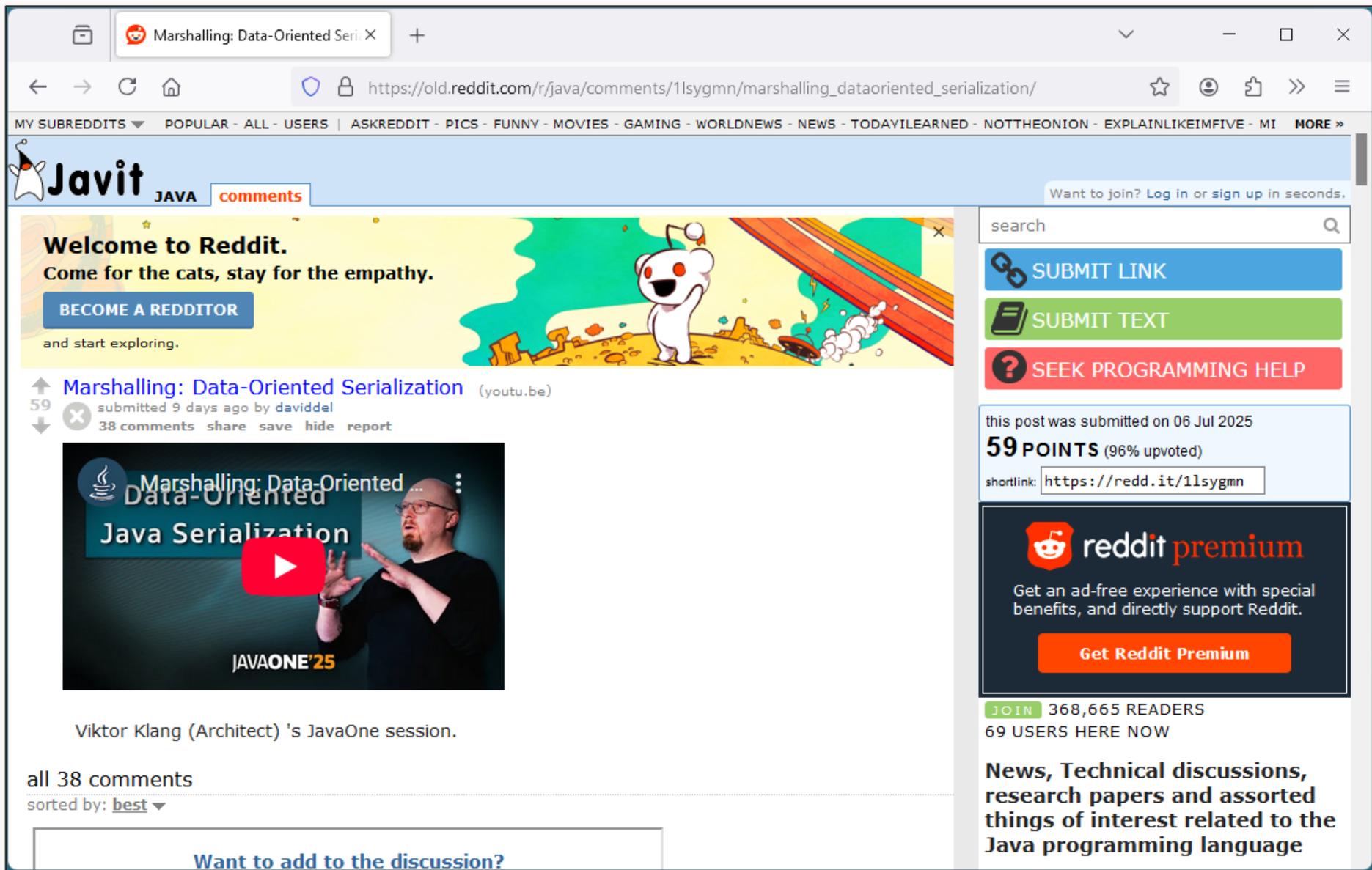
**A New Model for Java Object Initialization** ▶  
Dan Smith on July 27, 2025 Amber Valhalla Java Language

