ORACLE

Java™

JEP 500:
Prepare to Make Final Mean Final

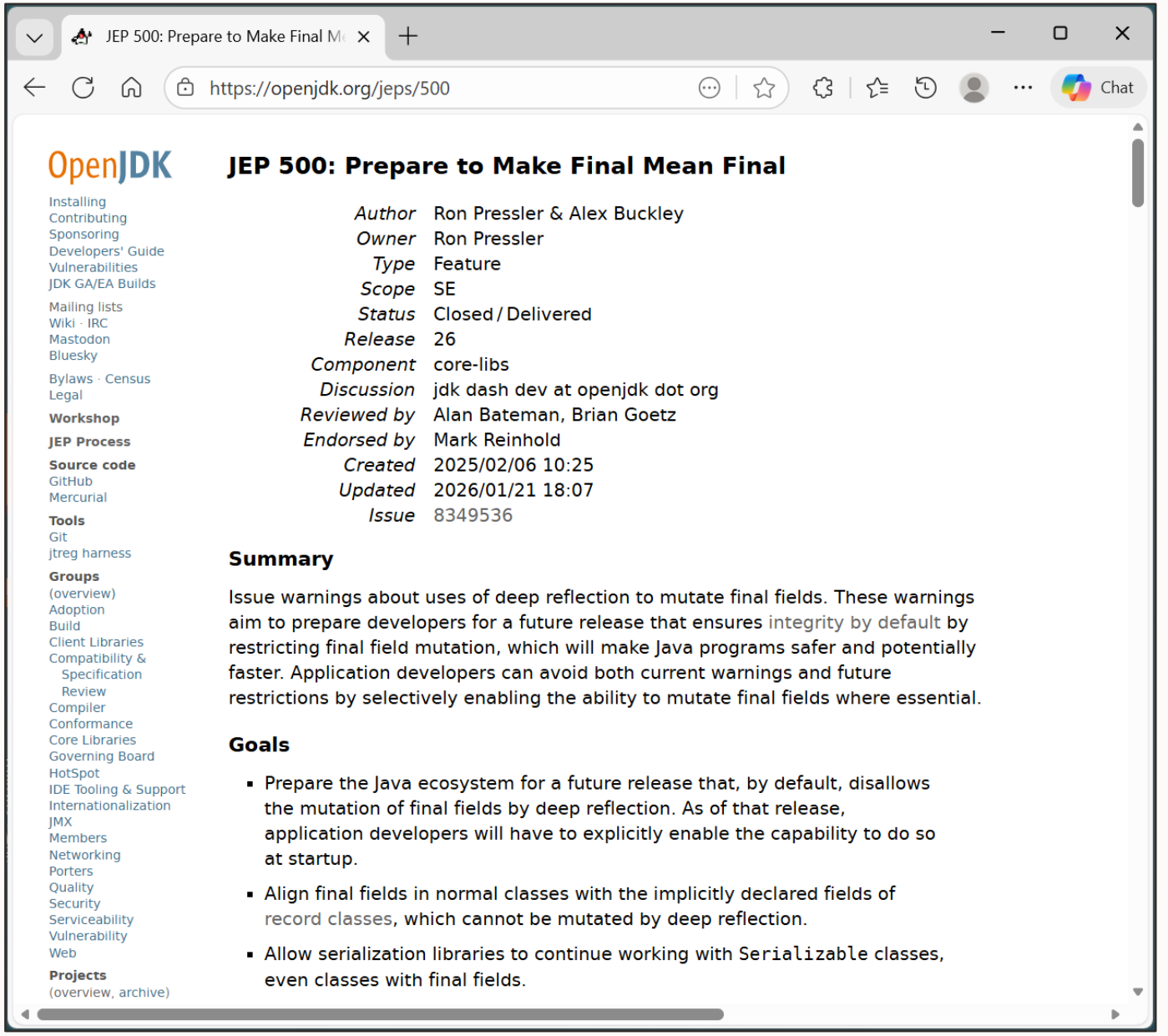
Alex Buckley

Java Platform Group

February 2026

At a high level

- Delivered in JDK 26
- A change to the Java SE Platform, not just the JDK
- Critical for *Integrity by Default*
- For compatibility reasons, the first of three JEPs about final-field mutation



https://openjdk.org/jeps/500

1. The Problem

What do you mean, “Make Final Mean Final” ?

