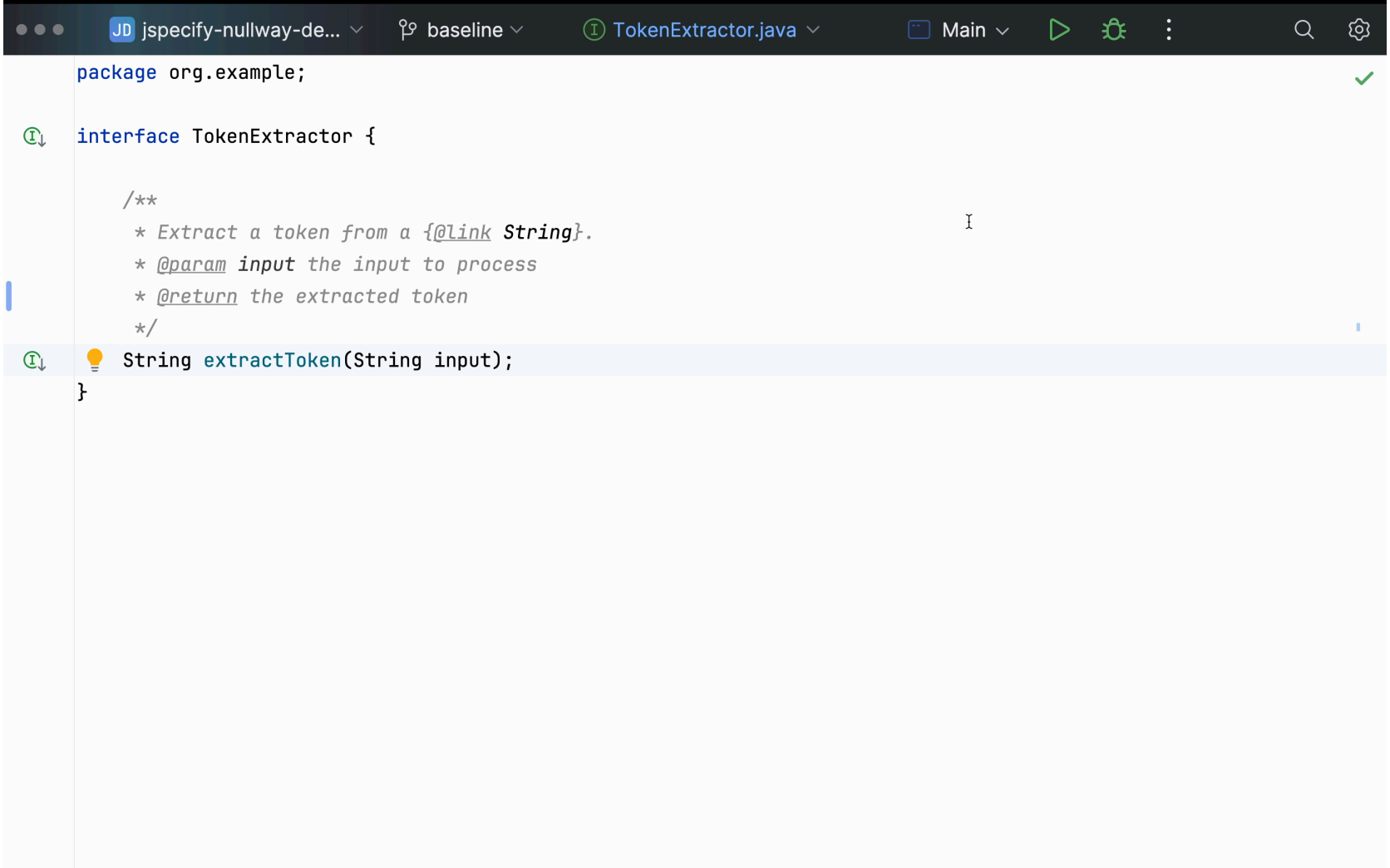
Null Safety in Java with JSpecify and NullAway

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);
}

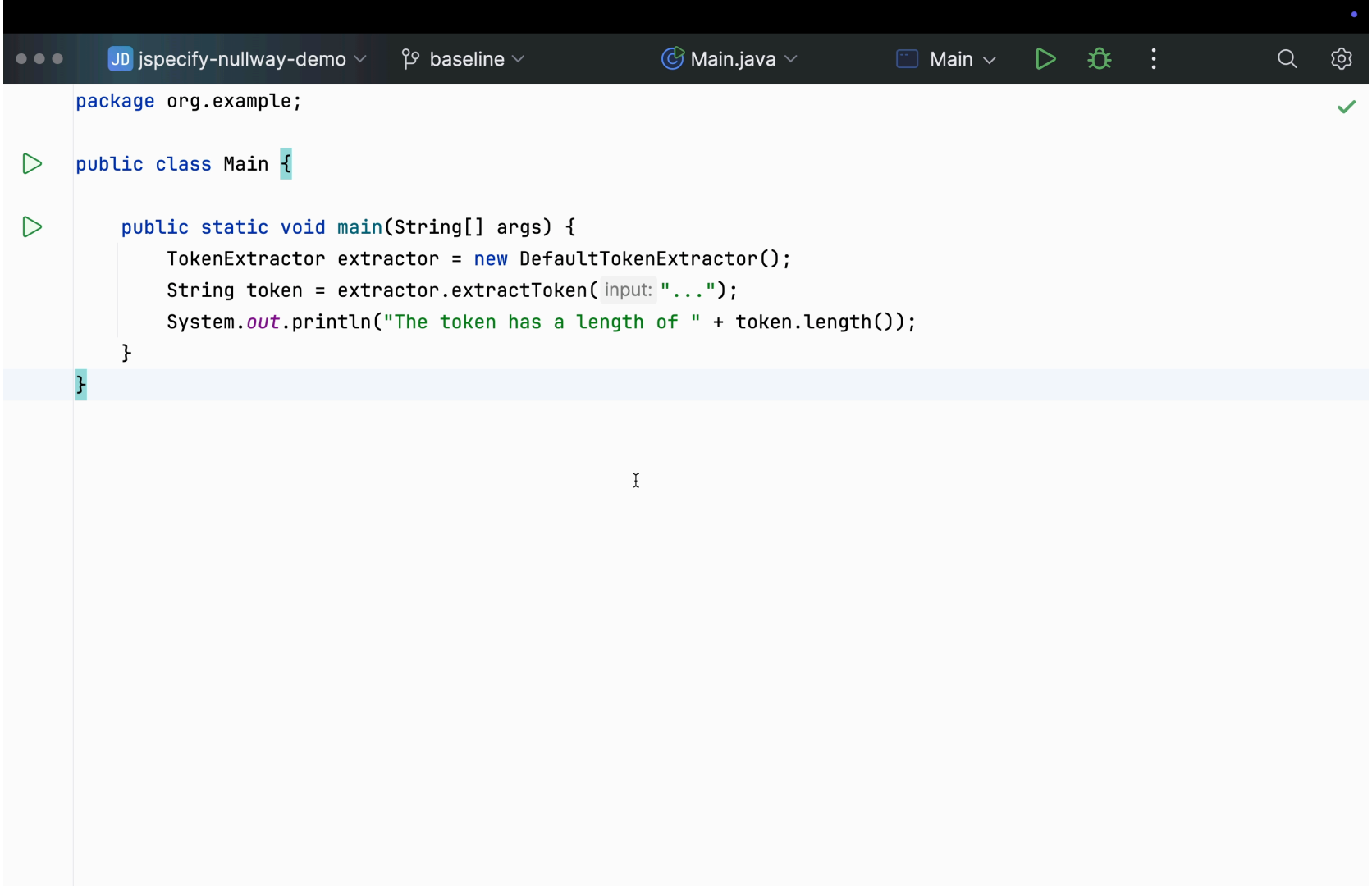
Root issue: nullness is implicit

```java
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


