
































-  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... 🔍

Abstract class names should comply with a naming convention

 Code Smell

Strings literals should be placed on the left side when checking for equality

 Code Smell

Files should contain an empty newline at the end

 Code Smell

Source code should be indented consistently

 Code Smell

A close curly brace should be located at the beginning of a line

 Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

 Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line

 Code Smell

An open curly brace should be located at the beginning of a line

 Code Smell

An open curly brace should be located at the end of a line

 Code Smell

Tabulation characters should not be used

 Code Smell

Functions should not be defined with a variable number of arguments

 Code Smell

Return of boolean expressions should not be wrapped into an "if-then-else" statement

Analyze your code

 Code Smell  Minor  clumsy

Return of boolean literal statements wrapped into if-then-else ones should be simplified.

Similarly, method invocations wrapped into if-then-else differing only from boolean literals should be simplified into a single invocation.

Noncompliant Code Example

```
boolean foo(Object param) {
    if (expression) { // Noncompliant
        bar(param, true, "qix");
    } else {
        bar(param, false, "qix");
    }

    if (expression) { // Noncompliant
        return true;
    } else {
        return false;
    }
}
```

Compliant Solution

```
boolean foo(Object param) {
    bar(param, expression, "qix");

    return expression;
}
```

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](#)

<b>Local-Variable Type Inference should be used</b>  Code Smell
<b>Migrate your tests from JUnit4 to the new JUnit5 annotations</b>  Code Smell
<b>Track uses of disallowed classes</b>  Code Smell
<b>Track uses of "@SuppressWarnings" annotations</b>  Code Smell