**JEP targeted to JDK 25: 520: JFR Method Timing & Tracing**  
Erik Gahlin on July 25, 2025 JDK 25 JFR

**A Sneak Peek at the Stable Values API** ▶  
Per-Ake Minborg on July 22, 2025 JDK 25 Core Libraries

**JEP targeted to JDK 25: 518: JFR Cooperative Sampling**  
Markus Grönlund on July 21, 2025 JDK 25 JFR

Browser window showing a Reddit post in the r/java subreddit. The post title is "Marshalling: Data-Oriented Serialization" (youtu.be), submitted 9 days ago by daviddel, with 59 points (96% upvoted) and 38 comments. The post content is a video thumbnail for "Marshalling: Data-Oriented Java Serialization" from JAVAONE'25, featuring Viktor Klang (Architect). The right sidebar shows the subreddit description: "News, Technical discussions, research papers and assorted things of interest related to the Java programming language".



Marshalling: Data-Oriented Ser...

https://old.reddit.com/r/java/comments/1lsygm/marshalling\_dataoriented\_serialization/

[ - ] VirtualAgentsAreDumb 2 points 6 days ago

That was a great presentation, thank you. A fellow serialization enthusiast here.

One thing I'm interested in hearing more of is how to handle different versions of classes.

Like, say you have a class that represents a timestamp, ie a specific moment in time. And the internal data is a long, representing the time since the epoch in seconds, ie "unix time". The marshal and unmarshal methods are super easy, as both handle a single long value.

But what if you later want to upgrade your class so it handles millisecond precision? And instead of adding a separate field for the millisecond part, you simply want to change the internal long so that it now represents number of milliseconds since the epoch instead of seconds. How would you handle the case where you get a serialised object of the old version? Both are represented by a single long value, so how would you differentiate between the two?

Will this new serialisation support versioning built in? Or will class authors have to handle that themselves? Like treating the version number as just another field that will be marshalled and unmarshalled?

permalink embed save report reply

[ - ] viktorklang 2 points 5 days ago

That was a great presentation, thank you. A fellow serialization enthusiast here.

Thank you!

How would you handle the case where you get a serialised object of the old version? Both are represented by a single long value, so how would you differentiate between the two?

```
{ "timestamp" = 573857303 } <-- is this seconds or milliseconds?
```

The answer is, that without any contextual information you just don't know.

Now, even if the payload includes the type, in this case, that won't really help us either:

```
{ "type" = "(J//timestamp)Lyour/Timestamp;", "timestamp" = 573857303 <-- is this seconds or milliseconds? }
```

As it currently stands, parameter names are not considered when determining potential clashes in signatures (because multiple constructors with the same types and different names would simply not compile), however, since static factories can be used as unmarshallers, if we were to decide that names do factor in (likely not worthwhile given that not all external formats include names, so it would just introduce conflicts), one could possibly use the names of the parameters to disambiguate.

Want to add to the discussion?

things of interest related to the Java programming language



Marshalli: Data-Oriented Ser...

https://old.reddit.com/r/java/comments/1lsygm/marshalling\_dataoriented\_serialization/

[ - ] VirtualAgentsAreDumb 2 points 6 days ago  
That was a great presentation, thank you. A fellow serialization enthusiast here.  
One thing I'm interested in hearing more of is how to handle different versions of classes.  
Like, say you have a class that represents a timestamp, ie a specific moment in time. And the internal data is a long, representing the time since the epoch in seconds, ie "unix time". The marshal and unmarshal methods are super easy, as both handle a single long value.

[ - ] pfirmsto 1 point 5 days ago  
Around 12 years ago, I reimplemented a subset of Java serialization that explicitly used a standard constructor signature with a single argument that encapsulated name object tuples and gave each class in an object inheritance heirarchy private access. it didn't support circular object graphs.  
When a circular object graph was required, it was possible to create it after input validation and deserialization using a "serializer" similar to a serialization proxy, without requiring it be supported by the serialization framework.  
The framework was a public api designed to allow use of other serialization protocols.  
If I could share a lesson, it's simply this, do not consider support for circular object graphs under any circumstances.  
permalink embed save report reply

[ - ] viktorklang 2 points 5 days ago  
Indeed. Supporting circular object graphs is out of scope, deliberately, for this feature.  
permalink embed save parent report reply

The answer is, that without any contextual information you just don't know.  
Now, even if the payload includes the type, in this case, that won't really help us either:  

```
{ "type" = "(J//timestamp)Lyour/Timestamp;", "timestamp" = 573857303 <-- is this seconds or milliseconds? }
```

  
As it currently stands, parameter names are not considered when determining potential clashes in signatures (because multiple constructors with the same types and different names would simply not compile), however, since static factories can be used as unmarshallers, if we were to decide that names do factor in (likely not worthwhile given that not all external formats include names, so it would just introduce conflicts), one could possibly use the names of the parameters to disambiguate.