JD jspecify-nullway-demo
baseline
Main.java
Main
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

Standard Annotations

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

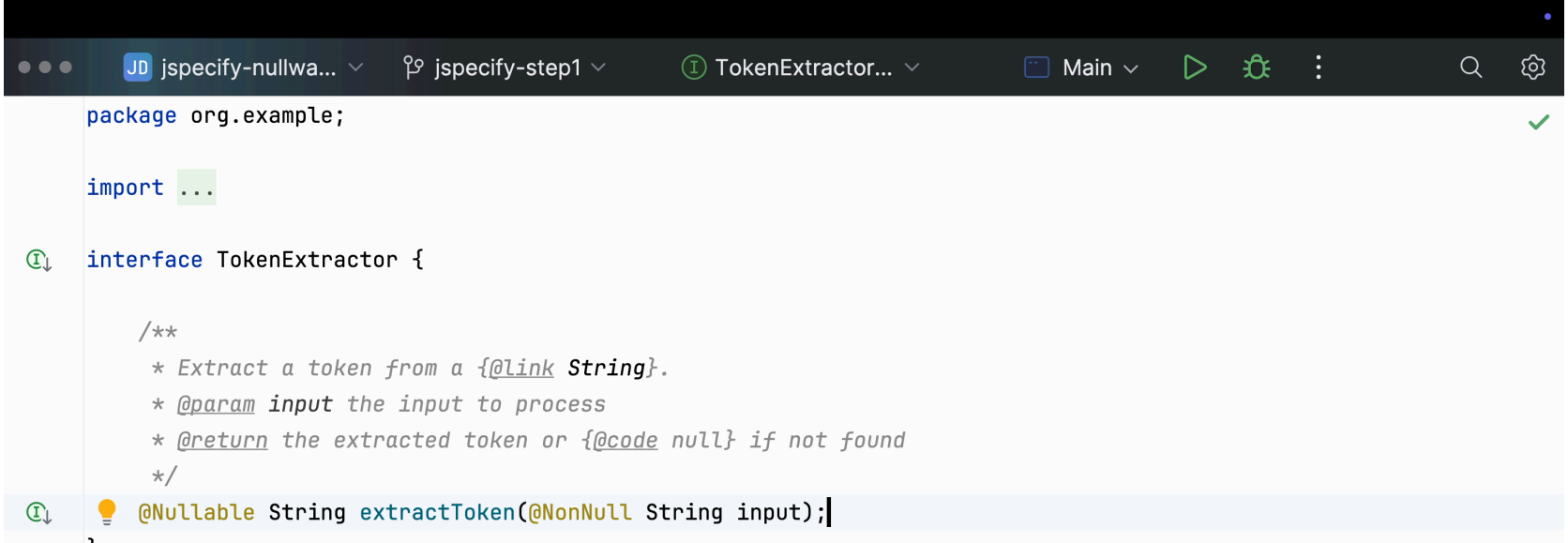
Next Level Static Analysis

