

Java Forum Nord / Hannover / 10.09.2024

Wie, schon wieder ein JDK Release? Die Neuerungen Live on Stage!

INNOQ



MICHAEL VITZ
SENIOR CONSULTANT

MICHAEL VITZ

Java Champion

Senior Consultant at INNOQ





Quick Recap



Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds
Mailing lists
Wiki · IRC

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

JDK 21

This release is the Reference Implementation of version 21 of the Java SE Platform, as specified by [JSR 396](#) in the Java Community Process.

JDK 21 reached [General Availability](#) on 19 September 2023. Production-ready binaries under the GPL are [available from Oracle](#); binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the [JEP Process](#), as amended by the [JEP 2.0 proposal](#). The release was produced using the [JDK Release Process \(JEP 3\)](#).

Features

430: [String Templates \(Preview\)](#)

431: [Sequenced Collections](#)

439: [Generational ZGC](#)

440: [Record Patterns](#)

441: [Pattern Matching for switch](#)

442: [Foreign Function & Memory API \(Third Preview\)](#)

443: [Unnamed Patterns and Variables \(Preview\)](#)

444: [Virtual Threads](#)

JDK 21

openjdk.org/projects/jdk/21/

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

Compiler Grammar

Developers' Guide

Device I/O

Duke

Galahad

Graal

IcedTea

JDK 7

JDK 8

JDK 8 Updates

JDK 9

JDK (... , 22, 23, 24)

JDK Updates

445: Unnamed Classes and Instance Main Methods (Preview)

446: Scoped Values (Preview)

448: Vector API (Sixth Incubator)

449: Deprecate the Windows 32-bit x86 Port for Removal

451: Prepare to Disallow the Dynamic Loading of Agents

452: Key Encapsulation Mechanism API

453: Structured Concurrency (Preview)

JDK 21 will be a long-term support (LTS) release from most vendors. For a complete list of the JEPs integrated since the previous LTS release, JDK 17, please see [here](#).

Schedule

2023/06/08	Rampdown Phase One (fork from main line)
2023/07/20	Rampdown Phase Two
2023/08/10	Initial Release Candidate
2023/08/24	Final Release Candidate
2023/09/19	General Availability

Last update: 2023/9/19 10:53 UTC



[Installing](#)
[Contributing](#)
[Sponsoring](#)
[Developers' Guide](#)
[Vulnerabilities](#)
[JDK GA/EA Builds](#)
[Mailing lists](#)
[Wiki · IRC](#)

[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](#)

JDK 22

This release is the Reference Implementation of version 22 of the Java SE Platform, as specified by [JSR 397](#) in the Java Community Process.

JDK 22 reached [General Availability](#) on 19 March 2024. Production-ready binaries under the GPL are [available from Oracle](#); binaries from other vendors will follow shortly.

The features and schedule of this release were proposed and tracked via the [JEP Process](#), as amended by the [JEP 2.0 proposal](#). The release was produced using the [JDK Release Process \(JEP 3\)](#).

Features

- [423: Region Pinning for G1](#)
- [447: Statements before super\(...\) \(Preview\)](#)
- [454: Foreign Function & Memory API](#)
- [456: Unnamed Variables & Patterns](#)
- [457: Class-File API \(Preview\)](#)
- [458: Launch Multi-File Source-Code Programs](#)
- [459: String Templates \(Second Preview\)](#)
- [460: Vector API \(Seventh Incubator\)](#)

JDK 22

openjdk.org/projects/jdk/22/

Incognito

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

Compiler Grammar

Developers' Guide

Device I/O

Duke

Galahad

Graal

IcedTea

JDK 7

JDK 8

JDK 8 Updates

JDK 9

JDK (... , 22, 23, 24)

JDK Updates

461: Stream Gatherers (Preview)

462: Structured Concurrency (Second Preview)

463: Implicitly Declared Classes and Instance Main Methods (Second Preview)

464: Scoped Values (Second Preview)

Schedule

2023/12/07

Rampdown Phase One (fork from main line)

2024/01/18

Rampdown Phase Two

2024/02/08

Initial Release Candidate

2024/02/22

Final Release Candidate

2024/03/19

General Availability

Last update: 2024/3/19 12:51 UTC



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 11: Incubator Modules

Authors Chris Hegarty, Alex Buckley

Owner Chris Hegarty

Type Process

Scope JDK

Status Active

Discussion jdk dash dev at openjdk dot java dot net

Effort S

Duration S

Reviewed by Alan Bateman, Alex Buckley, Brian Goetz, Paul Sandoz

Endorsed by Brian Goetz

Created 2016/11/16 09:17

Updated 2024/04/22 15:13

Issue [8169768](#)