Viktor Klang (Ar...)

all 38 comments  
sorted by: best

Want to add to the discussion?

things of interest related to the Java programming language

### 3. Practical steps

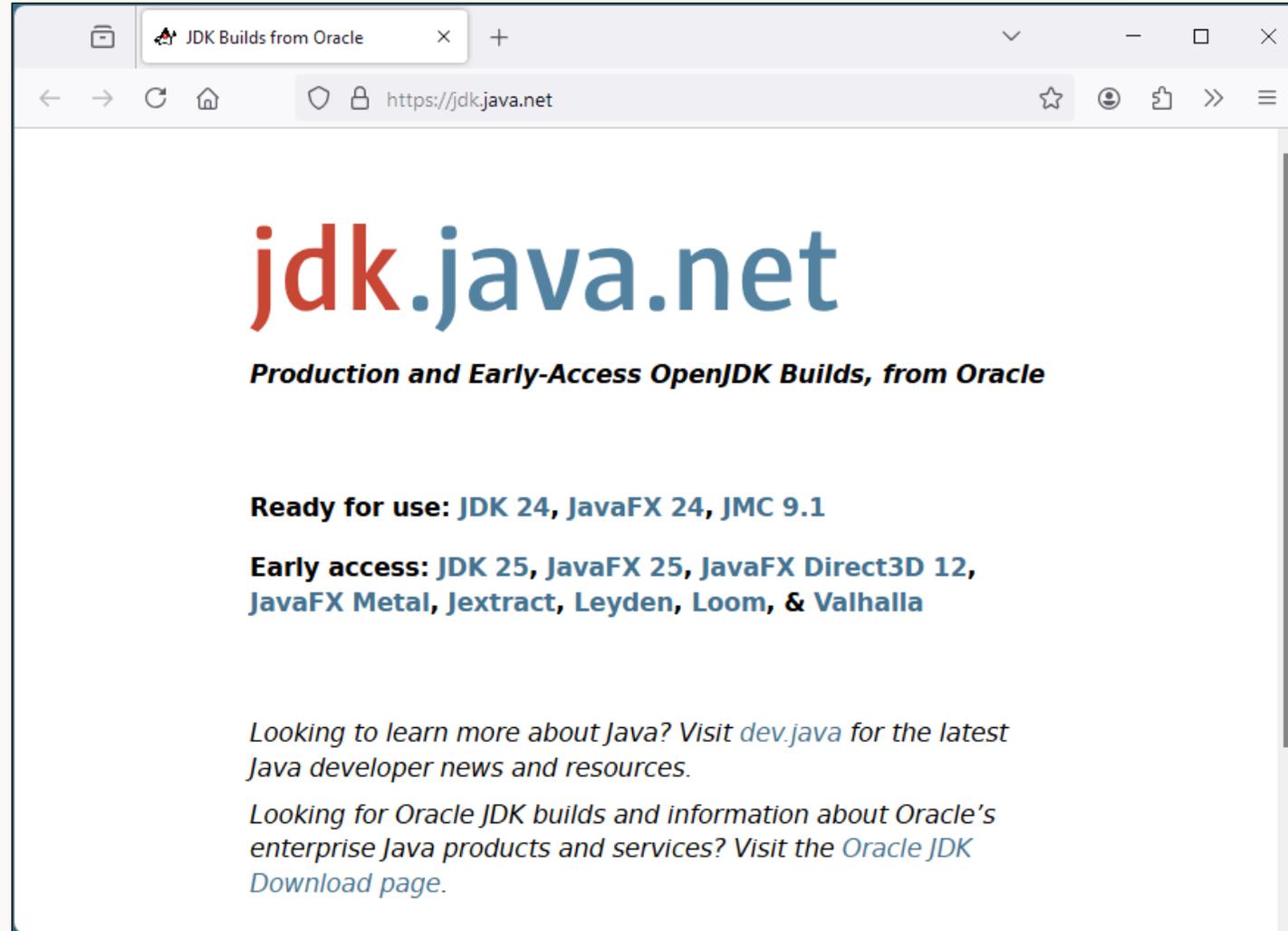
---



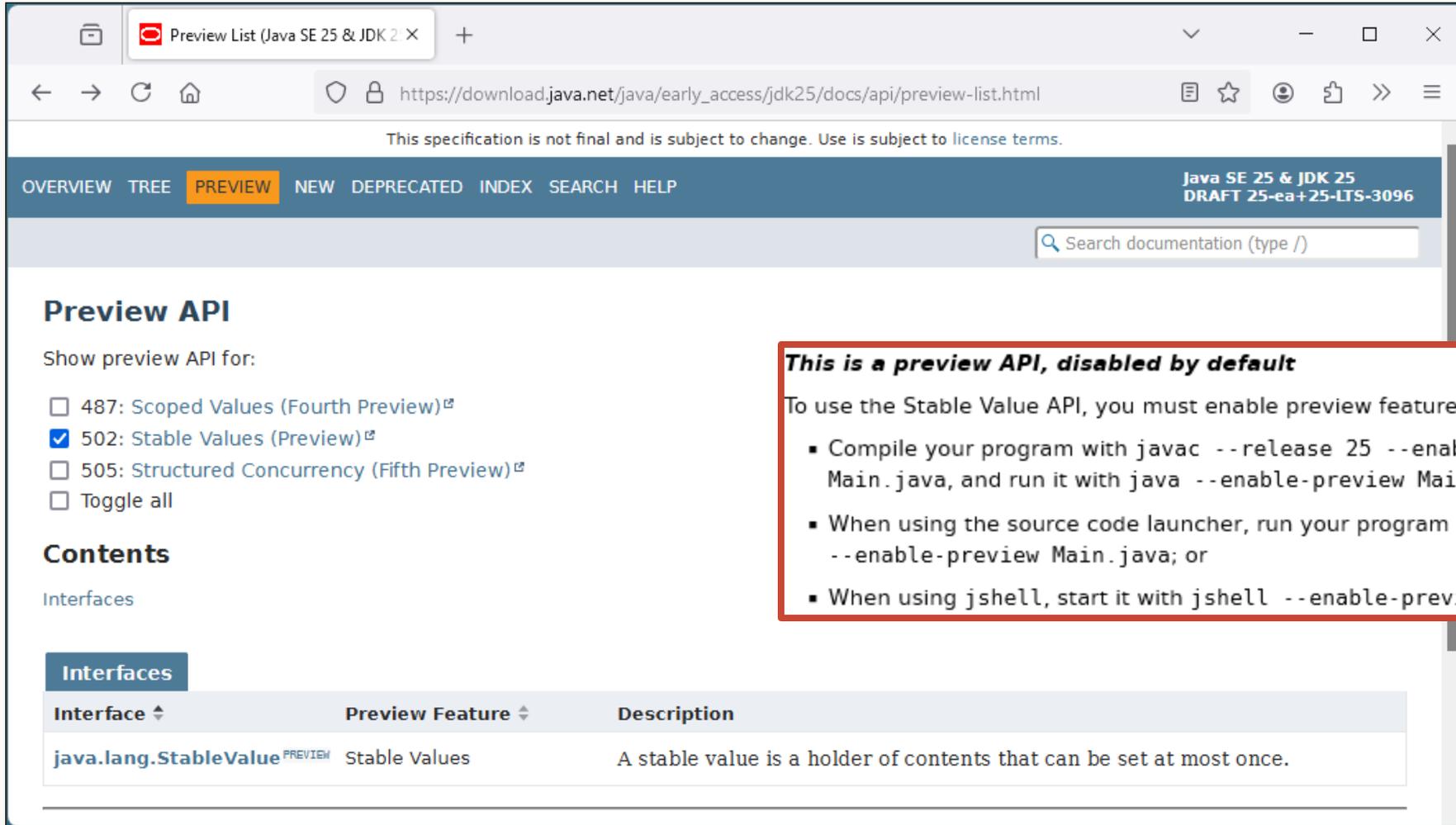
## Reporting your experience with a feature

1. Read the JEP carefully.
2. Download an Early Access JDK from [jdk.java.net](http://jdk.java.net).
3. Compile and run examples from the JEP.
4. Subscribe to the \*-dev list nominated by the JEP, and send your feedback.