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

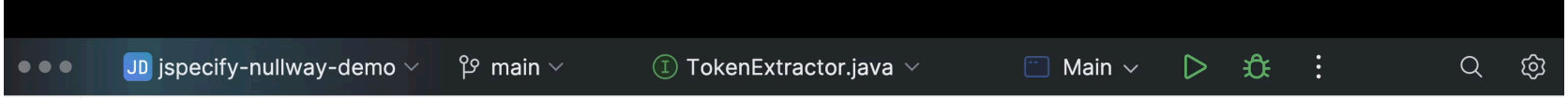
Community Effort

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

Set the default to non-null to reduce the noise

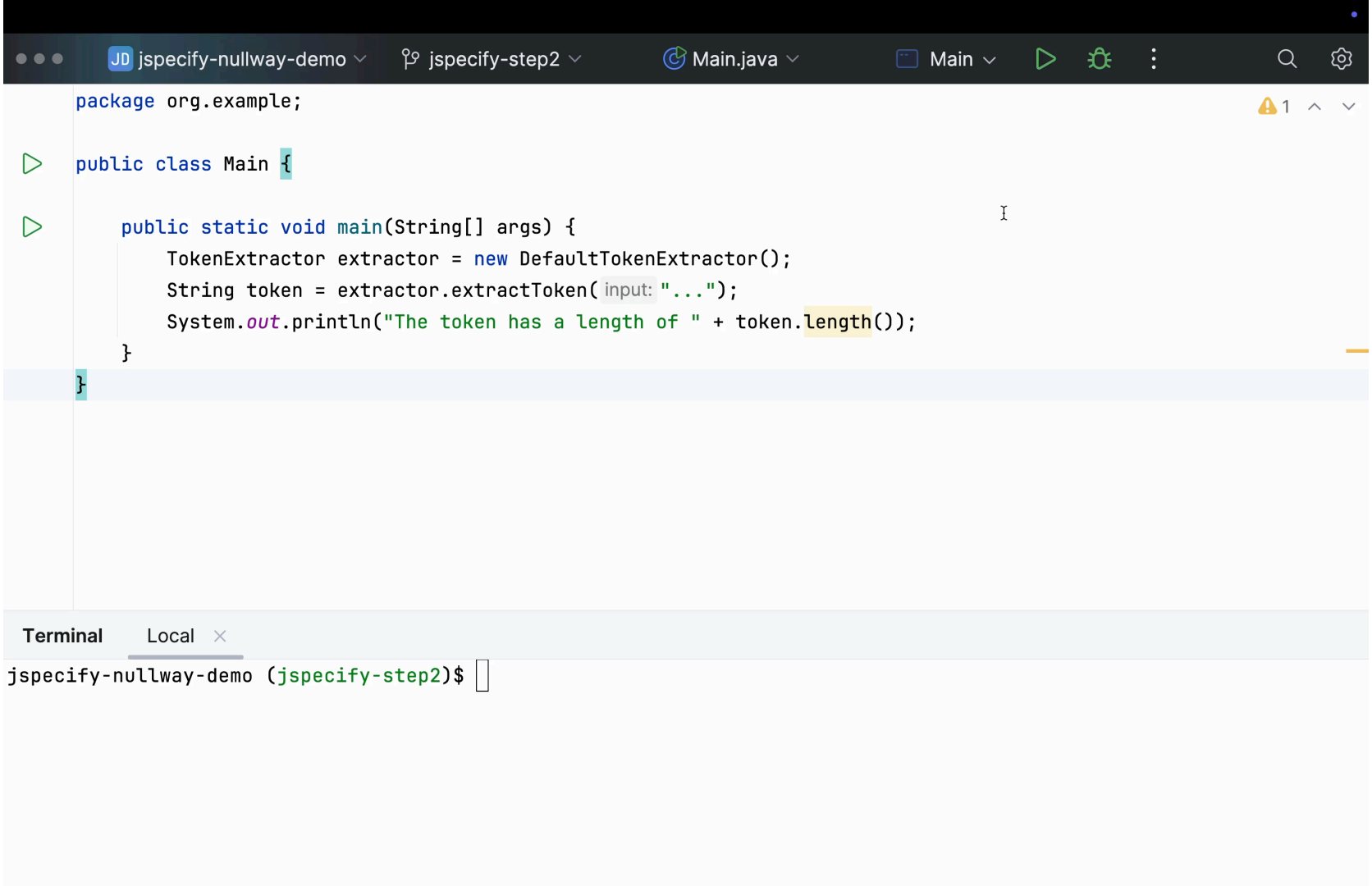

JD jspecify-nullwa...