- Final fields represent immutable state.
- The expectation that a final field cannot be reassigned in far-flung parts of the program is crucial for correctness and performance.
- Unfortunately, the expectation that a final field cannot be reassigned is **false**.
- Deep reflection strikes again!
- “Final Does Not Mean Final”
- Prevents JIT optimization for trusted fields within this run of the JVM, and across runs (Leyden)

```
// A normal class with a final field
class C {
    final int x;
    C() { x = 100; }
}

// 1. Perform deep reflection over the final field in C
java.lang.reflect.Field f = C.class.getDeclaredField("x");
f.setAccessible(true);      // Make C's final field mutable

// 2. Create an instance of C
C obj = new C();
System.out.println(obj.x);  // Prints 100

// 3. Mutate the final field in the object
f.set(obj, 200);
System.out.println(obj.x);  // Prints 200
f.set(obj, 300);
System.out.println(obj.x);  // Prints 300
```

The culprit: Serialization

It might seem absurd that the Java Platform includes an API that undermines the meaning of the `final` keyword. Final fields have played a crucial role in the Java Memory Model since JDK 5; in particular, their immutability underpins the safe initialization of objects in multi-threaded code. Unfortunately, their immutability also conflicts with the operation of serialization libraries that mutate fields to initialize objects during deserialization. This use case was sufficiently important to justify changing the reflection API in JDK 5 so that it could be used to mutate final fields.

In retrospect, offering such unconstrained functionality was a poor choice because it sacrificed integrity. When we introduced hidden classes in JDK 15 and record classes in JDK 16, we constrained deep reflection to disallow the mutation of final fields in hidden and record classes. We constrained deep reflection further when we strongly encapsulated JDK internals in JDK 17. In JDK 24, we started the process to remove methods in `sun.misc.Unsafe` that, like deep reflection, allow the mutation of final fields.

2. The Solution

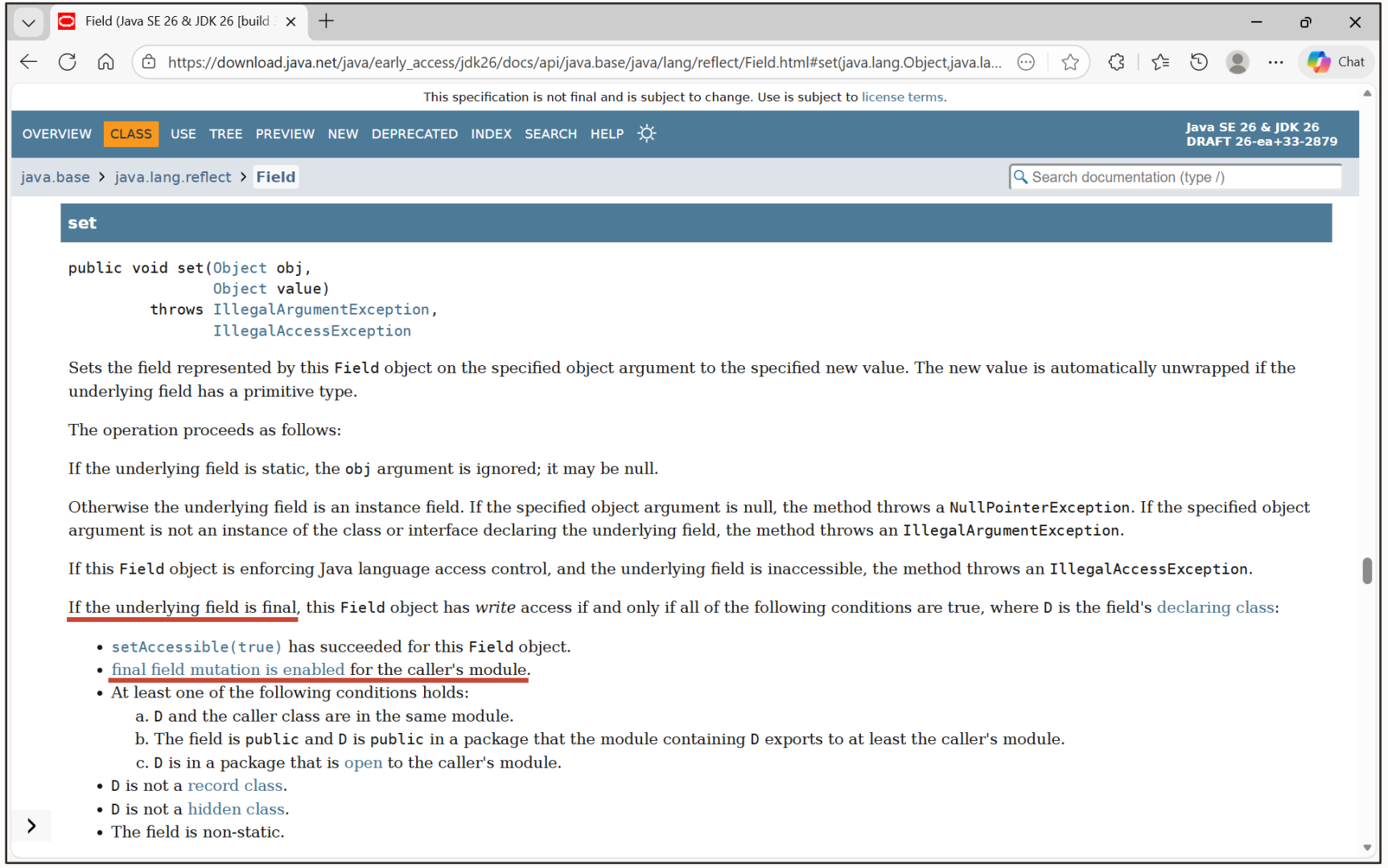
JEP 500: Prepare to Make Final Mean Final