Summary

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

Goals



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

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

JEP 12: Preview Features

<i>Owner</i>	Alex Buckley
<i>Type</i>	Process
<i>Scope</i>	SE
<i>Status</i>	Active
<i>Discussion</i>	jdk dash dev at openjdk dot java dot net
<i>Effort</i>	M
<i>Duration</i>	M
<i>Reviewed by</i>	Alan Bateman, Brian Goetz, Mark Reinhold
<i>Endorsed by</i>	Mark Reinhold
<i>Created</i>	2018/01/19 01:27
<i>Updated</i>	2024/04/22 15:13
<i>Issue</i>	8195734

Summary

A *preview feature* is a new feature of the Java language, Java Virtual Machine, or Java SE API that is fully specified, fully implemented, and yet impermanent. It is available in a JDK feature release to provoke developer feedback based on real world use; this may lead to it becoming permanent in a future Java SE Platform.

Goals



JDK 23



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

Mailing lists
Wiki · IRC

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

JDK 23

This release will be the Reference Implementation of version 23 of the Java SE Platform, as specified by [JSR 398](#) in the Java Community Process.

Status

JDK 23 is in the [Release Candidate](#) phase. The overall feature set is frozen. No further JEPs will be targeted to this release.

The stabilization branch, [jdk23](#), is open for critical bug fixes, [with approval](#), per the [JDK Release Process \(JEP 3\)](#). Integrate most stabilization changes via [backports](#) from the main line.

- [Release Candidate Bugs](#)
- [Fix-Request Process](#)
- [Bug-Deferral Process](#)
- [Bug fixes not backported from the main line](#)
- [Your bug fixes not backported from the main line](#)

Early-access builds under the GPL are available [here](#).

Schedule

2024/06/06

[Rampdown Phase One](#) (branch from main line)

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

Compiler Grammar

Developers' Guide

Device I/O

Duke

Galahad

Graal

IcedTea

JDK 7

JDK 8

Schedule

2024/06/06	Rampdown Phase One (branch from main line)
2024/07/18	Rampdown Phase Two
2024/08/08	Initial Release Candidate
2024/08/22	Final Release Candidate
2024/09/17	General Availability

Features

- 455: Primitive Types in Patterns, instanceof, and switch (Preview)
- 466: Class-File API (Second Preview)
- 467: Markdown Documentation Comments
- 469: Vector API (Eighth Incubator)
- 473: Stream Gatherers (Second Preview)
- 471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal
- 474: ZGC: Generational Mode by Default
- 476: Module Import Declarations (Preview)
- 477: Implicitly Declared Classes and Instance Main Methods (Third Preview)
- 480: Structured Concurrency (Third Preview)
- 481: Scoped Values (Third Preview)
- 482: Flexible Constructor Bodies (Second Preview)

JDK 23

openjdk.org/projects/jdk/23/

Incognito

Porters

Quality

Security

Serviceability

Vulnerability

Web

Projects

(overview, archive)

Amber

Babylon

CRaC

Code Tools

Coin

Common VM

Interface

Compiler Grammar

Developers' Guide

Device I/O

Duke

Galahad

Graal

IcedTea

JDK 7

JDK 8

JDK 8 Updates

JDK 9

JDK (... , 22, 23, 24)

JDK Updates

JavaDoc.Next

Jigsaw

Kona

Kulla

Lanai

Leyden

Lilliput

Locale Enhancement

Features

455: Primitive Types in Patterns, instanceof, and switch (Preview)

466: Class-File API (Second Preview)

467: Markdown Documentation Comments

469: Vector API (Eighth Incubator)

473: Stream Gatherers (Second Preview)

471: Deprecate the Memory-Access Methods in sun.misc.Unsafe for Removal

474: ZGC: Generational Mode by Default

476: Module Import Declarations (Preview)

477: Implicitly Declared Classes and Instance Main Methods (Third Preview)

480: Structured Concurrency (Third Preview)

481: Scoped Values (Third Preview)

482: Flexible Constructor Bodies (Second Preview)

Last update: 2024/8/12 16:47 UTC

<https://openjdk.java.net/projects/jdk/23>



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 465: String Templates (Third Preview)

<i>Author</i>	Jim Laskey
<i>Owner</i>	Gavin Bierman
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Withdrawn
<i>Component</i>	specification / language
<i>Discussion</i>	amber dash dev at openjdk dot org
<i>Effort</i>	M
<i>Duration</i>	M
<i>Relates to</i>	JEP 459: String Templates (Second Preview)
<i>Reviewed by</i>	Mark Reinhold
<i>Created</i>	2024/01/09 16:54
<i>Updated</i>	2024/06/21 08:56
<i>Issue</i>	8323333

Summary

Enhance the Java programming language with *string templates*. String templates complement Java's existing string literals and text blocks by coupling literal text with embedded expressions and *template processors* to produce specialized results.



