ORACLE

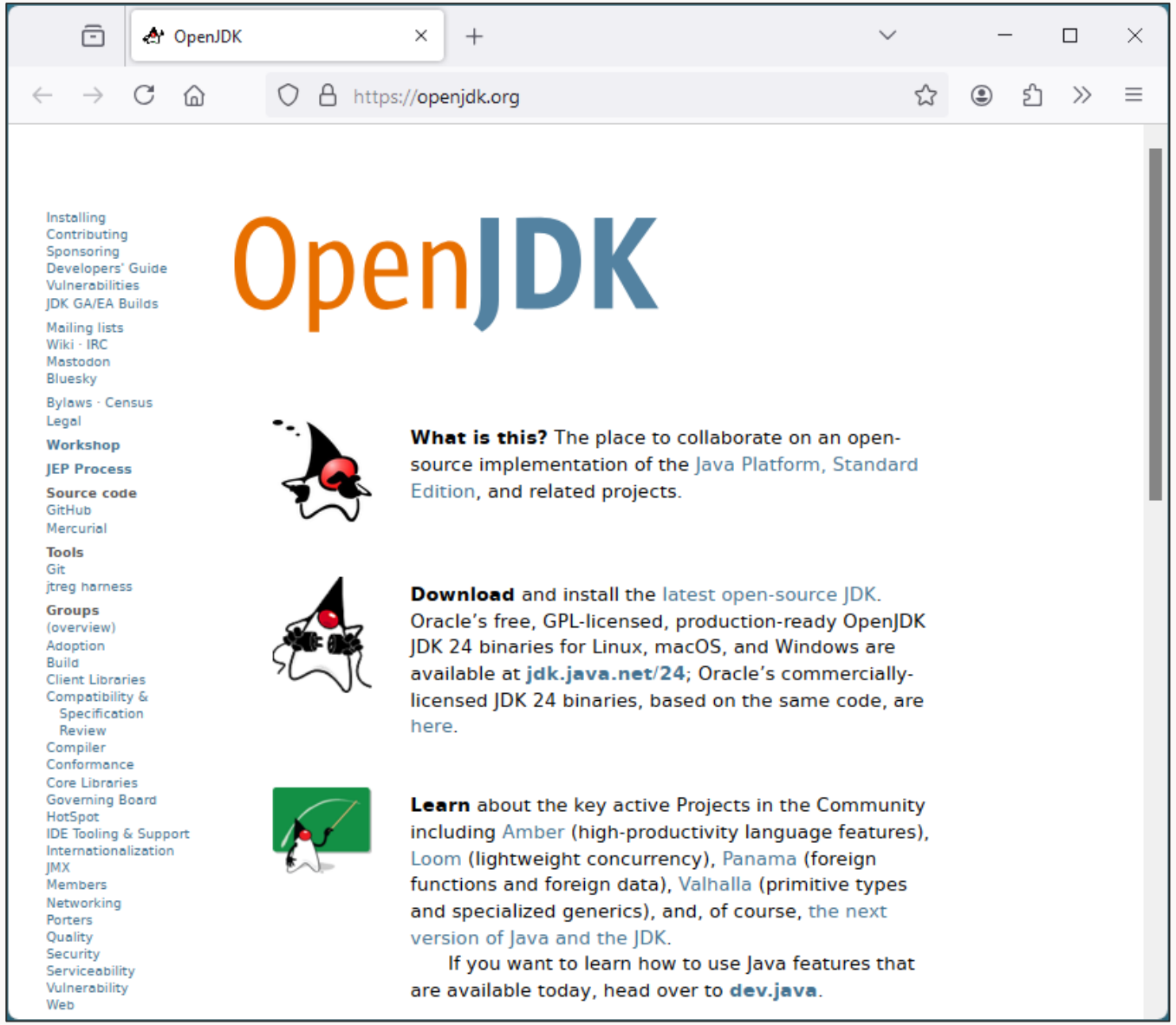
How to Contribute to OpenJDK Projects

Alex Buckley
Java Platform Group
Oracle
June 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

