




 Secrets


 ABAP


 Apex


 C


 C++


 CloudFormation


 COBOL


 C#


 CSS


 Flex


 Go


 HTML


 **Java**


 JavaScript


 Kotlin


 Objective C


 PHP


 PL/I


 PL/SQL


 Python


 RPG


 Ruby


 Scala


 Swift


 Terraform


 Text


 TypeScript

 T-SQL

 VB.NET

 VB6

 XML



## Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

All rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags ▾

Search by name... 🔍

Number patterns should be regular

Code Smell

Lazy initialization of "static" fields should be "synchronized"

Code Smell

Wildcard imports should not be used

Code Smell

Modulus results should not be checked for direct equality

Code Smell

Comparators should be "Serializable"

Code Smell

"Serializable" classes should have a "serialVersionUID"

Code Smell

"switch" statements and expressions should not be nested

Code Smell

Constructors should only call non-overridable methods

Code Smell

Methods should not be too complex

Code Smell

Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply

Code Smell

"if ... else if" constructs should end with "else" clauses

Code Smell

Control structures should use curly braces

Code Smell

### Nullness of parameters should be guaranteed

Analyze your code

Code Smell

Major ?

When using null-related annotations at global scope level, for instance using `javax.annotation.ParametersAreNonnullByDefault` (from JSR-305) at package level, it means that all the parameters to all the methods included in the package will, or should, be considered Non-null. It is equivalent to annotating every parameter in every method with non-null annotations (such as `@Nonnull`).

The rule raises an issue every time a parameter could be null for a method invocation, where the method is annotated as forbidding null parameters.

#### Noncompliant Code Example

```
@javax.annotation.ParametersAreNonnullByDefault
class A {

    void foo() {
        bar(getValue()); // Noncompliant - method 'bar' do not e
    }

    void bar(Object o) { // 'o' is by contract expected never
        // ...
    }

    @javax.annotation.CheckForNull
    abstract Object getValue();
}
```

#### Compliant Solution

Two solutions are possible:

- The signature of the method is correct, and null check should be done prior to the call.
- The signature of the method is not coherent and should be annotated to allow null values being passed as parameter

```
@javax.annotation.ParametersAreNonnullByDefault
abstract class A {





    void foo() {
        Object o = getValue();
        if (o != null) {
            bar(o); // Compliant - 'o' can not be null
        }
    }

    void bar(Object o) {
        // ...
    }

    @javax.annotation.CheckForNull
```

https://rules.sonarsource.com/java/RSPEC-4449

1/2

<div>Expressions should not be too complex</div> <div> Code Smell</div>
<div>Mockito argument matchers should be used on all parameters</div> <div> Bug</div>
<div>Spring "@Controller" classes should not use "@Scope"</div> <div> Bug</div>
<div>Constructor injection should be used instead of field injection</div> <div> Bug</div>

```
abstract Object getValue();
}
```

or

```
@javax.annotation.ParametersAreNonnullByDefault
abstract class A {

    void foo() {
        bar(getValue());
    }

    void bar(@javax.annotation.Nullable Object o) { // annotat
        // ...
    }

    @javax.annotation.CheckForNull
    abstract Object getValue();
}
```

Available In:  
sonarlint  | sonarcloud  | sonarqube 