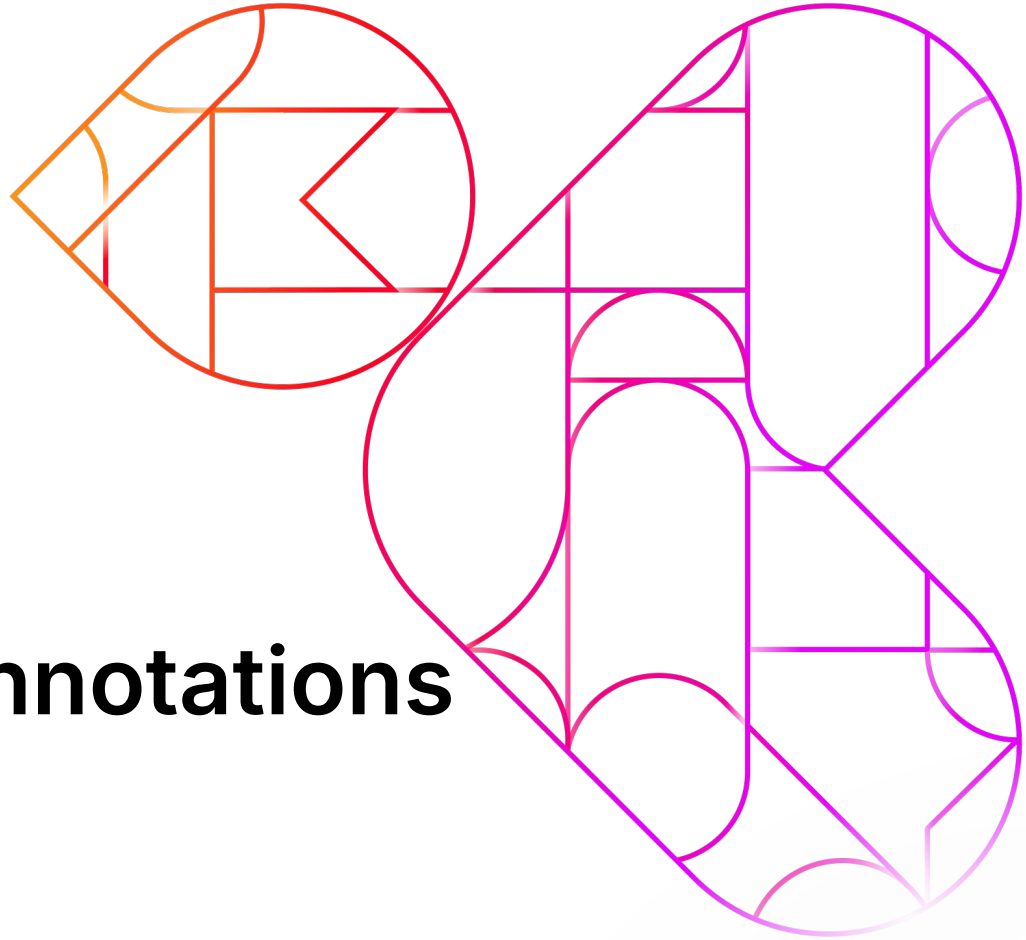
KotlinConf 2025



JSpecify:
Java Nullness Annotations and Kotlin

David Baker
Java and Kotlin Ecosystem Team @ Google

JSpecify

Working Group

- Google (Android, Error Prone, Guava, Truth…)
- JetBrains (Kotlin, IntelliJ)
- Oracle (Java, OpenJDK)
- Meta (Infer)
- Uber (NullAway)
- Broadcom (Spring Framework)
- Microsoft (Azure SDK for Java)
- Sonar (SonarQube, SonarCloud, SonarLint)
- Square (various libraries)
- EISOP (Checker Framework)

A **standard set of annotation types** to improve static analysis and language interop for JVM languages

Q: Why talk about Java nullness annotations at KotlinConf?

