




 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 rules 632

Vulnerability 53

Bug 154

Security Hotspot 36

Code Smell 389

Quick Fix 42

Tags ▾

Search by name... 🔍

☼ Code Smell

The names of methods with boolean return values should start with "is" or "has"

☼ Code Smell

☼ Code Smell

Files should contain only one top-level class or interface each

☼ Code Smell

☼ Code Smell

Classes should not have too many fields

☼ Code Smell

☼ Code Smell

The ternary operator should not be used

☼ Code Smell

☼ Code Smell

Standard functional interfaces should not be redefined

☼ Code Smell

☼ Code Smell

"NullPointerException" should not be caught

☼ Code Smell

☼ Code Smell

"NullPointerException" should not be explicitly thrown

☼ Code Smell

☼ Code Smell

Classes should not have too many methods

☼ Code Smell

☼ Code Smell

Methods should not have too many lines

☼ Code Smell

☼ Code Smell

Track uses of "NOSONAR" comments

☼ Code Smell

☼ Code Smell

Classes and enums with private members should have a constructor

☼ Code Smell

Boolean expressions should not be gratuitous

Analyze your code

☼ Code Smell

🔴 Major ?

🔍 cwe cert suspicious redundant

If a boolean expression doesn't change the evaluation of the condition, then it is entirely unnecessary, and can be removed. If it is gratuitous because it does not match the programmer's intent, then it's a bug and the expression should be fixed.

Noncompliant Code Example

```
a = true;
if (a) { // Noncompliant
    doSomething();
}

if (b && a) { // Noncompliant; "a" is always "true"
    doSomething();
}

if (c || !a) { // Noncompliant; "!a" is always "false"
    doSomething();
}
```

Compliant Solution

```
a = true;
if (foo(a)) {
    doSomething();
}




if (b) {
    doSomething();
}

if (c) {
    doSomething();
}
```

See

- [MITRE, CWE-571](#) - Expression is Always True
- [MITRE, CWE-570](#) - Expression is Always False




Available In:

 |  | 

https://rules.sonarsource.com/java/RSPEC-2589

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of

1/2

<div>Track comments matching a regular expression</div> <div> Code Smell</div>
<div>Statements should be on separate lines</div> <div> Code Smell</div>
<div>Classes should not be coupled to too many other classes (Single Responsibility Principle)</div> <div> Code Smell</div>
<div>"java.lang.Error" should not be extended</div>