# jdk.java.net: The home of Early Access open source binaries



# Most new Java features are in *preview*



This specification is not final and is subject to change. Use is subject to [license terms](#).

OVERVIEW TREE **PREVIEW** NEW DEPRECATED INDEX SEARCH HELP Java SE 25 & JDK 25  
DRAFT 25-ea+25-LTS-3096

Search documentation (type /)

## Preview API

Show preview API for:

- 487: Scoped Values (Fourth Preview) <sup>ⓘ</sup>
- 502: Stable Values (Preview) <sup>ⓘ</sup>
- 505: Structured Concurrency (Fifth Preview) <sup>ⓘ</sup>
- Toggle all

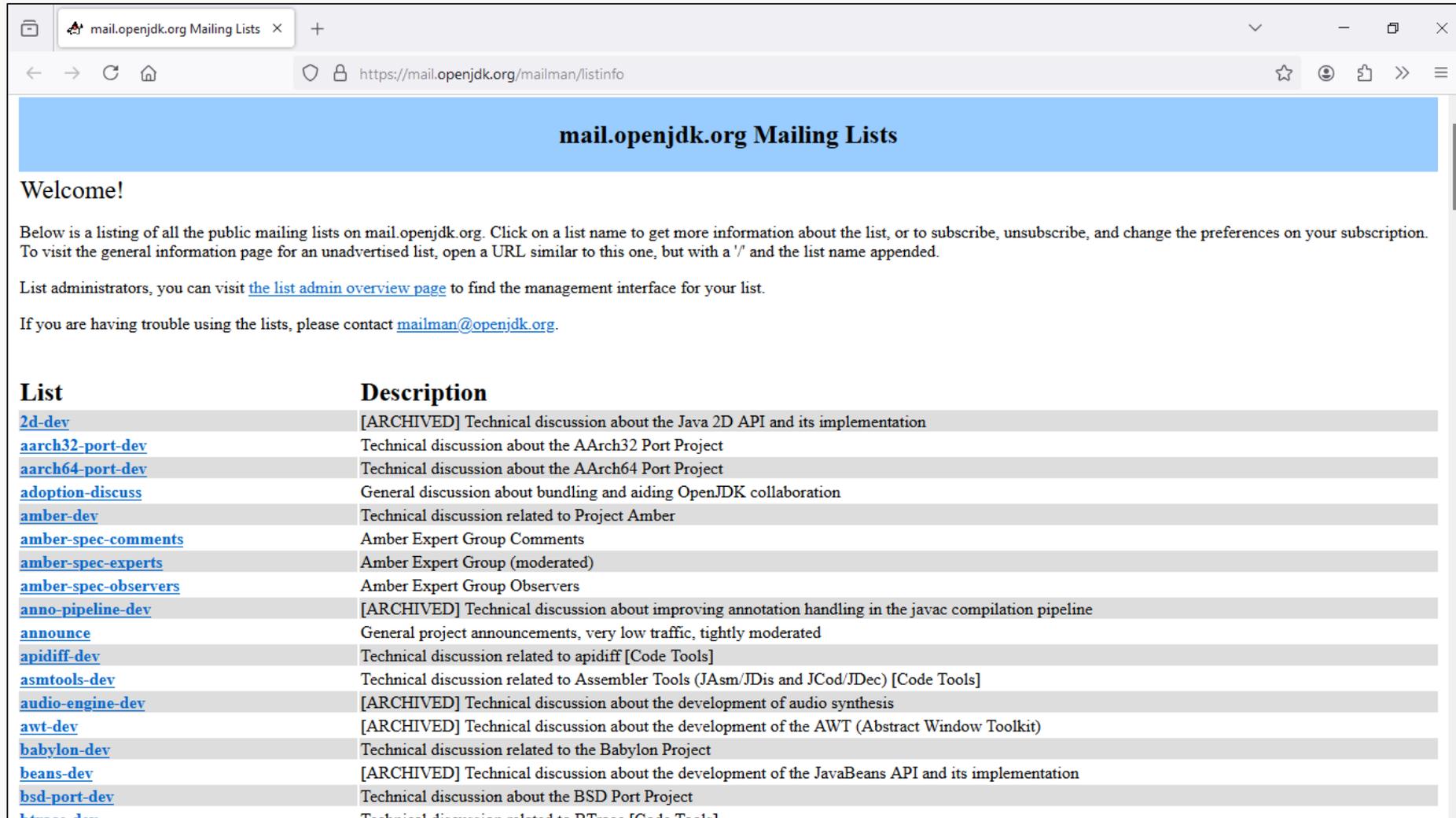
## Contents

Interfaces