A: Platform Types

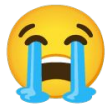
Platform Types

Kotlin has null-safe types...unless you use Java libraries.

Java API: `List<String> something(String arg);`

```
val isItThis: List<String> = something("string")
val orThis:   List<String?> = something("string")
val orThis2:  List<String>? = something("string")
val orThis3:  List<String?>? = something("string")

val doesThisNPE = something(null)
```



JSpecify: Nullness Annotations for Java

Java API:

```
@Nullable List<@NonNull String> something(@Nullable String arg);
```

```
val itsThis: List<String>? = something("string")
val wontNPE = something(null)
```

JSpecify Nullness

```
interface Simple {
  // returns String!
  String method1();

  // returns List<String!>!
  List<String> method2();
}
```

JSpecify Nullness

- `@Nullable`
- `@NonNull`

```
interface Simple {
  // returns String
  @NonNull String method1();

  // returns List<String?>
  @NonNull List<@Nullable String> method2();
}
```

JSpecify Nullness

- `@Nullable`
- `@NonNull`
- **`@NullMarked`**

```
@NullMarked
interface Simple {
  // returns String
  String method1();

  // returns List<String?>
  List<@Nullable String> method2();
}
```

JSpecify Nullness

- `@Nullable`
- `@NonNull`
- `@NullMarked`
- Parametric nullness

```
@NullMarked
interface Parameterized<T extends @Nullable Object> {
  // Parameterized<@NonNull String>: returns String
  // Parameterized<@Nullable String>: returns String?
  T foo();

  // returns T?
  @Nullable T bar();

  // returns T & Any
  @NonNull T baz();
}
```

JSpecify Nullness

- `@Nullable`
- `@NonNull`
- `@NullMarked`
- Parametric nullness
- **Unspecified nullness**
  - `@NullUnmarked`

```
interface Unannotated<T> {
  // returns String!
  String method1();

  // Unannotated<@NonNull String>: String!
  // Unannotated<@Nullable String>: String!
  T method2();
}
```

Don’t nullness annotations already exist?

Many possibilities:

- `androidx.annotation.Nullable`
- `edu.umd.cs.findbugs.annotations.Nullable`
- `jakarta.annotation.Nullable`
- `javax.annotation.Nullable` (JSR-305)
- `org.checkerframework.checker.nullness.qual.Nullable`
- `org.eclipse.jdt.annotation.Nullable`
- `org.jetbrains.annotations.Nullable`
- ….

Don’t nullness annotations already exist?

Many possibilities:

- `androidx.annotation.Nullable`
- `edu.umd.cs.findbugs.annotations.Nullable`
- `jakarta.annotation.Nullable`
- `javax.annotation.Nullable` (JSR-305)
- `org.checkerframework.checker.nullness.qual.Nullable`
- `org.eclipse.jdt.annotation.Nullable`
- `org.jetbrains.annotations.Nullable`
- ....

Some predate Java 8’s type-use annotation feature, so there’s no `List<@Nullable String>`.

Don’t nullness annotations already exist?

Many possibilities:

- `androidx.annotation.Nullable`
- `edu.umd.cs.findbugs.annotations.Nullable`
- `jakarta.annotation.Nullable`
- `javax.annotation.Nullable` (JSR-305)
- `org.checkerframework.checker.nullness.qual.Nullable`
- `org.eclipse.jdt.annotation.Nullable`
- `org.jetbrains.annotations.Nullable`
- ....

Most are defined by a specific analysis tool or IDE. But if you’re writing open-source libraries, *you don’t know which tool your users use*, and they all have subtly different semantics.

(defaults, type parameter bounds, wildcards, ...)

Don’t nullness annotations already exist?

Many possibilities:

- `androidx.annotation.Nullable`
- `edu.umd.cs.findbugs.annotations.Nullable`
- `jakarta.annotation.Nullable`
- `javax.annotation.Nullable` (JSR-305)
- `org.checkerframework.checker.nullness.qual.Nullable`
- `org.eclipse.jdt.annotation.Nullable`
- `org.jetbrains.annotations.Nullable`
- ....