JEPs



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 474: ZGC: Generational Mode by Default

<i>Owner</i>	Axel Boldt-Christmas
<i>Type</i>	Feature
<i>Scope</i>	Implementation
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	hotspot / gc
<i>Discussion</i>	hotspot dash gc dash dev at openjdk dot org
<i>Effort</i>	XS
<i>Reviewed by</i>	Stefan Karlsson, Vladimir Kozlov
<i>Endorsed by</i>	Vladimir Kozlov
<i>Created</i>	2024/02/26 11:16
<i>Updated</i>	2024/07/31 08:49
<i>Issue</i>	8326667

Summary

Switch the default mode of the Z Garbage Collector (ZGC) to the generational mode. Deprecate the non-generational mode, with the intent to remove it in a future release.

Goals



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 471: Deprecate the Memory-Access Methods in `sun.misc.Unsafe` for Removal

<i>Author</i>	Ron Pressler & Alex Buckley
<i>Owner</i>	Ron Pressler
<i>Type</i>	Feature
<i>Scope</i>	JDK
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs
<i>Discussion</i>	jdk dash dev at openjdk dot org
<i>Reviewed by</i>	Alan Bateman, Brian Goetz, Mark Reinhold, Maurizio Cimadamore, Paul Sandoz
<i>Endorsed by</i>	Alan Bateman
<i>Created</i>	2024/01/05 15:46
<i>Updated</i>	2024/05/31 10:40
<i>Issue</i>	8323072

Summary

Deprecate the memory-access methods in `sun.misc.Unsafe` for removal in a future release. These unsupported methods have been superseded by standard APIs, namely the `VarHandle` API ([JEP 193](#), JDK 9) and the `Foreign Function &`

```
// 'allocateMemory(long)' is deprecated since version 23 and marked for removal  
sun.misc.Unsafe.getUnsafe().allocateMemory(1);
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 467: Markdown Documentation Comments

<i>Owner</i>	Jonathan Gibbons
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	tools / javadoc(tool)
<i>Discussion</i>	javadoc dash dev at openjdk dot org
<i>Reviewed by</i>	Ron Pressler
<i>Endorsed by</i>	Paul Sandoz
<i>Created</i>	2023/09/11 17:45
<i>Updated</i>	2024/08/26 17:52
<i>Issue</i>	8316039

Summary

Enable JavaDoc documentation comments to be written in Markdown rather than solely in a mixture of HTML and JavaDoc @-tags.

Goals

- Make API documentation comments easier to write and easier to read in source form by introducing the ability to use Markdown syntax in

```
/// This is *important* javadoc comment
/// ```
/// Example code looks like that
///
/// public static void main(String args[]) {...}
/// ```
///
/// Lists are possible:
///
/// - item
/// - item2
///
/// And links to e.g. [String][java.lang.String] are possible, too.
public class MarkdownJavadoc {
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 455: Primitive Types in Patterns, instanceof, and switch (Preview)

<i>Owner</i>	Angelos Bimpoudis
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	specification / language
<i>Discussion</i>	amber dash dev at openjdk dot org
<i>Effort</i>	M
<i>Duration</i>	M
<i>Reviewed by</i>	Alex Buckley, Brian Goetz
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2022/06/15 10:05
<i>Updated</i>	2024/07/16 14:19
<i>Issue</i>	8288476

