



# Null Safety in Java with JSpecify and NullAway

---

Sébastien Deleuze

Spring Framework Core Committer

Tanzu Division, Broadcom

<https://seb.deleuze.fr>

# **Problem statement**



JD jspecify-nullway-de... baseline TokenExtractor.java Main > ⚡ ⚡ :

```
package org.example;

interface TokenExtractor {

    /**
     * Extract a token from a {@link String}.
     * @param input the input to process
     * @return the extracted token
     */
    String extractToken(String input);
}
```

# Root issue: nullness is implicit

```
package org.example;

interface TokenExtractor {

    /**
     * Extract a token from a {@link String}.
     * @param input the input to process
     * @return the extracted token
     */
    String extractToken(String input);
}
```



# Express nullness explicitly with JSpecify

# Mental model: nullness of type usages

- **Unspecified:** we don't know whether it can include null or not
- **Nullable:** it can include null
- **Non-null:** It will not include null



```
package org.example;

public class Main {

    public static void main(String[] args) {
        TokenExtractor extractor = new DefaultTokenExtractor();
        String token = extractor.extractToken(input: "...");
        System.out.println("The token has a length of " + token.length());
    }
}
```

# Why not just using Optional<T>?

# **Optional is not usable in a lot of use cases**

- Runtime overhead
- Breaks existing API signatures
- Not intended for use as parameters or fields
- Increase the complexity of your code and APIs

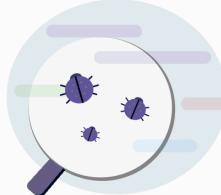
Why not another `@Nullable`  
annotation?

# JSpecify

Standard Annotations for Java Static Analysis

Learn More

```
@NullMarked  
public class JSpecify {  
    @Nullable  
    Integer numNPE() {  
        return null;  
    }  
}
```



## Standard Annotations

JSpecify is releasing the first artifact of tool-independent annotations for powering static analysis checks in your Java code.

## Next Level Static Analysis

JSpecify defines precise semantics, letting analysis tools find more bugs, and more consistently. Library owners won't have to decide which tool to support.



## Community Effort

JSpecify is developed by consensus of members representing a variety of stakeholders in Java static analysis, and we welcome your participation.

### Docs

[Start Here](#)

[User Guide](#)

### Contacts

[Public Group ↗](#)

[Mail The Team ↗](#)

### More

[GitHub ↗](#)

[Blog](#)

**Set the default to non-null  
to reduce the noise**

```
package org.example;
```

```
import ...
```

```
①     interface TokenExtractor {
```

/\*\*\*

\* Extract a token from a `{@link String}`.