jspecify-step1
TokenExtractor...
Main
package org.example;
import ...
interface TokenExtractor {
/**
* Extract a token from a {@link String}.
* @param input the input to process
* @return the extracted token or {@code null} if not found
*/
@Nullable String extractToken(@NonNull String input);
}

Be aware of @Target(TYPE_USE) annotations syntax




```java
package org.example;

import org.jspecify.annotations.Nullable;

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


jspecify-nullway-demo
jspecify-step2
Main.java
Main
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());
}
}
Terminal
Local
jspecify-nullway-demo (jspecify-step2)$

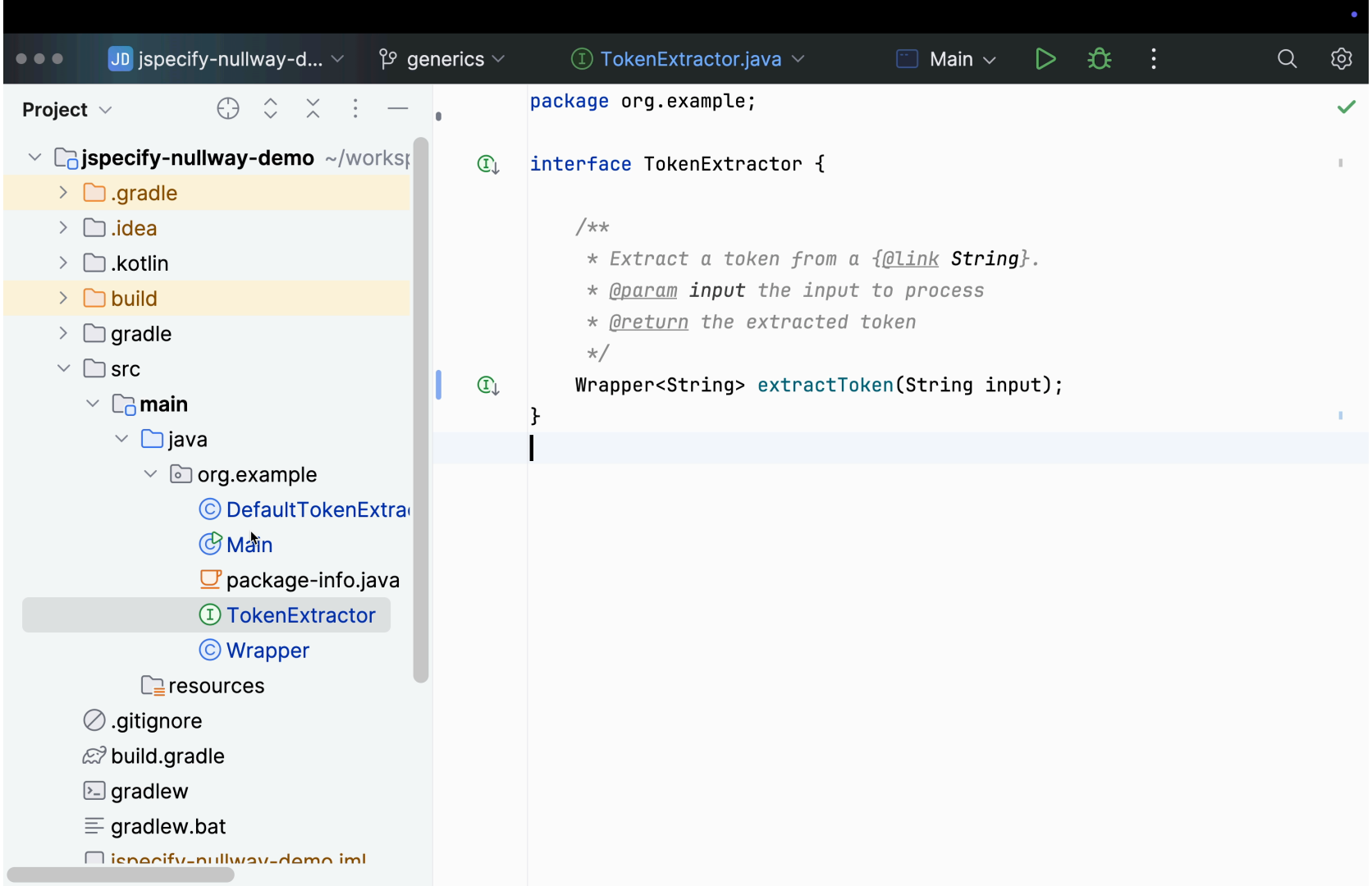
Make your code null-safe


```java
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());
    }
}
```

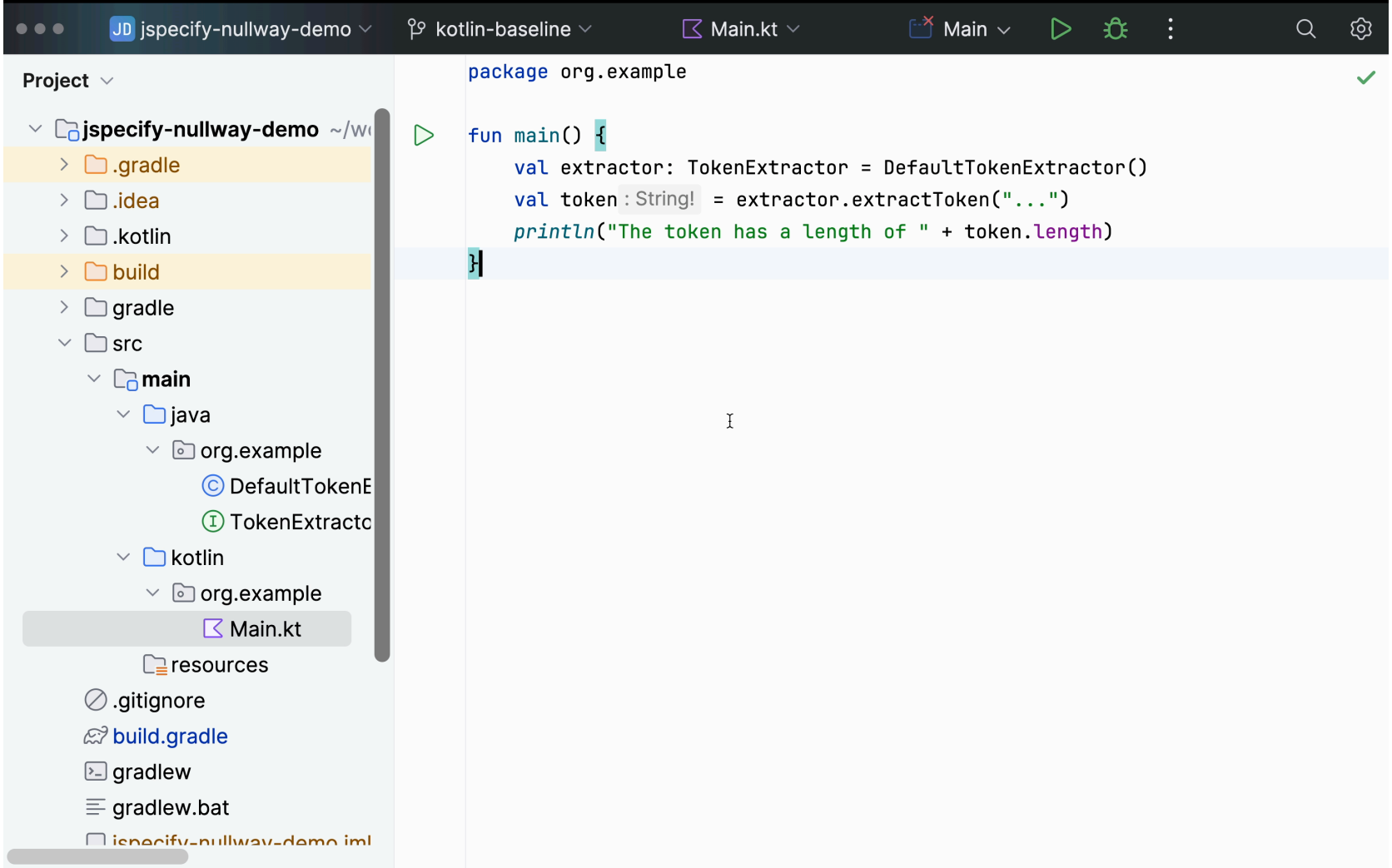
Nullness of arrays (and varargs)

```
id 'net.ltgt.errorprone' version '4.1.0' // https://github.com/tbroyer/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


jspecify-nullway-d...
