Java SE Platform JSRs

Iris Clark

Specification Lead
iris.clark@oracle.com
June 11, 2024

2024 Maintenance Reviews

JSR 392 MR 1 (Java SE 17)

- Spec: `java.awt.Robot` (8330603), KEM API (8330545), `java.version.maintenance.version` (8330418)
- RI: 17.0.0.1 (https://github.com/openjdk/jdk17u-ri)
- TCK: JCK 17a

JSR 384 MR 3 (Java SE 11)

- Spec: `java.awt.Robot` (8331036)
- RI: 11.0.0.2 (https://github.com/openjdk/jdk11u-ri)
- TCK: JCK 11a (re-use)

JSR 337 MR 6 (Java SE 8)

- Spec: `java.awt.Robot` (8331038)
- RI: 8u44 (https://github.com/openjdk/jdk8u-ri)
- TCK: JCK 8d (re-use)

Schedule

2024/04/16
Proposal e-mail

2024/05/16 – 2024/06/13
Maintenance Review

2024/06/18 – 2024/06/24
Maintenance Review Ballot

2024/07
Maintenance Release

2024/10
OpenJDK & Oracle JDK releases

JSR 398: Java SE 23

Specification
- Latest: https://cr.openjdk.org/~iris/se/23/latestSpec (soon!)
- Public Review to begin July

Reference Implementation (RI) – JDK 23
- Latest: https://jdk.java.net/23 (build 26)
- Repository: https://github.com/openjdk/jdk/tree/jdk23
- Rampdown Phase 1 (RDP1)
  - Feature set frozen
  - Development limited to selected bug fixes
- 9 Integrated SE JEPs; 88 approved SE CSRs
- General Availability (GA): 2024/09/17

Technology Compatibility Took Kit (TCK) – JCK 23
- Code freeze rapidly approaching; Stabilization fork soon thereafter

Schedule

2023/12
  Expert Group Formation

2024/07 – 2024/08
  Public Review

2024/08
  Public Review –
  Final Approval Ballot

2024/09
  Final Release

SE JEPs in Java SE 23

Language

455	Primitive Types in Patterns, `instanceof`, & `switch` (Preview)
476	Module Import Declarations (Preview)
482	Flexible Constructor Bodies (Second Preview)
477	Implicitly Declared Classes & Instance `main` Methods (Third Preview)

Libraries

466	Class-File API (Second Preview)
473	Stream Gatherers (Second Preview)
480	Structured Concurrency (Third Preview)
481	Scoped Values (Third Preview)

Tools

467	Markdown Documentation Comments

Preview Features

- Preview features are fully specified, fully implemented, but subject to change.
- Code using a preview feature may not necessarily compile or run in another release.
- Must be enabled at compile time and run time:

```
javac --release 23 --enable-preview Main.java

java --enable-preview Main
java --source 23 --enable-preview Main.java // source code launcher
jshell --enable-preview
```

- All preview features in the current release must take one of the following actions in the next feature release: remove, re-preview, standardize
- The “History” section immediately after the “Summary” section describes the feature’s evolution

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

Enhance pattern matching to support primitive types in contexts previously restricted to reference and some integer types.

```
if (i instanceof byte b) {
                             // no loss
    ... b ...
}
```

Why

- Eliminate restrictions on use of primitive type patterns to make the Java language more consistent and more expressive across types
- Eliminates code which may be lossy when narrowing between types

JEP 476: Module Import Declarations (Preview)

Provides a simple means to import all public types in all packages of a module.

```
import module java.base;


public class A {
    public static void main(String... args) {
        List<Path> l = new ArrayList<>[];
        System.out.println(l.getClass().getName());
    }
}
```

Why

- Eliminate the need for multiple import-on-demand declarations when using diverse parts of an API exported by a module
- Simplify the re-use of modular libraries without requiring code to be in a module itself

JEP 482: Flexible Constuctor Bodies (Second Preview)

In Java language constructors, allow selected statements that do not reference the instance being created to appear before invoking super(…) or this(…).

```
Class A { ... A(int i) { ... throw new IllegalArgumentException(); ... } }
Class B extends A {
    int save;
    B(int i) {
        if (i < 0) throw new IllegalArgumentException("Bad number: " + i);
        save = i;              // x belongs to “this”
        super(i);   }}
```

History

- First previewed in Java SE 22; New title and significant changes in Java SE 22

Why

- Allows argument validation, computation, and field initialization before delegation to another constructor

JEP 477: Implicitly Declared Classes & Instance `main` Methods (Third Preview)

Reduce syntactic complexity of simple programs for novice users.

```
void main() {
    println("Hello, World!");
}
```

History

- First previewed in Java SE 21; New title and significant changes in Java SE 22; Two additions for automatic imports for implicitly declared classes

Why

- “Hello, World” exposes too many concepts that may intimidate beginning programmers
- Reduce ceremony for simple programs such as scripts and command-line utilities

JEP 466: Class-File API (Second Preview)

A standard API for parsing, generating, and transforming Java class files in package `java.lang.classfile` and subpackages. Tree traversal and streaming are supported.