\* *@param* *input* the input to process

\* `@return` the extracted token or `{@code null}` if not found

\* /

```
    @Nullable String extractToken(@NonNull String input);  
}
```

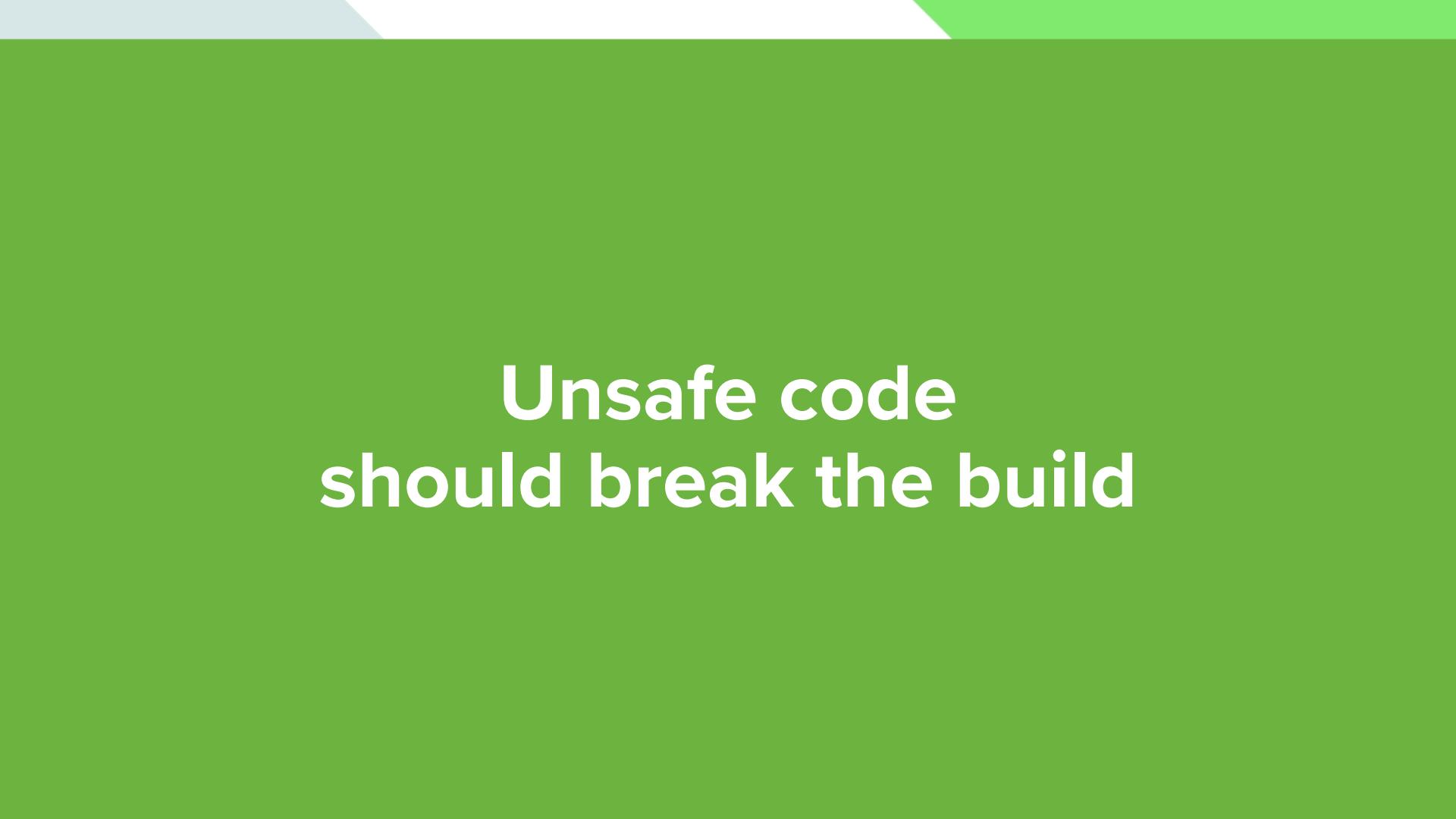
Be aware of `@Target(TYPE_USE)`  
annotations syntax

```
package org.example;

import org.jspecify.annotations.Nullable;

① interface TokenExtractor {

    /**
     * Extract a token from a {@link String}.
     * @param input the input to process
     * @return the extracted token or {@code null} if not found
     */
    @Nullable String extractToken(String input);
}
```