Summary

Enhance pattern matching by allowing primitive type patterns in all pattern contexts, and extend instanceof and switch to work with all primitive types. This is a [preview language feature](#).

```
Object o = 42;

var result = switch (o) {
    case int i when i > 50 -> "Found a big number doubled to: " + i * 2;
    case int i -> "Found a number: " + i;
    default -> "Some object";
};
System.out.println(result); // Found a number: 42

record JsonNumber(double value) {}
var json = new JsonNumber(42.0d);

result = switch (json) {
    case JsonNumber(int i) -> "Integer " + i;
    case JsonNumber(double d) -> "Double " + d;
};
System.out.println(result); // Integer 42

long l = 127L;
if (l instanceof byte b) {
    System.out.println("Byte " + b); // Byte 127
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &
Specification
Review

Compiler

Conformance

Core Libraries

JEP 482: Flexible Constructor Bodies (Second Preview)

<i>Author</i>	Archie Cobbs & Gavin Bierman
<i>Owner</i>	Archie Cobbs
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	specification / language
<i>Discussion</i>	amber dash dev at openjdk dot org
<i>Relates to</i>	JEP 447: Statements before super(...) (Preview)
<i>Reviewed by</i>	Alex Buckley, Brian Goetz
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2024/02/13 22:18
<i>Updated</i>	2024/07/08 14:28
<i>Issue</i>	8325803

Summary

In constructors in the Java programming language, allow statements to appear before an explicit constructor invocation, i.e., `super(..)` or `this(..)`. The statements cannot reference the instance under construction, but they can initialize its fields. Initializing fields before invoking another constructor makes a

```
public class FlexibleConstructorBodies {  
    static class Parent {  
        private final int i;  
  
        protected Parent(int i) {  
            this.i = i;  
        }  
    }  
  
    static class Child extends Parent {  
        Child(int i) {  
            if (i > 42) {  
                throw new RuntimeException();  
            }  
            super(i);  
        }  
    }  
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

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

JEP 473: Stream Gatherers (Second Preview)

<i>Owner</i>	Viktor Klang
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs / java.util.stream
<i>Discussion</i>	core dash libs dash dev at openjdk dot org
<i>Effort</i>	XS
<i>Duration</i>	XS
<i>Relates to</i>	JEP 461: Stream Gatherers (Preview) JEP 485: Stream Gatherers
<i>Reviewed by</i>	Alan Bateman, Paul Sandoz
<i>Endorsed by</i>	Paul Sandoz
<i>Created</i>	2024/03/11 19:39
<i>Updated</i>	2024/09/02 16:05
<i>Issue</i>	8327844

Summary

Enhance the [Stream API](#) to support custom intermediate operations. This will allow stream pipelines to transform data in ways that are not easily achievable with the

```
public class Gatherers {

    public static void main(String[] args) {
        Stream.of(2, 1, 5, 3)
            .gather(map((Integer element) -> element * 2).andThen(filter(element -> element > 2)))
            .forEach(System.out::println);
    }

    static <T, R> Gatherer<T, Void, R> map(Function<T, R> function) {
        return Gatherer.of(
            (_, element, downstream) ->
                downstream.push(function.apply(element));
        );
    }

    static <T> Gatherer<T, Void, T> filter(Predicate<T> predicate) {
        return Gatherer.of(
            (_, element, downstream) -> {
                if (!predicate.test(element)) {
                    return downstream.isRejecting();
                }
                return downstream.push(element);
            }
        );
    }
}
```

```

public class MaxGatherer {

    public static void main(String[] args) {
        Stream.of(2, 1, 5, 3)
            .gather(max())
            .findFirst()
            .ifPresent(System.out::println);
    }
}

```

```

static Gatherer<Integer, ?, Integer> max() {
    class State {
        Integer value;
        boolean hasValue;
    }

    return Gatherer.of(
        State::new,
        Gatherer.Integrator.ofGreedy((state, element, _) -> {
            if (!state.hasValue) {
                state.value = element;
                state.hasValue = true;
            } else {
                state.value = Math.max(state.value, element);
            }
            return true;
        }),
        (left, right) -> {
            if (!left.hasValue) {
                return right;
            } else if (!right.hasValue) {
                return left;
            } else {
                left.value = Math.max(left.value, right.value);
                return left;
            }
        },
        (state, downstream) -> {
            if (state.hasValue) {
                downstream.push(state.value);
            }
        }
    ));
}

```



Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds
Mailing lists
Wiki · IRC
Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

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

JEP 477: Implicitly Declared Classes and Instance Main Methods (Third Preview)

Authors Ron Pressler, Jim Laskey, & Gavin Bierman

Owner Gavin Bierman

Type Feature

Scope SE

Status Closed / Delivered

Release 23

Component specification / language

Discussion amber dash dev at openjdk dot org

Effort S

Duration S

Relates to [JEP 463: Implicitly Declared Classes and Instance Main Methods \(Second Preview\)](#)

[JEP 476: Module Import Declarations \(Preview\)](#)