**Interfaces**

Interface <sup>⌵</sup>	Preview Feature <sup>⌵</sup>	Description
<code>java.lang.StableValue</code> <sup>PREVIEW</sup>	Stable Values	A stable value is a holder of contents that can be set at most once.

# Subscribe to the appropriate \*-dev list (mail.openjdk.org)

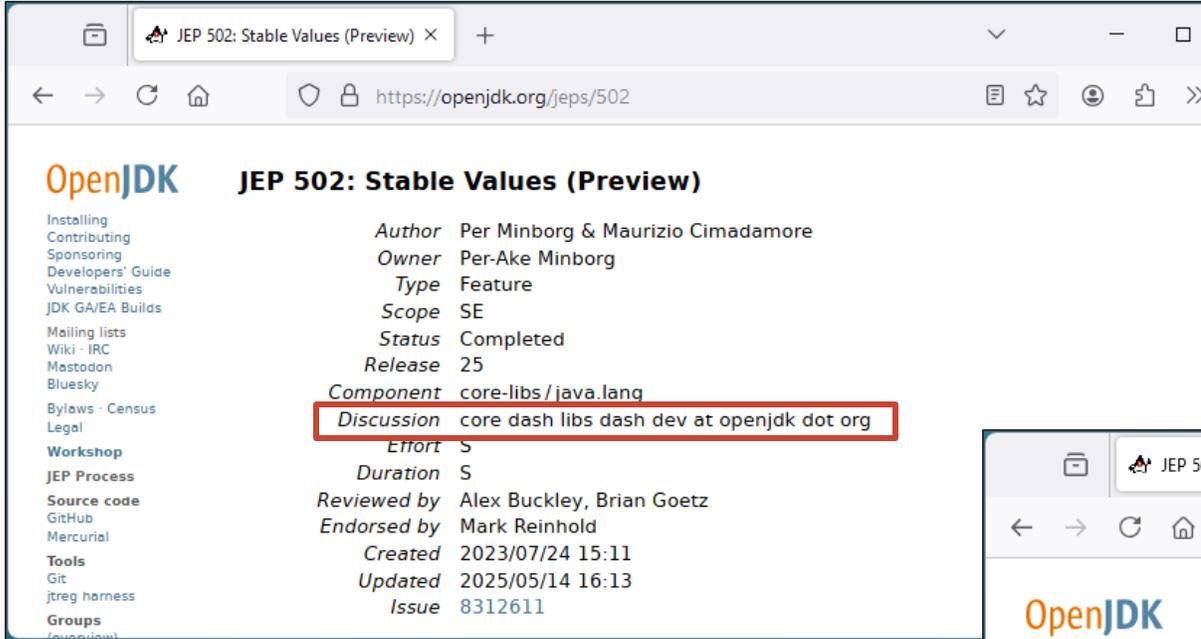


The screenshot shows a web browser window with the address bar displaying "https://mail.openjdk.org/mailman/listinfo". The page title is "mail.openjdk.org Mailing Lists". Below the title, there is a "Welcome!" message followed by instructions on how to use the mailing lists. A table lists various mailing lists with their descriptions.