None is **standard**.

JSpecify: Standard Nullness Annotations

https://xkcd.com/927

JSpecify: Standard Nullness Annotations

- JSpecify's nullness annotations have a **documented specification of their exact semantics**. They are **not tool-specific**.
- **Many tool providers and library authors participated** in the design process and are signing on to support and use them.
- All Java nullness analyzers (including the Kotlin compiler) can agree on **exactly what the nullness annotations mean**.
- 1.0 was released last July, and there are already ~500 Maven artifacts that use it.

JSpecify Users and Tools

Open-source libraries that use JSpecify

- Guava
- Truth
- Chromium
- Jetpack
- Spring Framework 7.0 (later this year)
- Apache Log4J
- Caffeine
- Gradle APIs (9.0)

Tools and IDEs that support JSpecify

- IntelliJ IDEA
- Android Studio
- Eclipse
- EISOP (Checker Framework)
- NullAway

JSpecify Language Support

- Helping OpenJDK on a possible future Java language nullness feature*.

- The Kotlin compiler treats JSpecify-annotated Java code as null-safe types as of version 2.1.0.

*openjdk.org/jeps/8303099

How does this help Kotlin users?

- Nullness-annotated Java libraries make Kotlin users null-safe.

- Nullness-annotated Java code is easier to convert to Kotlin.

JSpecify After Nullness

- ~~Java Nullness~~
- Checking return values
- Immutability
- ...

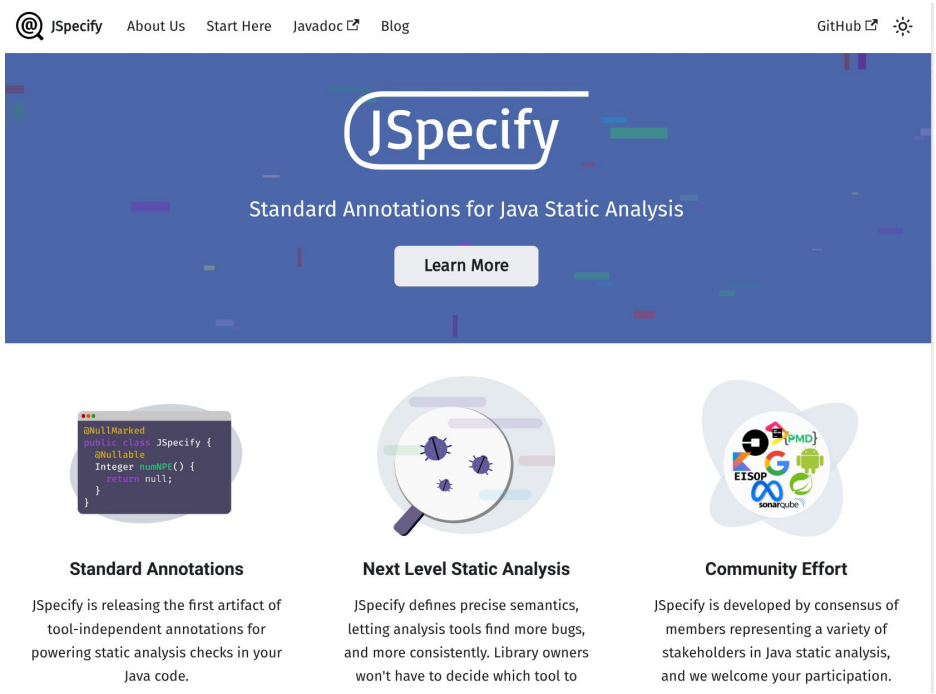
Get Involved!

JSpecify is open-source

- jspecify.dev
- github.com/jspecify

You can help

- Add JSpecify to open-source Java libraries
- Ask for JSpecify support from tools
- Help build our conformance test framework
- Join future domain discussions



Thank You, and Don’t Forget to Vote