**Unsafe code  
should break the build**

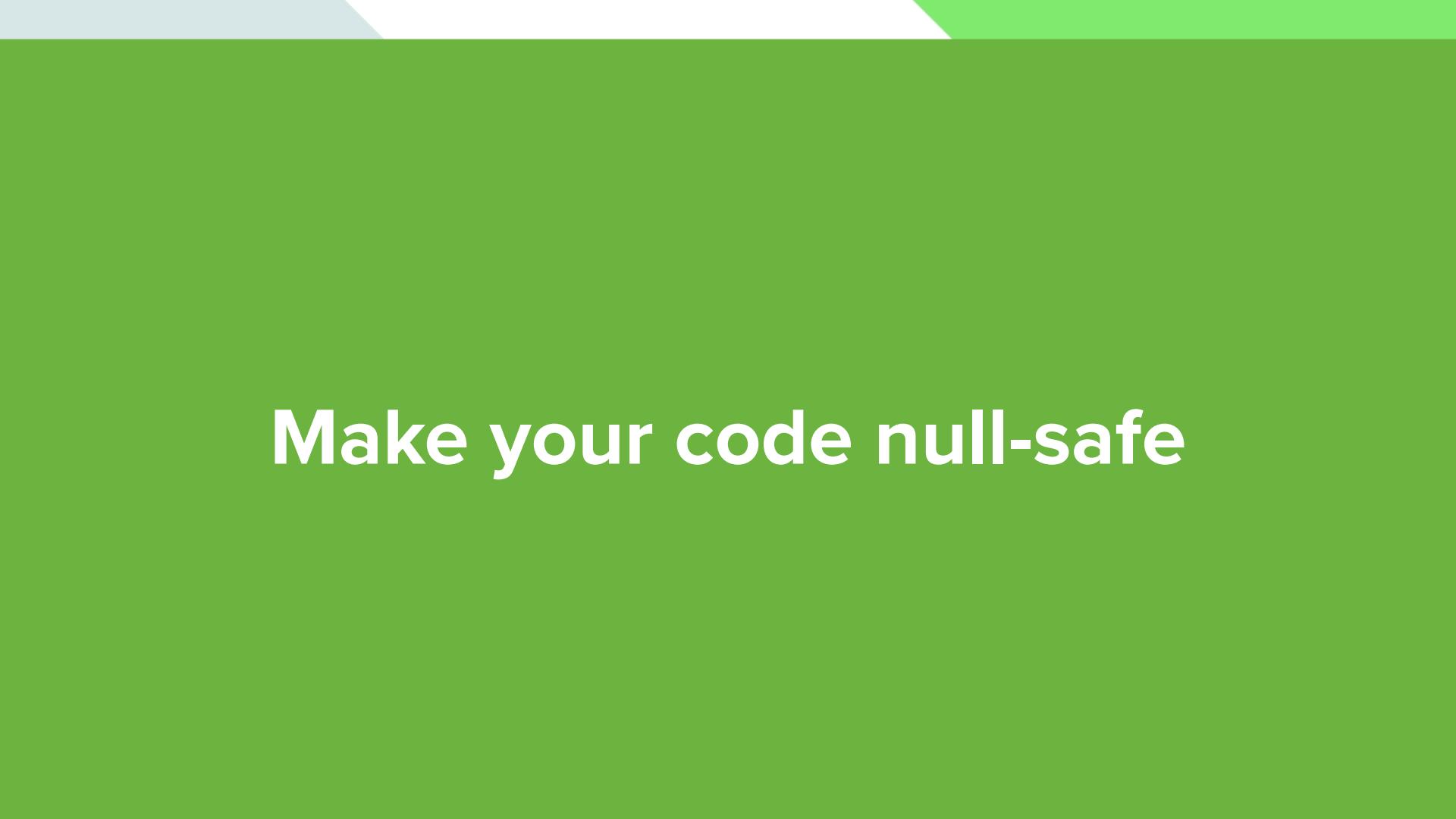
The screenshot shows a Java code editor interface. At the top, there are tabs for "jspecify-nullway-demo", "jspecify-step2", "Main.java", and "Main". On the far right of the toolbar, there is a search icon, a settings icon, and a warning icon with a yellow exclamation mark and the number "1". The main code area contains the following Java code:

```
package org.example;

public class Main {
    public static void main(String[] args) {
        TokenExtractor extractor = new DefaultTokenExtractor();
        String token = extractor.extractToken(input: "...");
        System.out.println("The token has a length of " + token.length());
    }
}
```

A yellow box highlights the call to `token.length()`. A small yellow warning icon with the number "1" is located in the bottom right corner of the code editor window.