Reviewed by Brian Goetz

Endorsed by Brian Goetz

Created 2024/01/09 17:02

Updated 2024/09/03 13:42

Issue [8323335](#)

```
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello");  
    }  
}
```



```
void main() {  
    IO.println("Hello");  
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 481: Scoped Values (Third Preview)

<i>Owner</i>	Andrew Haley
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs
<i>Discussion</i>	loom dash dev at openjdk dot org
<i>Relates to</i>	JEP 464: Scoped Values (Second Preview)
<i>Reviewed by</i>	Alan Bateman
<i>Endorsed by</i>	Paul Sandoz
<i>Created</i>	2024/04/24 14:31
<i>Updated</i>	2024/08/15 17:08
<i>Issue</i>	8331056

Summary

Introduce *scoped values*, which enable a method to share immutable data both with its callees within a thread, and with child threads. Scoped values are easier to reason about than thread-local variables. They also have lower space and time costs, especially when used together with virtual threads ([JEP 444](#)) and structured concurrency ([JEP 480](#)). This is a [preview API](#).

```
ScopedValue<Integer> F00 = ScopedValue.newInstance( );  
  
ScopedValue.runWhere(F00, 42, ( ) -> {  
    System.out.println(F00.get( )); // can be done anywhere within the call stack  
});
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 480: Structured Concurrency (Third Preview)

<i>Author</i>	Ron Pressler & Alan Bateman
<i>Owner</i>	Alan Bateman
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs
<i>Discussion</i>	loom dash dev at openjdk dot org
<i>Relates to</i>	JEP 462: Structured Concurrency (Second Preview)
<i>Reviewed by</i>	Paul Sandoz
<i>Endorsed by</i>	Paul Sandoz
<i>Created</i>	2024/04/22 12:11
<i>Updated</i>	2024/07/16 15:03
<i>Issue</i>	8330818

Summary

Simplify concurrent programming by introducing an API for *structured concurrency*. Structured concurrency treats groups of related tasks running in different threads as a single unit of work, thereby streamlining error handling and cancellation, improving reliability, and enhancing observability. This is a [preview API](#).

```
try (var scope = new StructuredTaskScope.ShutdownOnFailure()) {  
    scope.fork(() -> ... );  
    scope.fork(() -> ... );  
    scope.fork(() -> ... );  
    scope.join().throwIfFailed(); // waits until all three finished or one throwed exception  
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

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

JEP 466: Class-File API (Second Preview)

<i>Author</i>	Brian Goetz
<i>Owner</i>	Adam Sotona
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs / java.lang.classfile
<i>Discussion</i>	core dash libs dash dev at openjdk dot org
<i>Effort</i>	S
<i>Duration</i>	M
<i>Relates to</i>	JEP 457: Class-File API (Preview) JEP 484: Class-File API
<i>Reviewed by</i>	Alex Buckley, Paul Sandoz
<i>Endorsed by</i>	Paul Sandoz
<i>Created</i>	2024/01/30 13:45
<i>Updated</i>	2024/08/27 13:30
<i>Issue</i>	8324965

Summary

Provide a standard API for parsing, generating, and transforming Java class files.

```
try (var in = String.class.getResourceAsStream( "/java/lang/String.class" )) {  
    var classModel = ClassFile.of().parse(in.readAllBytes());  
  
    System.out.println( "## Methods" );  
    classModel.methods().stream()  
        .map(method -> method.methodName().stringValue())  
        .map(methodName -> " * " + methodName)  
        .forEach(System.out::println);  
  
    System.out.println( "## Fields" );  
    classModel.fields().stream()  
        .map(field -> field.fieldName().stringValue())  
        .map(methodName -> " * " + methodName)  
        .forEach(System.out::println);  
}
```

```
public class EchoGenerator {

    public static void main(String[] args) throws IOException {
        var system = of("java.lang", "System");
        var printStream = of("java.io", "PrintStream");
        ClassFile.of().buildTo(
            Path.of("Echo.class"),
            of("Echo"),
            classBuilder -> classBuilder
                .withMethodBody(
                    "main",
                    MethodTypeDesc.of(CD_void, CD_String.arrayType()),
                    ACC_PUBLIC | ACC_STATIC,
                    codeBuilder -> codeBuilder
                        .getstatic(system, "out", printStream)
                        .aload(codeBuilder.parameterSlot(0))
                        .iconst_0()
                        .aload()
                        .invokevirtual(printStream, "println", MethodTypeDesc.of(CD_void, CD_String))
                        .return_()));
    }
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 476: Module Import Declarations (Preview)