List	Description
<a href="#">2d-dev</a>	[ARCHIVED] Technical discussion about the Java 2D API and its implementation
<a href="#">aarch32-port-dev</a>	Technical discussion about the AArch32 Port Project
<a href="#">aarch64-port-dev</a>	Technical discussion about the AArch64 Port Project
<a href="#">adoption-discuss</a>	General discussion about bundling and aiding OpenJDK collaboration
<a href="#">amber-dev</a>	Technical discussion related to Project Amber
<a href="#">amber-spec-comments</a>	Amber Expert Group Comments
<a href="#">amber-spec-experts</a>	Amber Expert Group (moderated)
<a href="#">amber-spec-observers</a>	Amber Expert Group Observers
<a href="#">anno-pipeline-dev</a>	[ARCHIVED] Technical discussion about improving annotation handling in the javac compilation pipeline
<a href="#">announce</a>	General project announcements, very low traffic, tightly moderated
<a href="#">apidiff-dev</a>	Technical discussion related to apidiff [Code Tools]
<a href="#">asmtools-dev</a>	Technical discussion related to Assembler Tools (JAsm/JDis and JCod/JDec) [Code Tools]
<a href="#">audio-engine-dev</a>	[ARCHIVED] Technical discussion about the development of audio synthesis
<a href="#">awt-dev</a>	[ARCHIVED] Technical discussion about the development of the AWT (Abstract Window Toolkit)
<a href="#">babylon-dev</a>	Technical discussion related to the Babylon Project
<a href="#">beans-dev</a>	[ARCHIVED] Technical discussion about the development of the JavaBeans API and its implementation
<a href="#">bsd-port-dev</a>	Technical discussion about the BSD Port Project
<a href="#">btrees-dev</a>	Technical discussion related to BTree [Code Tools]