The screenshot shows a terminal window with two tabs: "Terminal" and "Local". The "Terminal" tab is active, showing the command "jspecify-nullway-demo (jspecify-step2)\$" followed by a cursor. The background of the terminal window is light gray.



# Make your code null-safe

The screenshot shows a Java code editor interface. At the top, there are tabs for 'jspecify-nullway-demo' (selected), 'main', 'Main.java' (with a green circular icon), 'Main' (with a blue square icon), and several other icons for search, refresh, and settings. A yellow warning icon with the number '1' is located in the top right corner. The main area displays the following Java code:

```
package org.example;

public class Main {

    public static void main(String[] args) {
        TokenExtractor extractor = new DefaultTokenExtractor();
        String token = extractor.extractToken(input: "...");
        System.out.println("The token has a length of " + token.length());
    }
}
```

# Nullness of arrays (and varargs)

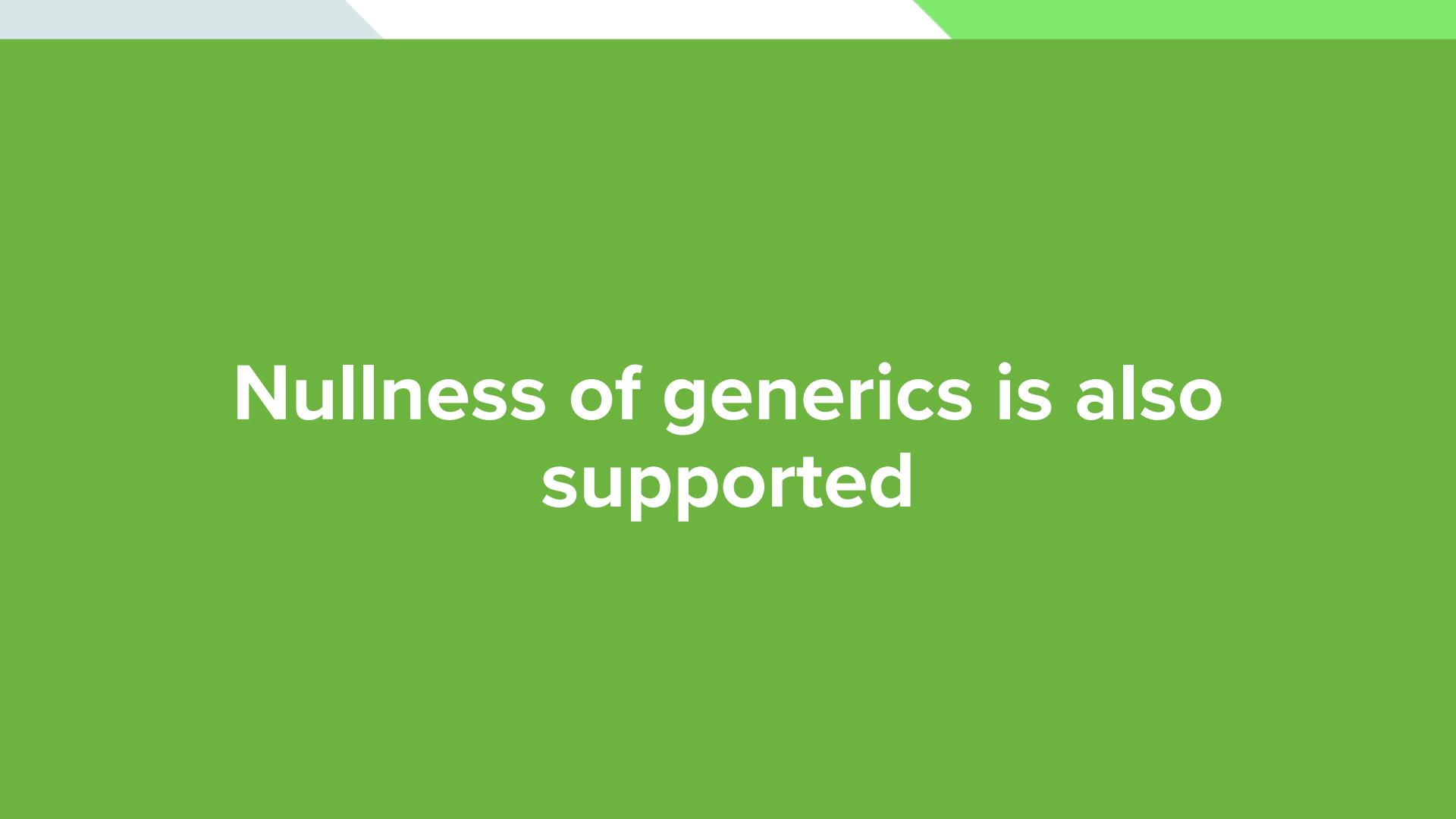
```
JD jspecify-nullway... handle-null build.gradle (jspe... Main > ⚡ : ⚡ 🔍 ⚡
    id 'net.ltgt.errorprone' version '4.1.0' // https://github.com/uber/gradle-errorprone-plugin
}

group = 'org.example'
version = '1.0-SNAPSHOT'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.jspecify:jspecify:1.0.0' // https://jspecify.org
    errorprone 'com.uber.nullaway:nullaway:0.12.6' // https://github.com/uber/NullAway
    errorprone 'com.google.errorprone:error_prone_core:2.37.0' // https://github.com/google/error-prone
}

tasks.withType(JavaCompile).configureEach { JavaCompile it ->
    options.errorprone {
        disableAllChecks = true // Other error prone checks are disabled
        option("NullAway:OnlyNullMarked", "true") // Enable nullness checks only in null-marked code
        error("NullAway") // bump checks from warnings (default) to errors
    }
}
```