generics
TokenExtractor.java
Main
Project
jspecify-nullway-demo ~/worksp
.gradle
.idea
.kotlin
build
gradle
src
main
java
org.example
DefaultTokenExtra
Main
package-info.java
TokenExtractor
Wrapper
resources
.gitignore
build.gradle
gradlew
gradlew.bat
jspecify-nullway-demo.iml
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

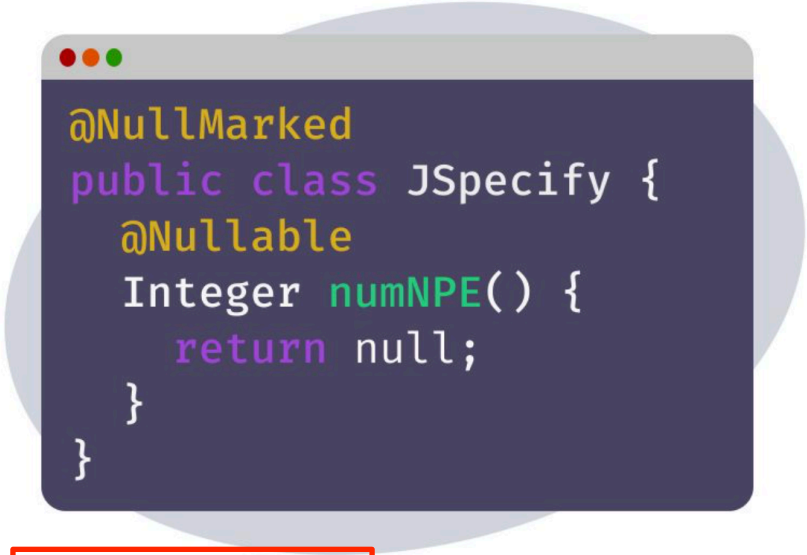

jspecify-nullway-demo
kotlin-baseline
Main.kt
Main
Project
jspecify-nullway-demo ~/w
.gradle
.idea
.kotlin
build
gradle
src
main
java
org.example
DefaultTokenE
TokenExtractc
kotlin
org.example
Main.kt
resources
.gitignore
build.gradle
gradlew
gradlew.bat
jspecify-nullway-demo.iml
package org.example
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.

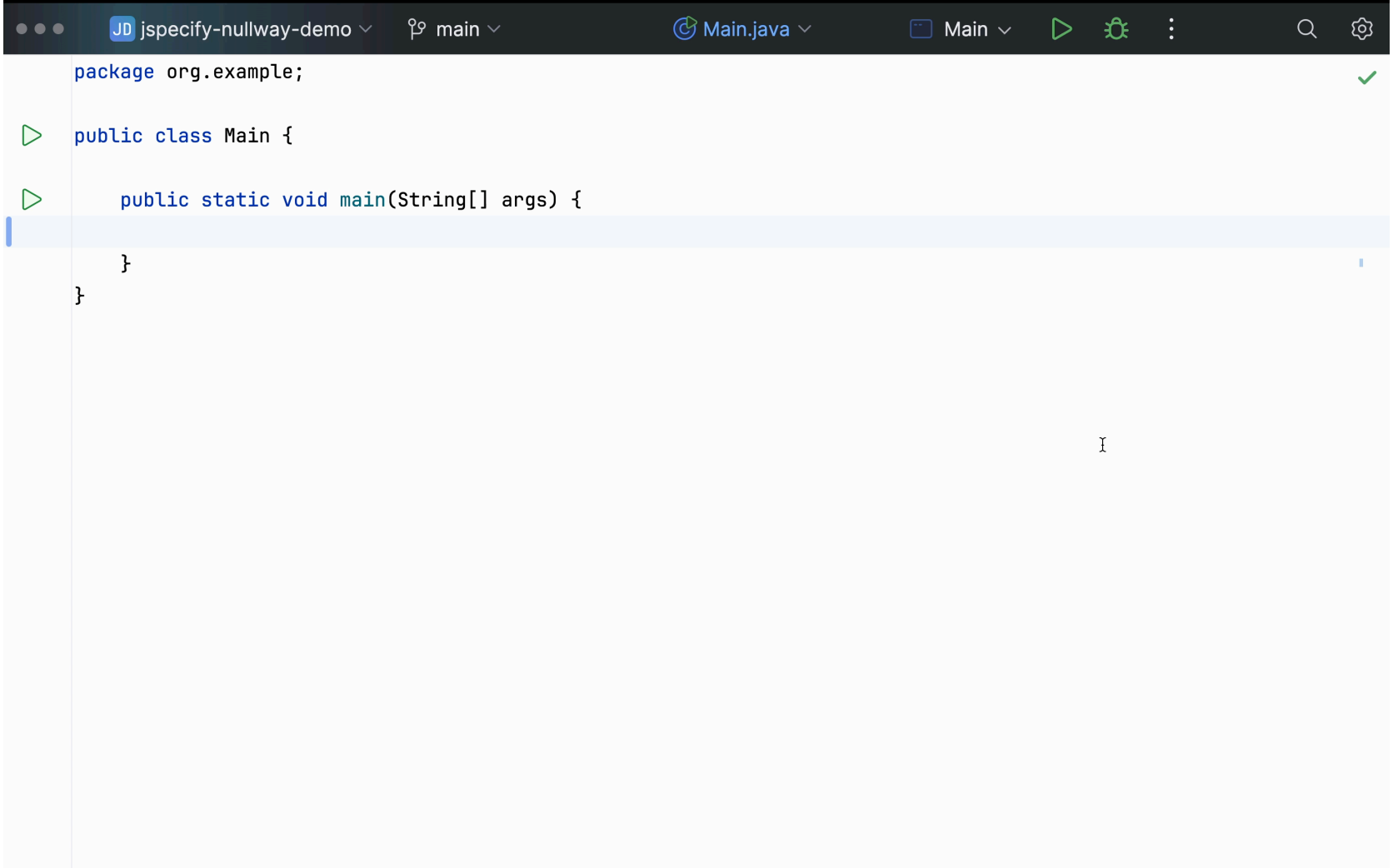
See related documentation at https://lnkd.in/d_uABV64.