Description

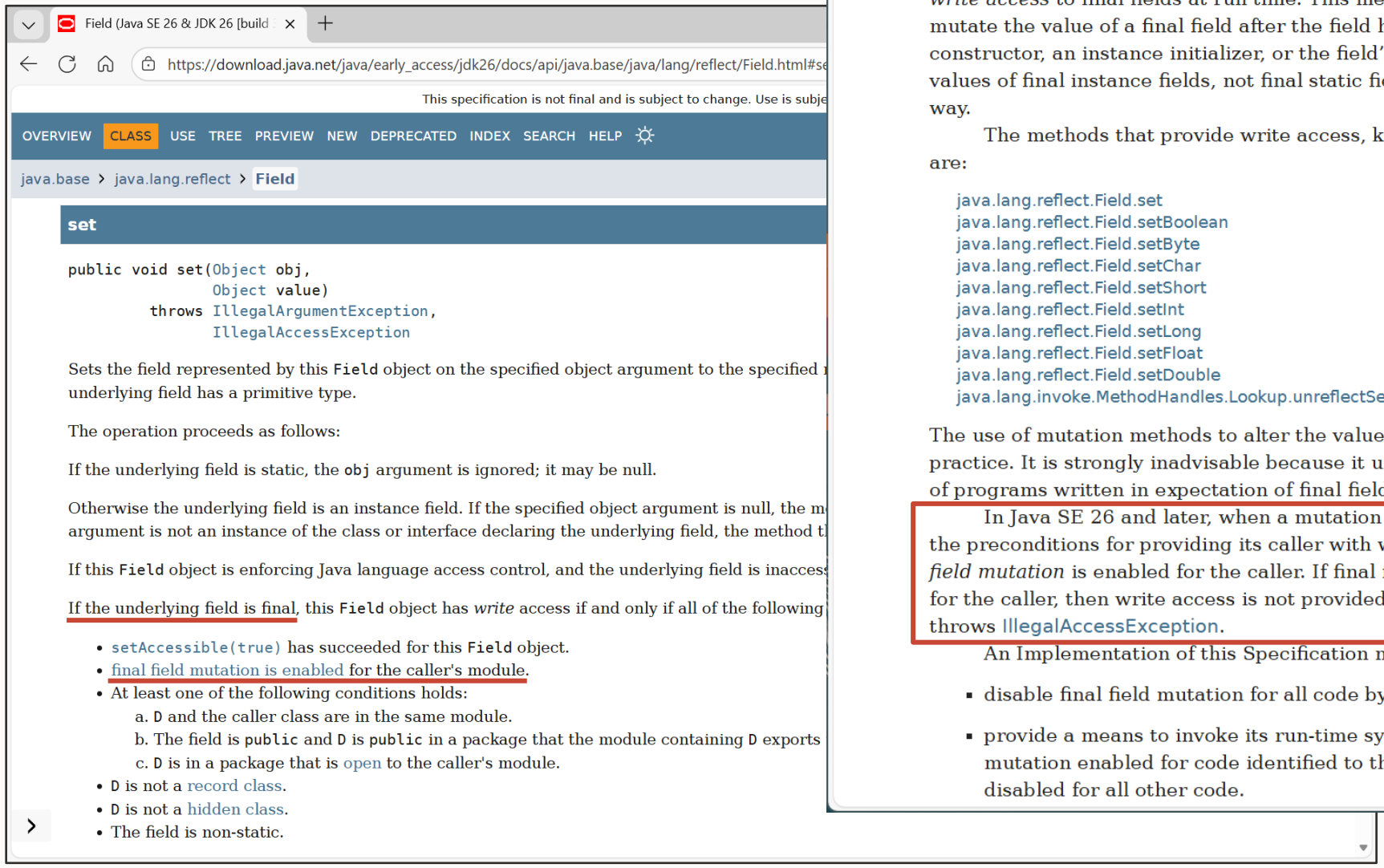
In JDK 5 and later releases, you can mutate final fields via deep reflection, i.e., the `setAccessible` and `set` methods of `java.lang.reflect.Field`. In JDK 26, we will restrict deep reflection so that, by default, mutating a final field causes a warning to be issued at run time. It will not be possible to avoid the warning simply by using `--add-opens` to enable the deep reflection of classes with final fields.