- Reading - `ClassModel`
- Writing - `ClassBuilder and MethodBuilder`
- Transforming - `ClassTransform`

History

- Previewed in Java SE 22; Updated in Java SE 23

Why

- Enables faster evolution of Java class file format defined by the Java Virtual Machine Specification to provide support for new Java language features
- Eventually replace the JDK's internal copy of the third-party ASM library

Presentation

- *A Classfile API for the* JDK, Brian Goetz, JVM Language Summit 2023 (video)

JEP 473: Stream Gatherers (Second Preview)

Enhances the Stream API with support for custom intermediate operations through the `java.util.stream.Gatherer` interface. The `java.util.stream.Gatherers` class provides methods supporting windowing, scanning, and folding.

```
// window of fixed size
List<List<Integer>> windowsOfThree
    = Stream.of(0,1,2,3,4,5,6,7)
            .gather(Gatherers.windowFixed(3))
            .toList();
// windowsOfThree = [[0, 1, 2], [3, 4, 5], [6, 7]]
```

History

- First previewed in Java SE 22; Unchanged in Java SE 23

Why

- Fixed set of existing intermediate operations makes some complex tasks difficult
- Set of potential, useful intermediate operations is large

JEP 480: Structured Concurrency (Third Preview)

Introduce APIs to structure a task as a family of concurrent subtasks, and to coordinate them as a unit.

```
Callable<String> task1 = ...
Callable<Integer> task2 = ...
try (var scope = new StructuredTaskScope<Object>()) {
    Subtask<String>  subtask1 = scope.fork(task1);
    Subtask<Integer> subtask2 = scope.fork(task2);
    scope.join();
    ... process results/exceptions ...
} // close
```

History

- Incubated in Java SE 19 and Java SE 20; Previewed in Java SE 21; Unchanged in Java SE 22 and Java SE 23

Why

- Provide structure for large numbers of virtual threads
- Streamline error handling, improving reliability and enhancing observability

JEP 481: Scoped Values (Third Preview)

Introduce scoped values, which enable safe and efficient sharing of immutable data within and across threads.

```
final static ScopedValue<...> NAME = ScopedValue.newInstance();

// In some method
ScopedValue.runWhere(NAME, “duke”, () -> { ... NAME.get() ... call methods ... });

// In a method called directly or indirectly from the lambda expression
... NAME.get() ...
```

History

- Incubated in Java SE 20; Previewed in Java SE 21; Unchanged in Java SE 22; One concern addresses in Java SE 23

Why

- Alternative to thread-local variables and method arguments for sharing data across components

JEP 467: Markdown Documentation Comments

Enable Markdown syntax in JavaDoc documentation.

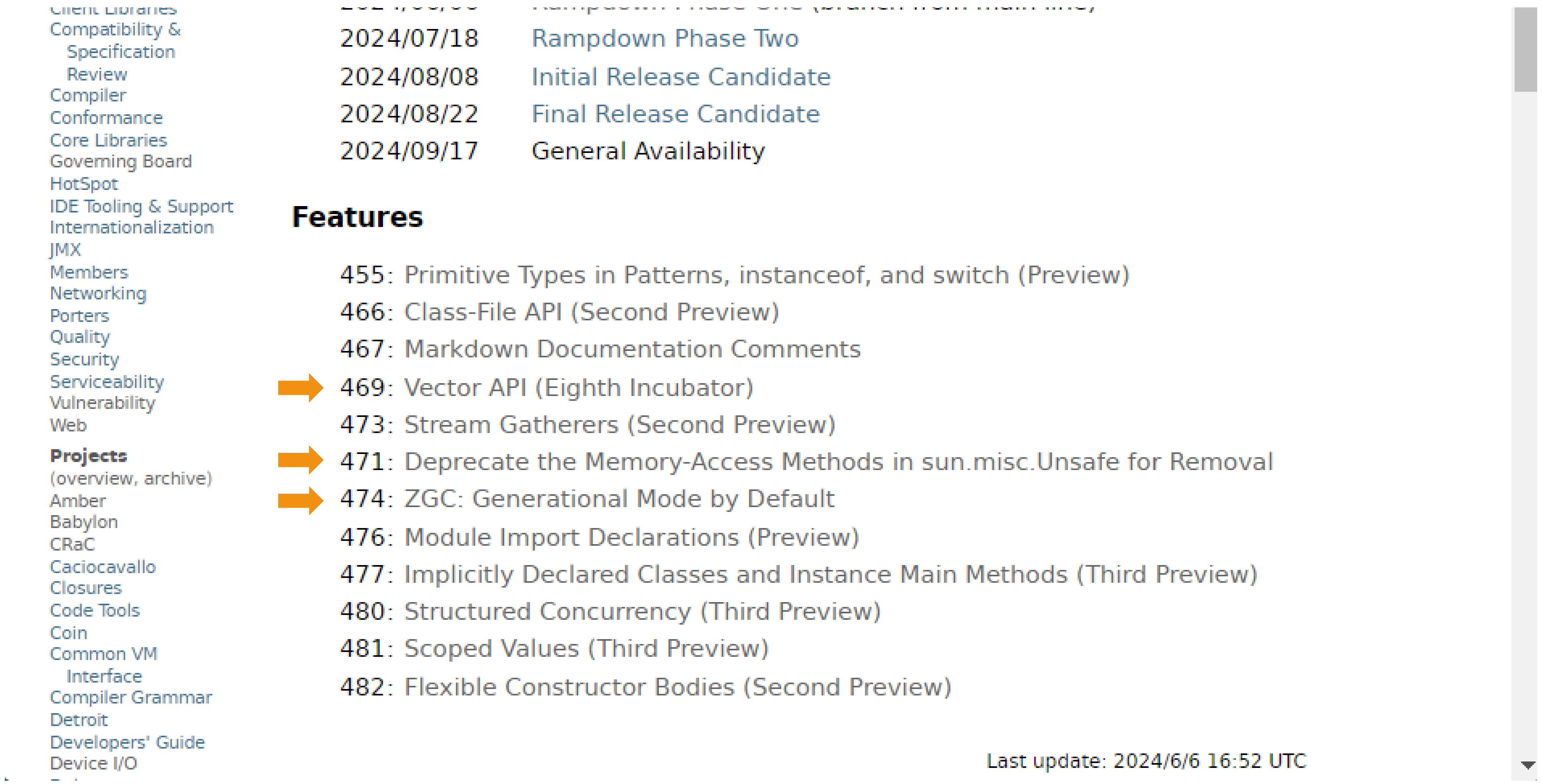
```
/// Returns `true` if, and only if, [#length()] is `0`.
///
/// @return `true` if [#length()] is `0`, otherwise
/// `false`
///
/// @since 1.6

public Boolean isEmpty()
```