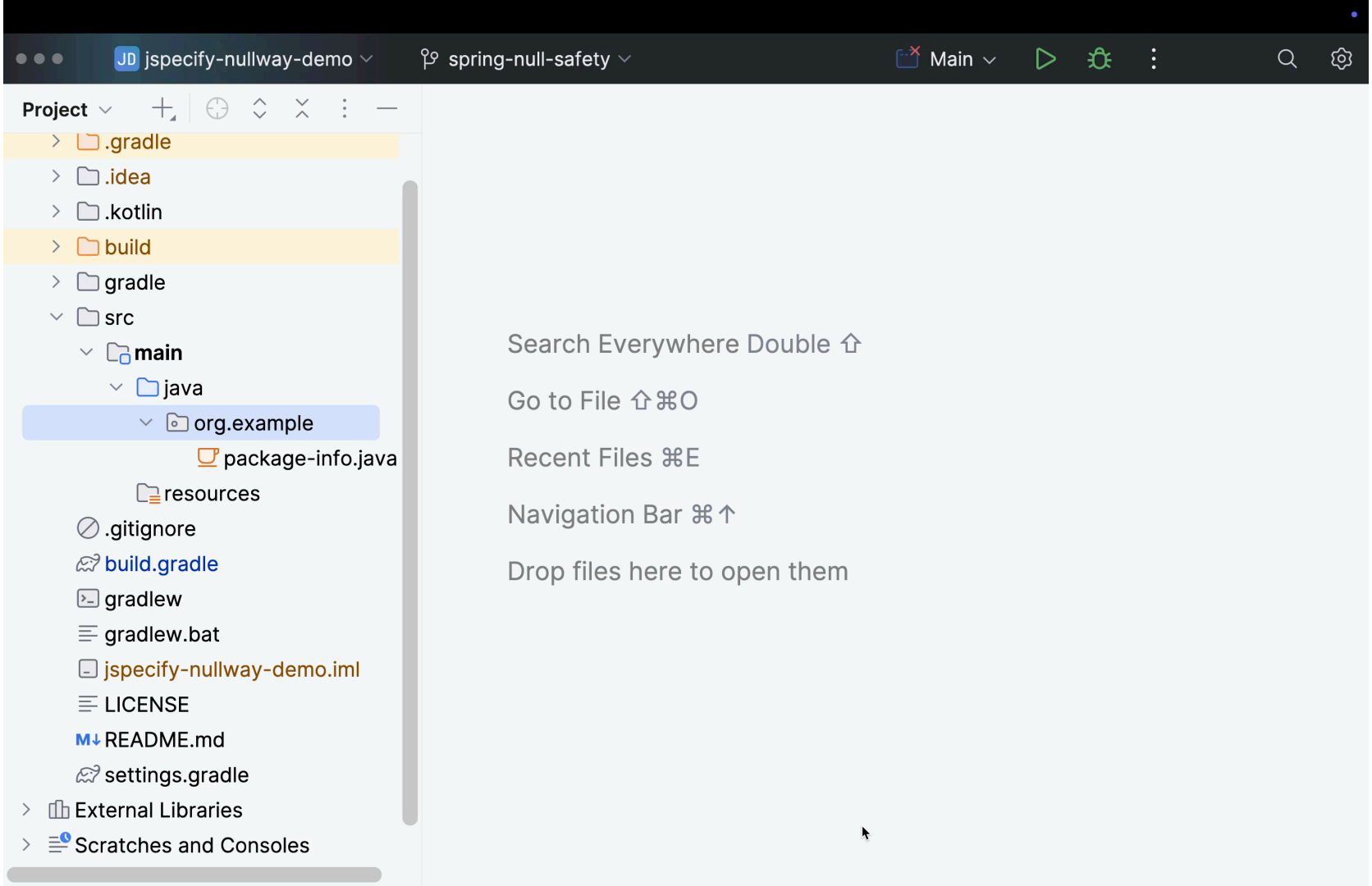
Spencer Gibb and 418 others

39 comments · 24 reposts

Reactions


JD jspecify-nullway-demo
main
Main.java
Main
package org.example;
public class Main {
public static void main(String[] args) {
}
}

Lazy initialization


jspecify-nullway-demo
spring-null-safety
Main
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

JEP 502: Stable Values (Preview)

Author	Per Minborg & Maurizio Cimadamore
