




 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... 🔍

Code Smell

Execution of the Garbage Collector should be triggered only by the JVM

Code Smell

Constants should not be defined in interfaces

Code Smell

String literals should not be duplicated

Code Smell

Methods should not be empty

Code Smell

"Object.finalize()" should remain protected (versus public) when overriding

Code Smell

Exceptions should not be thrown in finally blocks

Code Smell

Constant names should comply with a naming convention

Code Smell

The Object.finalize() method should not be overridden

Code Smell

XML operations should not be vulnerable to injection attacks

Vulnerability

JSON operations should not be vulnerable to injection attacks

Vulnerability

XML signatures should be validated securely

Vulnerability

Assertions comparing incompatible types should not be made

Bug

Critical

tests

Assertions comparing incompatible types always fail, and negative assertions always pass. At best, negative assertions are useless. At worst, the developer loses time trying to fix his code logic before noticing wrong assertions.

Dissimilar types are:

- comparing a primitive with null
- comparing an object with an unrelated primitive (E.G. a string with an int)
- comparing unrelated classes
- comparing an array to a non-array
- comparing two arrays of dissimilar types

This rule also raises issues for unrelated class and interface or unrelated interface types in negative assertions. Because except in some corner cases, those types are more likely to be dissimilar. And inside a negative assertion, there is no test failure to inform the developer about this unusual comparison.

Supported test frameworks:

- JUnit4
- JUnit5
- AssertJ

Noncompliant Code Example

```
interface KitchenTool {}
interface Plant {}
class Spatula implements KitchenTool {}
class Tree implements Plant {}

void assertValues(int size,
                  Spatula spatula, KitchenTool tool, Kitch
                  Tree tree, Plant plant, Tree[])

// Whatever the given values, those negative assertions wi
assertThat(size).isNotNull(); // Noncompliant; p
assertThat(spatula).isNotEqualTo(tree); // Noncompliant; u
assertThat(tool).isNotSameAs(tools); // Noncompliant; a
assertThat(trees).isNotEqualTo(tools); // Noncompliant; i

// Those assertions will always fail
assertThat(size).isNull(); // Noncom
assertThat(spatula).isEqualTo(tree); // Noncom

// Those negative assertions are more likely to always pas
assertThat(spatula).isNotEqualTo(plant); // Noncompliant;
assertThat(tool).isNotEqualTo(plant); // Noncompliant;
}
```

https://rules.sonarsource.com/java/RSPEC-5845

1/2

XML parsers should not be vulnerable to Denial of Service attacks

 Vulnerability

XML parsers should not load external schemas

 Vulnerability

Mobile database encryption keys should not be disclosed

 Vulnerability

Reflection should not be vulnerable to injection attacks

 Vulnerability

- {rule:java:S2159} - Silly equality checks should not be made

Available In:

 |  | 

© 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](#)