




 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... 🔍


 Bug


Conditionally executed code should be reachable







"notifyAll" should be used







Blocks should be synchronized on "private final" fields







Non-serializable objects should not be stored in "HttpSession" objects







"wait", "notify" and "notifyAll" should only be called when a lock is obviously held on an object





Null pointers should not be dereferenced





Loop conditions should be true at least once

A "for" loop update clause should move the counter in the right direction

Non-public methods should not be "@Transactional"

Servlets should not have mutable instance fields

"toString()" and "clone()" methods should not return null

"equals" method parameters should not be marked "@Nonnull"

Analyze your code

Code Smell

Critical

Quick Fix

By contract, the `equals(Object)` method, from `java.lang.Object`, should accept a `null` argument. Among all the other cases, the `null` case is even explicitly detailed in the `Object.equals(...)` Javadoc, stating *"For any non-null reference value `x`, `x.equals(null)` should return `false`."*

Assuming that the argument to `equals` is always non-null, and enforcing that assumption with an annotation is not only a fundamental violation of the contract of `equals`, but it is also likely to cause problems in the future as the use of the class evolves over time.

The rule raises an issue when the `equals` method is overridden and its parameter annotated with any kind of `@Nonnull` annotation.

Noncompliant Code Example

```
public boolean equals(@javax.annotation.Nonnull Object obj)
// ...
}
```

Compliant Solution

```
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    // ...
}
```

Available In:

sonarlint





sonarcloud

sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.
[Privacy Policy](#)

https://rules.sonarsource.com/java/RSPEC-4454

1/2

<p>".equals()" should not be used to test the values of "Atomic" classes</p> <p> Bug</p>
<p>Return values from functions without side effects should not be ignored</p> <p> Bug</p>
<p>Child class methods named for parent class methods should be overrides</p> <p> Bug</p>
<p>Inappropriate "Collection" calls should not be made</p> <p> Bug</p>