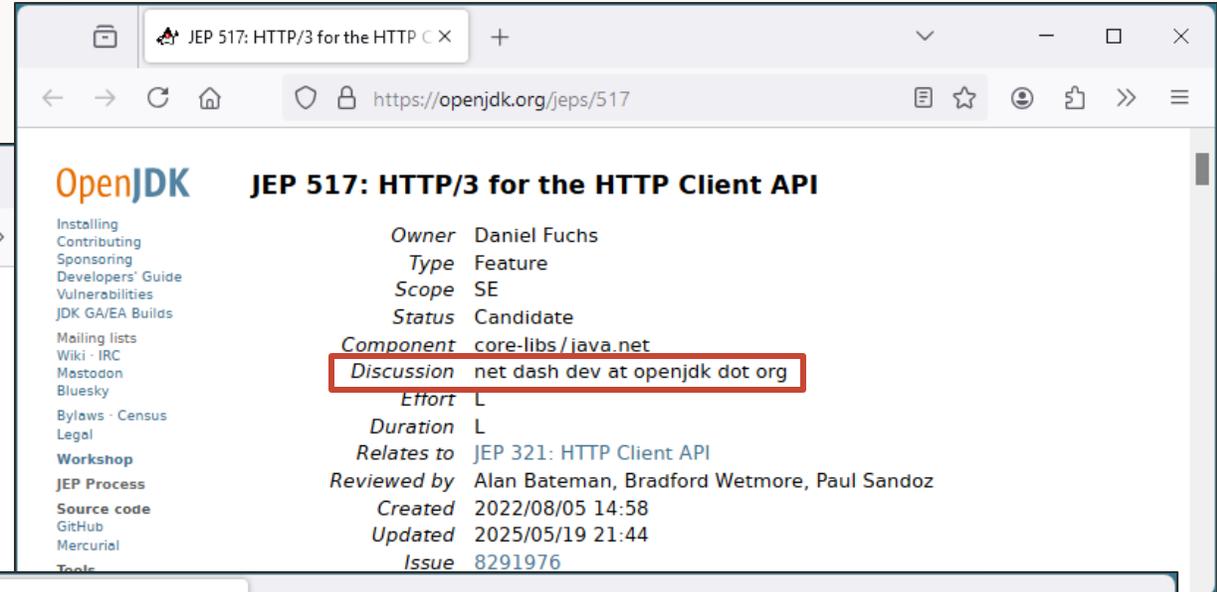


# Every JEP nominates a mailing list



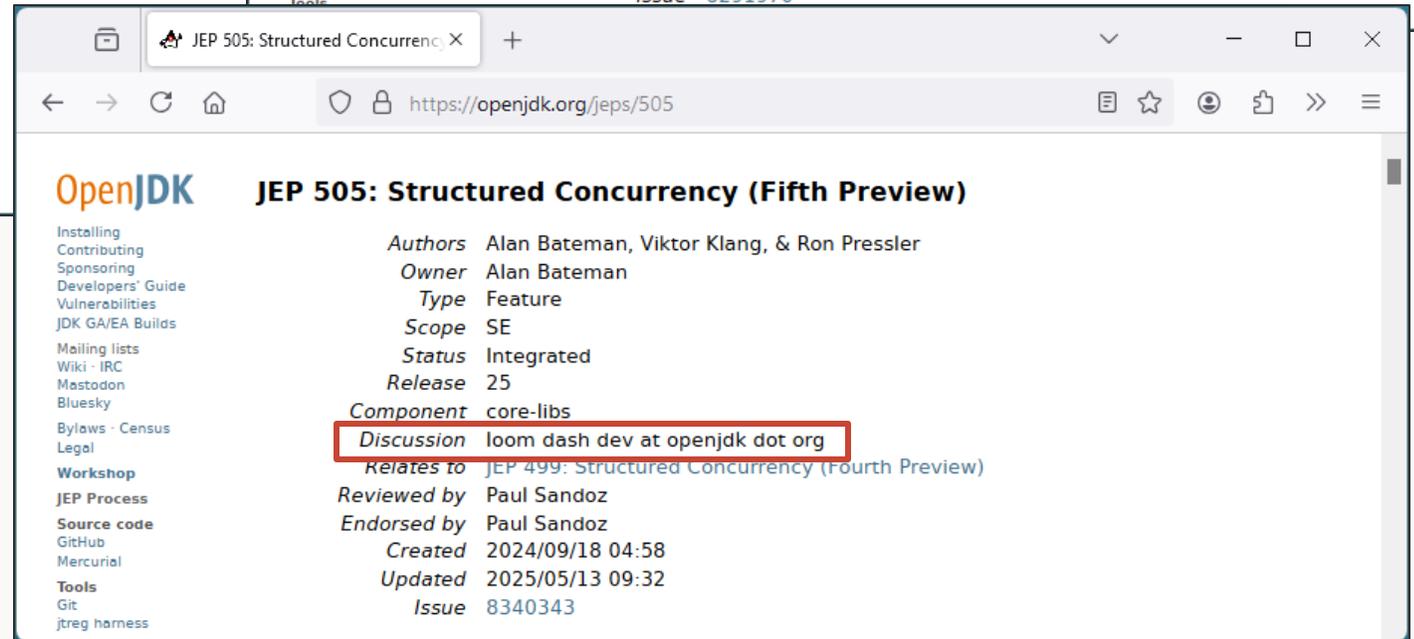
**OpenJDK** **JEP 502: Stable Values (Preview)**

Author Per Minborg & Maurizio Cimadamore  
Owner Per-Ake Minborg  
Type Feature  
Scope SE  
Status Completed  
Release 25  
Component core-libs / java.lang  
Discussion core dash libs dash dev at openjdk dot org  
Effort S  
Duration S  
Reviewed by Alex Buckley, Brian Goetz  
Endorsed by Mark Reinhold  
Created 2023/07/24 15:11  
Updated 2025/05/14 16:13  
Issue 8312611



**OpenJDK** **JEP 517: HTTP/3 for the HTTP Client API**

Owner Daniel Fuchs  
Type Feature  
Scope SE  
Status Candidate  
Component core-libs / java.net  
Discussion net dash dev at openjdk dot org  
Effort L  
Duration L  
Relates to JEP 321: HTTP Client API  
Reviewed by Alan Bateman, Bradford Wetmore, Paul Sandoz  
Created 2022/08/05 14:58  
Updated 2025/05/19 21:44  
Issue 8291976



**OpenJDK** **JEP 505: Structured Concurrency (Fifth Preview)**

Authors Alan Bateman, Viktor Klang, & Ron Pressler  
Owner Alan Bateman  
Type Feature  
Scope SE  
Status Integrated  
Release 25  
Component core-libs  
Discussion loom dash dev at openjdk dot org  
Relates to JEP 499: Structured Concurrency (Fourth Preview)  
Reviewed by Paul Sandoz  
Endorsed by Paul Sandoz  
Created 2024/09/18 04:58  
Updated 2025/05/13 09:32  
Issue 8340343



**Reporting your experience  
is the most valuable contribution  
you can make**