**Nullness of generics is also supported**

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Bar:** JD jspecify-nullway-d... (selected), generics, TokenExtractor.java (selected), Main, and other icons.
- Project Tree (Left):** Shows the project structure:
  - jspecify-nullway-demo (~/workspace)
  - .gradle
  - .idea
  - .kotlin
  - build
  - gradle
  - src
    - main
      - java
        - org.example
  - resources
  - .gitignore
  - build.gradle
  - gradlew
  - gradlew.bat
  - jspecify-nullway-demo.iml
- Editor (Right):** Displays the code for `TokenExtractor.java`:

```
package org.example;

interface TokenExtractor {

    /**
     * Extract a token from a {@link String}.
     * @param input the input to process
     * @return the extracted token
     */
    Wrapper<String> extractToken(String input);
}
```

# Consume Java libraries from Kotlin

The screenshot shows a Java IDE interface with the following details:

- Project Bar:** JD jspecify-nullway-demo, kotlin-baseline, Main.kt, Main, and other icons.
- Project Tree (Left):** Shows the project structure under "jspecify-nullway-demo".
  - build** (highlighted)
  - .gradle
  - .idea
  - .kotlin
  - src
    - main
      - java
        - org.example
      - kotlin
        - org.example
    - resources
  - .gitignore
  - build.gradle
  - gradlew
  - gradlew.bat
  - jspecify-nullway-demo.iml
- Code Editor (Right):** Displays the following Kotlin code:

```
package org.example

fun main() {
    val extractor: TokenExtractor = DefaultTokenExtractor()
    val token: String! = extractor.extractToken("...")
    println("The token has a length of " + token.length)
}
```

# Null Safety in Spring applications

Sébastien Deleuze · You  
Spring @ Broadcom  
3mo •

...

I have just merged into Spring Framework main branch the huge commit (3458 files changed) that migrates the codebase to JSpecify annotations. That will allow Spring Framework 7 and related portfolio projects to provide next-level null-safety support to avoid NullPointerException at runtime.