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
Tsan
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
Tsan
Valhalla
Verona
VisualVM
Wakefield
Zero
ZGC

JDK

https://openjdk.org/projects/jdk/

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

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

JDK 25

https://openjdk.org/projects/jdk/25/

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects
(overview, archive)
Amber
Babylon

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

JEPs proposed to target JDK 25	*review ends*
509: JFR CPU-Time Profiling (Experimental)	2025/06/04

JEPs targeted to JDK 25, so far

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

JDK 25

https://openjdk.org/projects/jdk/25/

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects
(overview, archive)
Amber
Babylon

JDK 25

This release will be the Reference Implementat
Platform, as specified by JSR 400 in the Java Co

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 Obje
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

JEP 3: JDK Release Process

https://openjdk.org/jeps/3

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality

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?
RDP 1	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
RDP 2	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
RC	Current P1 Targeted P1	Current P1, with approval	All P2–P5 Targeted P1	Current P1, with approval	No

JDK 25

https://openjdk.org/projects/jdk/25/

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability
Vulnerability
Web

Projects
(overview, archive)
Amber
Babylon

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

JEPs proposed to target JDK 25 — *review ends*

509: JFR CPU-Time Profiling (Experimental) — 2025/06/04

JEPs targeted to JDK 25, so far

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

JDK 25

JEP 520: JFR Method Timing &

https://openjdk.org/jeps/520

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
Conformance
Core Libraries
Governing Board
HotSpot
IDE Tooling & Support
Internationalization
JMX
Members
Networking
Porters
Quality
Security
Serviceability

JEP 520: JFR Method Timing & Tracing

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

Group Project JEP Code Documentation Experience

Harder *Easier*

Reporting your experience is the most valuable contribution you can make

The OpenJDK Developers' Guide

https://openjdk.org/guide/#contributing-to-an-openjdk-project

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?

Constructive critiques about usability

Bug report

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

https://old.reddit.com/r/java/comments/1kbzu6v/is_there_any_way_i_can_help_contribute_to_valh

36

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

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.

