




 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... 🔍

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

Labels should not be used

Analyze your code

 Code Smell

 Major

 confusing

Labels are not commonly used in Java, and many developers do not understand how they work. Moreover, their usage makes the control flow harder to follow, which reduces the code's readability.

Noncompliant Code Example




```
int matrix[][] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};

outer: for (int row = 0; row < matrix.length; row++) { //
    for (int col = 0; col < matrix[row].length; col++) {
        if (col == row) {
            continue outer;
        }
        System.out.println(matrix[row][col]); //
    }
}
```

Compliant Solution

```
for (int row = 1; row < matrix.length; row++) { //
    for (int col = 0; col < row; col++) {
        System.out.println(matrix[row][col]); //
    }
}
```





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](#)

https://rules.sonarsource.com/java/RSPEC-1119

1/2

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