See related documentation at [https://lnkd.in/d\\_uABV64](https://lnkd.in/d_uABV64).

```
@NullMarked
public class JSPECIFY {
    @Nullable
    Integer numNPE() {
        return null;
    }
}
```



Spencer Gibb and 418 others

39 comments · 24 reposts

#### Reactions

...

```
package org.example;

public class Main {

    public static void main(String[] args) {
        }
}
```

# Lazy initialization

Project

- > .gradle
- > .idea
- > .kotlin
- > build
- > gradle
- < src
  - < main
    - < java
      - < org.example
        - package-info.java
  - < resources
  - .gitignore
  - build.gradle
  - gradlew
  - gradlew.bat
  - jspecify-nullway-demo.iml
  - LICENSE
  - README.md
  - settings.gradle
- > External Libraries
- > Scratches and Consoles

Search Everywhere Double

Go to File ⌘O

Recent Files ⌘E

Navigation Bar ⌘↑

Drop files here to open them

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

<b>Author</b>	Per Minborg & Maurizio Cimadamore
<b>Owner</b>	Per-Ake Minborg
<b>Type</b>	Feature
<b>Scope</b>	SE
<b>Status</b>	Targeted
<b>Release</b>	25
<b>Component</b>	core-libs / java.lang
<b>Discussion</b>	core dash libs dash dev at openjdk dot org
<b>Effort</b>	S
<b>Duration</b>	S
<b>Reviewed by</b>	Alex Buckley, Brian Goetz
<b>Endorsed by</b>	Mark Reinhold
<b>Created</b>	2023/07/24 15:11
<b>Updated</b>	2025/04/01 14:52
<b>Issue</b>	<a href="#">8312611</a>



## Summary

Introduce an API for *stable values*, which are objects that hold immutable data. Stable values are treated as constants by the JVM, enabling the same performance optimizations that are enabled by declaring a field final. Compared to final fields, however, stable values offer greater flexibility as to the timing of their initialization. This is a [preview API](#).

Ongoing effort to provide  
null-safe APIs in Spring Boot 4.0

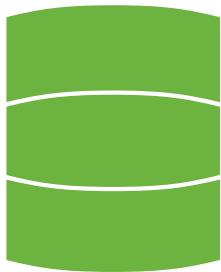
# Upcoming Null Safety portfolio-wide!



Spring  
Boot



Spring  
Framework



Spring  
Data



Spring  
Security



Spring  
AI



Upcoming recommendation to  
enable JSpecify based null-safety  
in Spring applications

**Let's turn "the billion dollar mistake"  
into a zero cost abstraction allowing to  
express the potential absence of value**

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



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](#)

**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
- Support conversions between types with different nullness properties, accompanied by warnings about possibly-mishandled null values
- Compatibly interoperate with traditional Java code that makes no assertions about the null compatibility of its types, and support gradual adoption of these new features without introducing source or binary incompatibilities

# Looking for contributors

uber / NullAway

Type  to search

Code Issues 119 Pull requests 3 Actions Projects Wiki Security Insights

Watch 71 Fork 303 Starred 3.7k

master 15 Branches 104 Tags

Go to file  + <> Code

 **dhruv-agr** and **msridhar** Allowing NewClassTree to be passed i... 56c7f2b · 2 weeks ago 936 Commits

 .buildscript Modify JMH Benchmark Workflow For Shellcheck (...) 2 years ago

 .github Update to Error Prone 2.38.0 (#1203) last month

 annotations Better @MonotonicNonNull support (#1149) 3 months ago

 buildSrc Test on JDK 24 (#1187) last month

 code-coverage-report External Library Models Integration (#922) last year

 config/hooks Switch to Spotless for formatting Java code (#780) 2 years ago

 gradle Update to Gradle 8.14 (#1213) 2 weeks ago

**About**

A tool to help eliminate NullPointerExceptions (NPEs) in your Java code with low build-time overhead

android java static-code-analysis  
static-analysis nullability  
nullability-analysis

Readme MIT license Code of conduct Activity Custom properties 3.7k stars

# Thank you



Copyright © 2005-2025 Broadcom, Inc. or its affiliates.