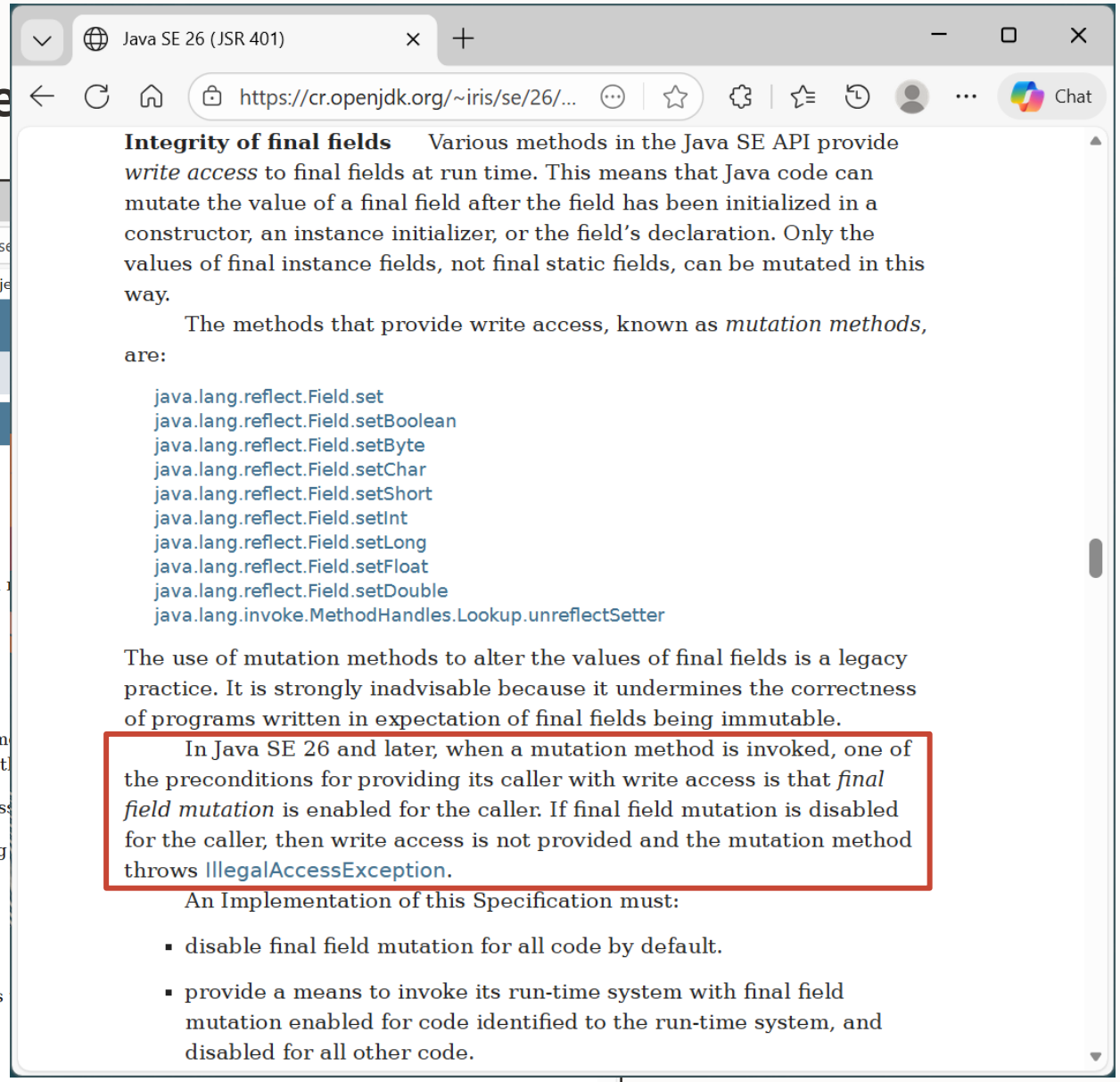
We refer to restrictions on mutating final fields as *final field restrictions*. We will strengthen the effect of final field restrictions over time. Rather than issue warnings, a future JDK release will, by default, throw exceptions when Java code uses deep reflection to mutate final fields. This will enable the Java Platform and applications running on it to have integrity by default. The warning we introduce here is intended to prepare developers for that future.