<i>Author</i>	Jim Laskey & Gavin Bierman
<i>Owner</i>	Gavin Bierman
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	specification / language
<i>Discussion</i>	amber dash dev at openjdk dot org
<i>Effort</i>	S
<i>Duration</i>	S
<i>Relates to</i>	JEP 477: Implicitly Declared Classes and Instance Main Methods (Third Preview)
<i>Reviewed by</i>	Alex Buckley, Brian Goetz
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2023/08/28 16:57
<i>Updated</i>	2024/09/03 13:42
<i>Issue</i>	8315129

Summary

Enhance the Java programming language with the ability to succinctly import all of

```
import module java.base;

// no imports for e.g. java.util.* needed

public class ModuleImport {

    public static void main(String[] args) {
        List.of();
        new HashMap<String, String>( );
    }
}
```



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 469: Vector API (Eighth Incubator)

<i>Owner</i>	Paul Sandoz
<i>Type</i>	Feature
<i>Scope</i>	JDK
<i>Status</i>	Closed / Delivered
<i>Release</i>	23
<i>Component</i>	core-libs
<i>Discussion</i>	panama dash dev at openjdk dot org
<i>Effort</i>	XS
<i>Duration</i>	XS
<i>Relates to</i>	JEP 460: Vector API (Seventh Incubator)
<i>Reviewed by</i>	Vladimir Ivanov
<i>Endorsed by</i>	John Rose
<i>Created</i>	2024/02/27 20:50
<i>Updated</i>	2024/07/15 15:59
<i>Issue</i>	8326878

Summary

Introduce an API to express vector computations that reliably compile at runtime to optimal vector instructions on supported CPU architectures, thus achieving performance superior to equivalent scalar computations.



API Changes



The Java Version Almanac

javaalmanac.io

GitHub



The Java Version Almanac / JDK Releases / Java 23 /

[Feedback on this page?](#)

New APIs in Java 23

Comparing **Java 23** (23+37-2369-open) with **Java 22** (22.0.2+9-tem).

Element	Modification
📁 java.base	
📁 java.io	
🕒 Console	
● format(Locale, String, Object...)	added 📺
● print(Object)	added preview 📺
● printf(Locale, String, Object...)	added 📺
● println(Object)	added preview 📺
● readLine(Locale, String, Object...)	added 📺
● readPassword(Locale, String, Object...)	added 📺
● readln(String)	added preview 📺

New APIs in Java 23	
Comparing Java 23 (23+37-2369-open) with Java 22 (22.0.2+9-tem).	
Element	Modification
📁 java.base	
📁 java.io	
🕒 Console	
● format(Locale, String, Object...)	added 🗒
● print(Object)	added preview 🗒
● printf(Locale, String, Object...)	added 🗒
● println(Object)	added preview 🗒
● readLine(Locale, String, Object...)	added 🗒
● readPassword(Locale, String, Object...)	added 🗒
● readln(String)	added preview 🗒
🕒 IO	added preview 🗒
🕒 ObjectOutputStream.PutField	
● write(ObjectOutput)	+ forRemoval
📁 java.lang.classfile.constantpool	
🕒 ConstantPool	
● entryByIndex(int, Class)	added preview 🗒

● get(Supplier)	removed preview
Ⓢ ScopedValue	
● callWhere(ScopedValue, Object, Callable)	removed preview
● callWhere(ScopedValue, Object, ...)	added preview
● getWhere(ScopedValue, Object, Supplier)	removed preview
Ⓢ ThreadGroup	
● resume()	removed
● stop()	removed
● suspend()	removed
Ⓢ Thread	
● resume()	removed
● suspend()	removed
ⓘ ScopedValue.CallableOp	added preview
ⓘ StringTemplate.Processor.Linkage	removed preview
ⓘ StringTemplate.Processor	removed preview
ⓘ StringTemplate	removed preview
📦 java.net	
Ⓢ DatagramSocketImpl	
● getTTL()	+ forRemoval
● setTTL(byte)	+ forRemoval

● isStrict()	added
● setStrict(boolean)	added
SimpleDateFormat	
● toString()	added
java.time	
Instant	
● until(Instant)	added
java.util.zip	
Deflater	
● getTotalIn()	+ deprecated
● getTotalOut()	+ deprecated
Inflater	
● getTotalIn()	+ deprecated
● getTotalOut()	+ deprecated
ZipFile	
● toString()	added
java.util	
FormatProcessor	removed preview
java.compiler	
javax.lang.model.util	



JDK 24

OpenJDK

Installing

Contributing

Sponsoring

Developers' Guide

Vulnerabilities

JDK GA/EA Builds

Mailing lists

Wiki · IRC

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

JDK 24

This release will be the Reference Implementation of version 24 of the Java SE Platform, as specified by [JSR 399](#) in the Java Community Process.