Owner	Per-Ake Minborg
Type	Feature
Scope	SE
Status	Targeted
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	2025/04/01 14:52
Issue	8312611

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)

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

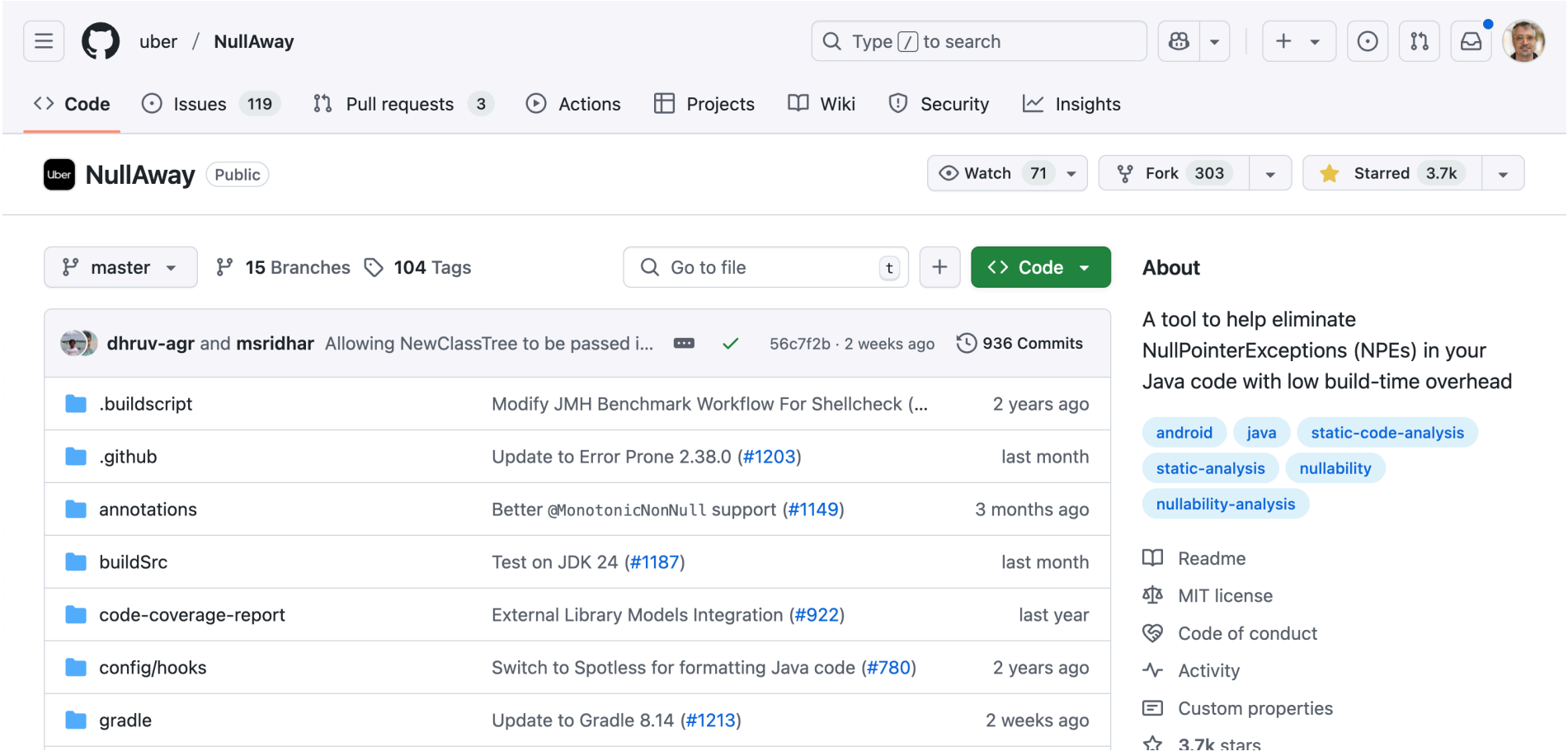
Summary

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
- Compatibibly 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



Thank you