`java.lang.reflect.Field::set` requires *final field mutation* to be *enabled*



java.lang.reflect.Field::set require





Demo: Enable final field mutation

```
C:\Users\Alex Buckley>java -version
openjdk version "26-ea" 2026-03-17
OpenJDK Runtime Environment (build 26-ea+33-2879)
OpenJDK 64-Bit Server VM (build 26-ea+33-2879, mixed mode, sharing)

C:\Users\Alex Buckley>java Test
100
WARNING: Final field x in class Test$C has been mutated reflectively by class Test in unnamed module @4e0e2f2a
 (file:/C:/Users/Alex%20Buckley/)
WARNING: Use --enable-final-field-mutation=ALL-UNNAMED to avoid a warning
WARNING: Mutating final fields will be blocked in a future release unless final field mutation is enabled
200
300

C:\Users\Alex Buckley>java --enable-final-field-mutation=ALL-UNNAMED Test
100
200
300
```


3. The Impact

Recommendation for serialization libraries: `sun.reflect.ReflectionFactory`

When final field restrictions are strengthened in a future JDK release, serialization libraries will no longer be able to use deep reflection out-of-the-box. Rather than ask users to enable final field mutation on the command line, maintainers of serialization libraries should serialize and deserialize objects using the `sun.reflect.ReflectionFactory` API, which is supported for this purpose. This API allows a serialization library to obtain a method handle to special code that initializes an object by assigning to its instance fields directly, including final fields. This code, which is dynamically generated by the JDK, gives the serialization library the same powers as the JDK's own serialization facilities; it is not necessary to enable final field mutation by the module of the serialization library.

Recommendation for other libraries: Avoid mutating final fields altogether

> Some libraries and frameworks for dependency injection, unit testing, and mocking use deep reflection to manipulate objects, including by mutating final fields. The maintainers of such components should only ask users to enable final field mutation as a last resort. Instead, maintainers should find architectural approaches that avoid the need to mutate final fields or access private fields altogether. For example, most dependency injection frameworks now forbid the injection of final fields, and all discourage it, instead recommending constructor injection.