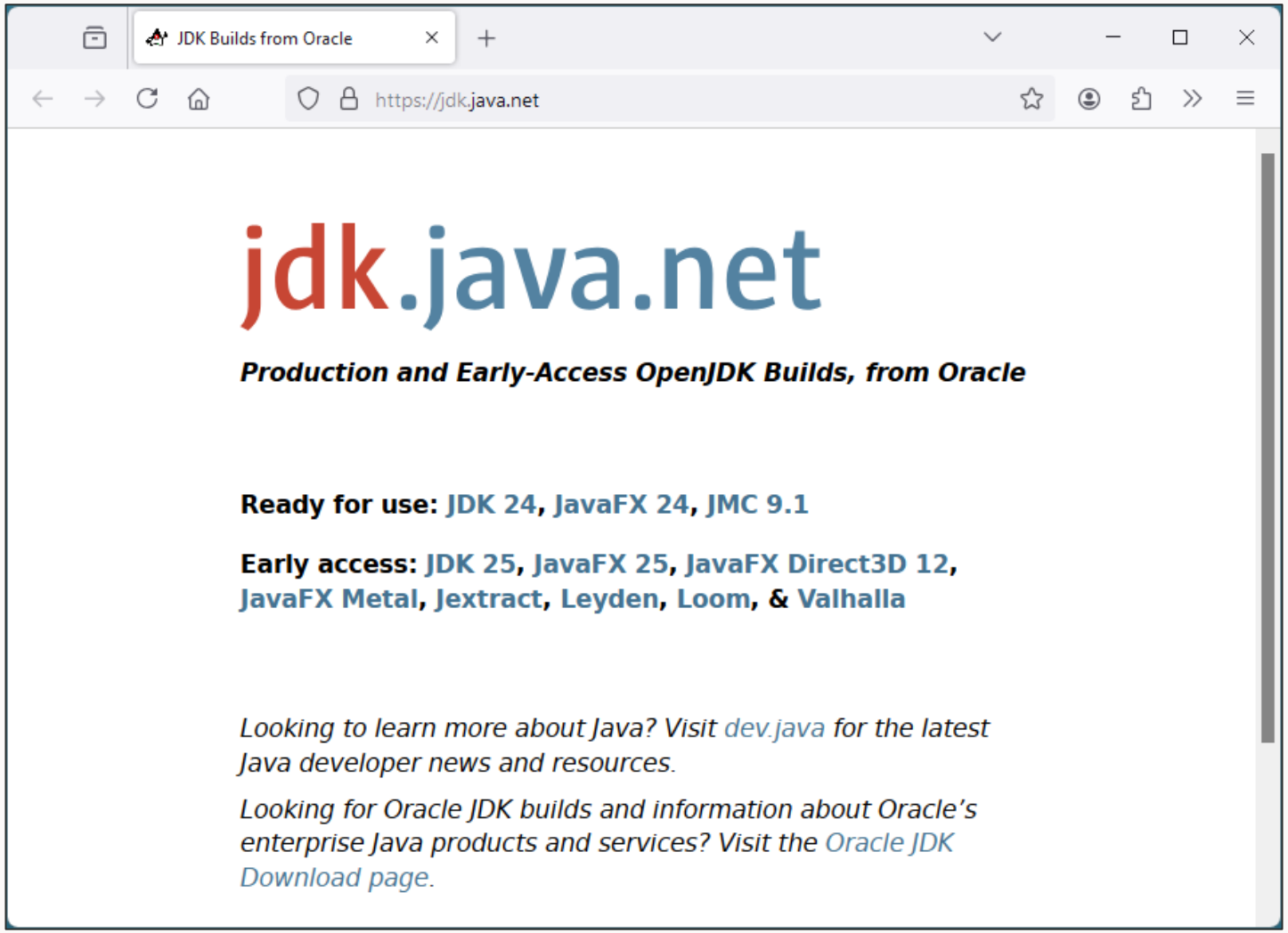
PLEASE SEEK HELP WITH JAVA PROGRAMMING IN /R/JAVAHELP!

3. Practical steps

Reporting your experience with a feature

1. Read the JEP carefully
2. Download an Early Access JDK from jdk.java.net
3. Compile and run examples from the JEP
4. Subscribe to the appropriate *-dev list and send your feedback

jdk.java.net: The home of Early Access binaries



Most new features are in *preview*

Preview List (Java SE 25 & JDK 2

https://download.java.net/java/early_access/jdk25/docs/api/preview-list.html

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)
- [x] 502: Stable Values (Preview)
- [ ] 505: Structured Concurrency (Fifth Preview)
- [ ] Toggle all

Contents

Interfaces

This is a preview API, disabled by default

To use the Stable Value API, you must enable preview features:

- Compile your program with `javac --release 25 --enable-preview Main.java`, and run it with `java --enable-preview Main`; or,
- When using the source code launcher, run your program with `java --enable-preview Main.java`; or
- When using `jshell`, start it with `jshell --enable-preview`.

Interfaces

Interface	Preview Feature	Description
java.lang.StableValue PREVIEW	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 Mailing Lists

https://mail.openjdk.org/mailman/listinfo

mail.openjdk.org Mailing Lists

Welcome!

Below is a listing of all the public mailing lists on mail.openjdk.org. Click on a list name to get more information about the list, or to subscribe, unsubscribe, and change the preferences on your subscription. To visit the general information page for an unadvertised list, open a URL similar to this one, but with a '/' and the list name appended.

List administrators, you can visit the list admin overview page to find the management interface for your list.

If you are having trouble using the lists, please contact mailman@openjdk.org.

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

Every JEP nominates a mailing list

JEP 502: Stable Values (Preview)

https://openjdk.org/jeps/502

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

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

JEP 517: HTTP/3 for the HTTP C

https://openjdk.org/jeps/517

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

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

JEP 505: Structured Concurrenc

https://openjdk.org/jeps/505

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

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