Why

- Modernize JavaDoc syntax to support a simple, popular markup language
- API documentation comments are easier to write and read in source form

openjdk.org/projects/jdk/23



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
Caciocavallo
Closures
Code Tools
Coin
Common VM Interface
Compiler Grammar
Detroit
Developers' Guide
Device I/O

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)

Last update: 2024/6/6 16:52 UTC

Other notable changes in Java SE 23

88 Compatibility & Specification Review (CSR) Requests
https://bugs.openjdk.org/issues/?filter=44960

2 JSR Maintenance Releases
199: Java Compiler API [MR 7]
269: Pluggable Annotations
Processing API [MR 17]

4 Removed APIs, 1 Removed Feature
`javax.management.loading.MLet` (20)
`javax.management.loading.MLetContent` (20)
`javax.management.loading.PrivateMLet` (20)
`javax.management.loading.MLetMBean` (20)

JMX Subject Delegation (21)

7 Terminally Deprecated APIs Added
`java.beans.beancontext` package (23)
`java.io.ObjectOutputStream.PutField.write` (1.4)
`java.net.MulticastSocket.getTTL` (1.2.2)
`java.net.MulticastSocket.setTTL` (1.2.2)
`java.net.MulticastSocket.send` (1.4)
`java.net.Socket(InetAddress, int. Boolean)` (1.1)
`java.net.Socket(String, int, Boolean)` (1.1)

JEP 459: String Templates (Second Preview)

REMOVED

Project Amber

Introduce string composition that couples literal text with embedded expressions and template processors.

```
String name = "Duke";
String info = STR."My name is \{name}";
assert info.equals("My name is Duke");   // true
```

History

- First previewed in Java SE 21; Practically unchanged in Java SE 22; Removed in Java SE 23

Why

- Commonly used feature in other popular programming languages
- Existing string composition techniques (String concatenation with '`+`', `StringBuilder`, `Formatter.format()`) are verbose
- String composition that achieves the clarity of string interpolation without the inherent hazards (e.g. SQL injection attacks)

Upcoming Ballots

June

- JSR 337 MR 6: Java SE 8
- JSR 384 MR 3: Java SE 11
- JSR 392 MR 1: Java SE 17

July

- JSR 199 MR 7: Java Compiler API
- JSR 269 MR 17: Pluggable Annotation Processing API

August

- JSR 398: Java SE 23

JSR 397: Java SE 23
Thank You!

Resources

- https://openjdk.org/projects/jdk/23/spec/
  - https://jcp.org/en/jsr/detail?id=398
  - JEPs: https://bugs.openjdk.org/secure/Dashboard.jspa?selectPageId=22205
  - CSRs: https://bugs.openjdk.org/secure/Dashboard.jspa?selectPageId=22204
  - https://mail.openjdk.org/mailman/listinfo/java-se-spec-experts
  - https://jdk.java.net/23/
- https://openjdk.org/projects/jdk/24/spec/
- https://mail.openjdk.org
- https://github.com/openjdk