4. The User Experience

At a high level: Final-field mutation follows the playbook for *Integrity by Default*

> Most of the unsafe APIs in the Java Platform were in use for years before they were deemed unsafe, so it is not practical to restrict them without notice: applications would fail unexpectedly. In addition, configuring the Java runtime to allow the use of unsafe APIs by libraries, frameworks, and tools is not part of the traditional developer experience. To alert application developers to the need to configure the Java runtime, we restrict the use of a pre-existing unsafe API in a gradual fashion:

https://openjdk.org/jeps/8305968#Adapting-to-restrictions-on-unsafe-APIs

Java	Feature	Short-term default	Long-term policy
9	JEP 261 Module System	`--illegal-access=permit (warn)`	`--add-opens` `--add-exports`
24	JEP 472 Prepare to Restrict JNI	`--illegal-native-access=warn`	`--enable-native-access`
26	JEP 500 Prepare to Make Final Mean Final	`--illegal-final-field-mutation=warn`	`--enable-final-field-mutation`

JDK 26 defaults to `--illegal-final-field-mutation=warn`

```
C:\Users\Alex Buckley>java Test
100
WARNING: Final field x in class Test$C has been mutated reflectively by class Test in unnamed module @4e0e2f2a
 (file:/C:/Users/Alex%20Buckley/)
WARNING: Use --enable-final-field-mutation=ALL-UNNAMED to avoid a warning
WARNING: Mutating final fields will be blocked in a future release unless final field mutation is enabled
200
300
```

```
C:\Users\Alex Buckley>java --illegal-final-field-mutation=warn Test
100
WARNING: Final field x in class Test$C has been mutated reflectively by class Test in unnamed module @4e0e2f2a
 (file:/C:/Users/Alex%20Buckley/)
WARNING: Use --enable-final-field-mutation=ALL-UNNAMED to avoid a warning
WARNING: Mutating final fields will be blocked in a future release unless final field mutation is enabled
200
300
```


--illegal-final-field-mutation=debug

```
C:\Users\Alex Buckley>java --illegal-final-field-mutation=debug Test
100
Final field x in class Test$C has been mutated reflectively by class Test in unnamed module @c387f44 (file:/C:/Users/Alex%20Buckley/)
        at java.base/java.lang.reflect.Field.postSetFinal(Field.java:1534)
        at java.base/java.lang.reflect.Field.setFinal(Field.java:1449)
        at java.base/java.lang.reflect.Field.set(Field.java:909)
        at Test.main(Test.java:19)
200
Final field x in class Test$C has been mutated reflectively by class Test in unnamed module @c387f44 (file:/C:/Users/Alex%20Buckley/)
        at java.base/java.lang.reflect.Field.postSetFinal(Field.java:1534)
        at java.base/java.lang.reflect.Field.setFinal(Field.java:1449)
        at java.base/java.lang.reflect.Field.set(Field.java:909)
        at Test.main(Test.java:21)
300
```

--illegal-final-field-mutation=deny

```
C:\Users\Alex Buckley>java --illegal-final-field-mutation=deny Test
100
java.lang.IllegalAccessException: class Test (in unnamed module @4e0e2f2a) cannot set final field Test$C.x (in unnamed module @4e0e2f2a), unnamed module @4e0e2f2a is not allowed to mutate final fields
        at java.base/java.lang.reflect.Field.preSetFinal(Field.java:1504)
        at java.base/java.lang.reflect.Field.setFinal(Field.java:1447)
        at java.base/java.lang.reflect.Field.set(Field.java:909)
        at Test.main(Test.java:19)
```

--illegal-final-field-mutation=allow

```
C:\Users\Alex Buckley>java --illegal-final-field-mutation=allow Test
100
200
300
```

A Change for *Integrity by Default* Occurs in Three Phases

Strong Encapsulation

Phase	Java	Feature	Short-term default
1	9	JEP 261 Module System	`--illegal-access=permit (warn)`
2	16	JEP 396 Strongly Encapsulate JDK Internals by Default	`--illegal-access=deny`
3	17	JEP 403 Strongly Encapsulate JDK Internals	`--illegal-access is ignored`

Final Field Restrictions

Phase	Java	Feature	Short-term default
1	26	JEP 500 Prepare to Make Final Mean Final	`--illegal-final-field-mutation=warn`
2	YY	JEP YYY Make Final Mean Final by Default	`--illegal-final-field-mutation=deny`
3	ZZ	JEP ZZZ Make Final Mean Final	`--illegal-final-field-mutation is ignored`

A Change for *Integrity by Default*: Phase 1 of 3 (Java 26 / JEP 500)

`--illegal-final-field-mutation=allow` `--illegal-final-field-mutation=warn` `--illegal-final-field-mutation=debug` `--illegal-final-field-mutation=deny`

DEFAULT

A Change for *Integrity by Default*: Phase 2 of 3 (Java YY / JEP YYY)