Status

The [main line](#) is open for bug fixes, small enhancements, and JEPs as proposed and tracked via the [JEP Process](#).

Features

JEPs targeted to JDK 24, so far

472: [Prepare to Restrict the Use of JNI](#)

Last update: 2024/7/16 12:53 UTC



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &
Specification
Review

Compiler

Conformance

Core Libraries

JEP 472: Prepare to Restrict the Use of JNI

<i>Owner</i>	Ron Pressler
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Targeted
<i>Release</i>	24
<i>Component</i>	core-libs
<i>Discussion</i>	jdk dash dev at openjdk dot org
<i>Relates to</i>	JEP 454: Foreign Function & Memory API
<i>Reviewed by</i>	Alex Buckley, Dan Heidinga, Jorn Vernee, Mark Reinhold, Maurizio Cimadamore
<i>Endorsed by</i>	Alan Bateman
<i>Created</i>	2023/05/03 09:08
<i>Updated</i>	2024/09/07 21:48
<i>Issue</i>	8307341

Summary

Issue warnings about uses of the [Java Native Interface \(JNI\)](#) and adjust the [Foreign Function & Memory \(FFM\) API](#) to issue warnings in a consistent manner. All such warnings aim to prepare developers for a future release that ensures [integrity by default](#) by uniformly restricting JNI and the FFM API. Application developers can



Future



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

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

JEP 468: Derived Record Creation (Preview)

<i>Author</i>	Gavin Bierman & Brian Goetz
<i>Owner</i>	Gavin Bierman
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Candidate
<i>Component</i>	specification / language
<i>Discussion</i>	amber dash dev at openjdk dot org
<i>Reviewed by</i>	Alex Buckley, Brian Goetz
<i>Endorsed by</i>	Brian Goetz
<i>Created</i>	2023/11/30 17:45
<i>Updated</i>	2024/04/23 13:37
<i>Issue</i>	8321133

Summary

Enhance the Java language with derived creation for records. Records are immutable objects, so developers frequently create new records from old records to model new data. Derived creation streamlines code by deriving a new record from an existing record, specifying only the components that are different. This is a [preview language feature](#).



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP 401: Value Classes and Objects (Preview)

<i>Owner</i>	Dan Smith
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Draft
<i>Component</i>	specification
<i>Discussion</i>	valhalla dash dev at openjdk dot java dot net
<i>Effort</i>	XL
<i>Duration</i>	XL
<i>Reviewed by</i>	Brian Goetz
<i>Created</i>	2020/08/13 19:31
<i>Updated</i>	2024/07/25 00:38
<i>Issue</i>	8251554

Summary

Enhance the Java Platform with *value objects*, class instances that have only final fields and lack object identity. This is a [preview language and VM feature](#).

Goals

- Allow developers to opt in to a programming model for simple values in which objects are distinguished solely by their field values, much as the



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

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

JEP 483: Ahead-of-Time Class Loading & Linking

Authors Ioi Lam, Dan Heidinga, & John Rose

Owner Ioi Lam

Type Feature

Scope JDK

Status Candidate

Component hotspot / runtime

Discussion [leyden dash dev at openjdk dot org](#)

Reviewed by Alex Buckley, Brian Goetz, Mark Reinhold, Vladimir Kozlov

Created 2023/09/06 04:07

Updated 2024/08/22 03:52

Issue [8315737](#)

Summary

Improve startup time by making the classes of an application instantly available, in a loaded and linked state, when the HotSpot Java Virtual Machine starts. Achieve this by monitoring the application during one run and storing the loaded and linked forms of all classes in a cache for use in subsequent runs. Lay a foundation for future improvements to both startup and warmup time.

Goals



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP draft: Null-Restricted and Nullable Types (Preview)

Owner	Dan Smith
Type	Feature
Scope	SE
Status	Draft
Component	tools / javac
Discussion	valhalla dash dev at openjdk dot org
Effort	L
Duration	M
Created	2023/02/23 01:23
Updated	2024/08/20 20:19
Issue	8303099

Summary

Support *nullness markers* on Java types to indicate that a type rejects or deliberately allows nulls. This is a [preview language feature](#).