`--illegal-final-field-mutation=warn`

`--illegal-final-field-mutation=debug`

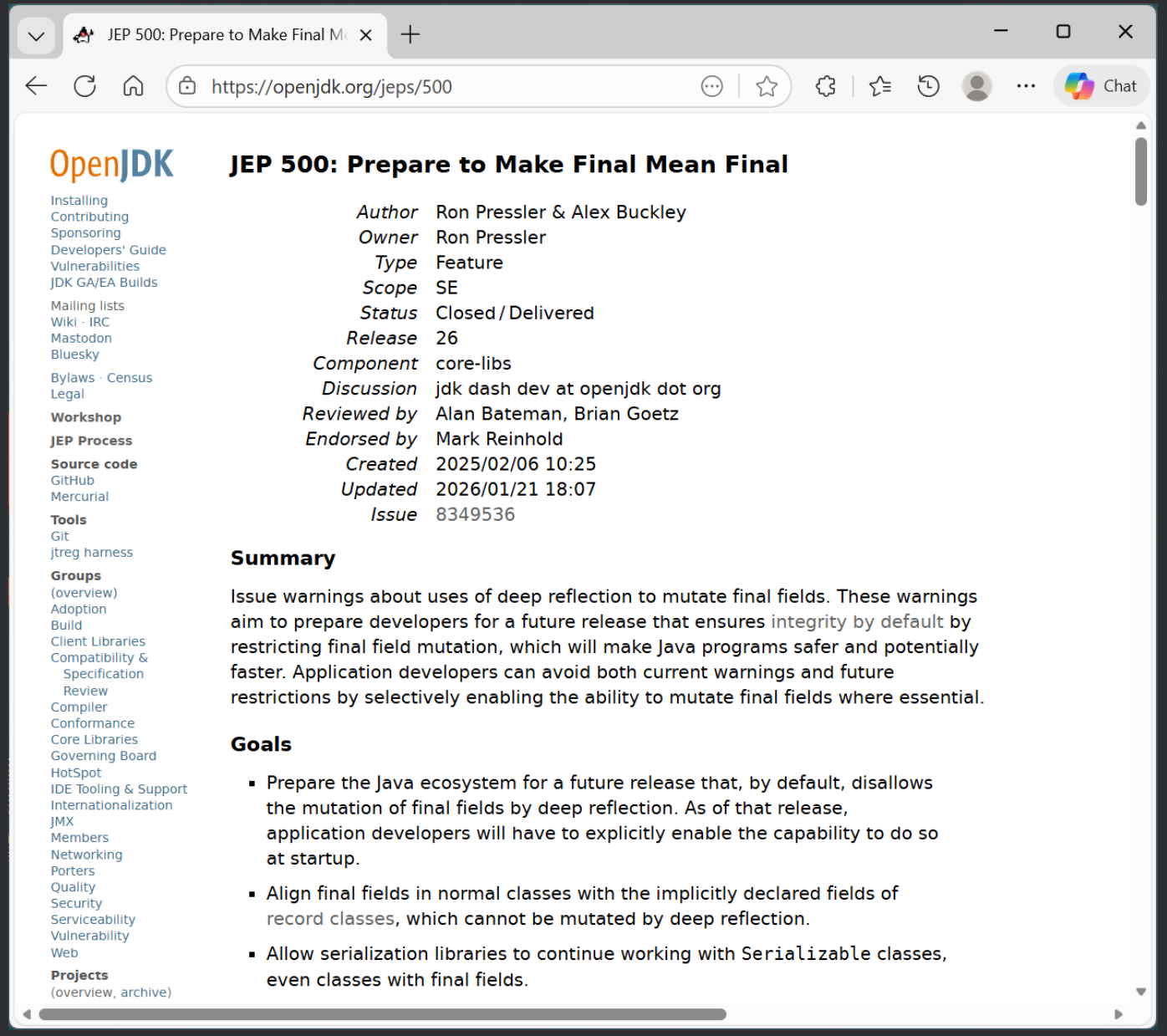
`--illegal-final-field-mutation=deny`

DEFAULT

A Change for *Integrity by Default*: Phase 3 of 3 (Java ZZ / JEP ZZZ)


```
--illegal-final-field-mutation=... is obsolete and ignored
```

Thank You



https://openjdk.org/jeps/500