Goals

- Enhance Java's reference types to let programmers express whether null references are expected as values of the type



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP draft: Stable Values (Preview)

Authors Per Minborg, Maurizio Cimadamore

Type Feature

Scope SE

Status Draft

Component core-libs / java.lang

Effort S

Duration S

Created 2023/07/24 15:11

Updated 2024/08/06 13:19

Issue [8312611](#)

Summary

Introduce a *Stable Values API*, which provides performant immutable value holders where elements are initialized *at most once*. Stable Values offer the performance and safety benefits of final fields, while offering greater flexibility as to the timing of initialization. This is a [preview API](#).

Goals

- Provide an easy and intuitive API to describe value holders that can change at most once



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jtreg harness

Groups

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

JEP draft: Support HTTP/3 in the HttpClient

<i>Owner</i>	Daniel Fuchs
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Submitted
<i>Component</i>	core-libs / java.net
<i>Effort</i>	L
<i>Duration</i>	L
<i>Relates to</i>	JEP 321: HTTP Client API
<i>Reviewed by</i>	Alan Bateman, Bradford Wetmore, Paul Sandoz
<i>Created</i>	2022/08/05 14:58
<i>Updated</i>	2024/09/06 16:30
<i>Issue</i>	8291976

Summary

Update the [HTTP Client API](#) to support the [HTTP/3](#) protocol. This will allow applications and libraries to interact with HTTP/3 servers and get the benefits of HTTP/3 with minimal code changes.

Goals

- Update the HttpClient API to send and receive HTTP/3 requests and



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

Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

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

JEP draft: PEM API (Preview)

<i>Owner</i>	Anthony Scarpino
<i>Type</i>	Feature
<i>Scope</i>	SE
<i>Status</i>	Submitted
<i>Component</i>	security-libs / java.security
<i>Discussion</i>	security dash dev at openjdk dot org
<i>Effort</i>	M
<i>Duration</i>	M
<i>Reviewed by</i>	Alan Bateman, Sean Mullan
<i>Created</i>	2023/01/23 18:28
<i>Updated</i>	2024/09/05 03:51
<i>Issue</i>	8300911

Summary

Introduce an API for encoding and decoding the [Privacy-Enhanced Mail](#) (PEM) format. The PEM format is used for storing and sending cryptographic keys, certificates, and certificate revocations lists. This is a [preview API](#).

Goals

- Ease of use - Define a concise API that converts between PEM data and



Installing
Contributing
Sponsoring
Developers' Guide
Vulnerabilities
JDK GA/EA Builds
Mailing lists
Wiki · IRC

Bylaws · Census
Legal

Workshop

JEP Process

Source code

Mercurial
GitHub

Tools

Git
jreg harness

Groups

(overview)

Adoption

Build

Client Libraries

Compatibility &

Specification

Review

Compiler

Conformance

Core Libraries

JEP draft: Ahead of Time Compilation for the Java Virtual Machine

Author Julian Waters

Type Feature

Scope JDK

Status Draft

Component hotspot / compiler

Created 2023/07/28 02:38

Updated 2024/06/27 08:37

Issue [8313278](#)

Summary

Enhance the Java Virtual Machine with the ability to load Java applications and libraries compiled to native code for faster startup and baseline execution.

Goals

Enable the Java Virtual Machine to load Java code compiled Ahead of Time to native code, if it was compiled by a compiler from a matching Java Virtual Machine. Allow C1, C2, and JVMCI Compilers to compile native code Ahead of Time for a Java Virtual Machine to use. "Profiling" code is code compiled by C1 in tiers 2 and 3 for



Resources

Resources

- German article by me:
<https://www.innoq.com/de/articles/2024/03/jdk-zukunft-features/>
- What Happened to Java's String Templates?
<https://nipafx.dev/inside-java-newscast-71/>
- Brainf*ck Compiler with Class-File API:
<https://jameshamilton.eu/programming/build-compiler-jep-457-class-file-api>
- Presentation of Gatherers:
<https://www.youtube.com/watch?v=8fMFa6OqIY8>

Thanks! Questions?



Michael Vitz

Mail michael.vitz@innoq.com

X [@michaelvitz](https://twitter.com/michaelvitz)

Mastodon [@michaelvitz@innoq.social](https://mastodon.social/@michaelvitz)

LinkedIn [michaelvitz](https://www.linkedin.com/in/michaelvitz)



<https://www.innoq.com/de/talks/2024/09/java-forum-nord-2024-neues-vom-jdk/>

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Wendenstraße 130
20537 Hamburg

Spichernstraße 44
50672 Köln

Königstorgraben 11
90402 Nürnberg