

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

Java

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

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... 🔍

"switch" statements and expressions should not be nested

Analyze your code

Code Smell

Critical

pitfall

Nested switch structures are difficult to understand because you can easily confuse the cases of an inner switch as belonging to an outer statement or expression. Therefore nested switch statements and expressions should be avoided.

Specifically, you should structure your code to avoid the need for nested switch statements or expressions, but if you cannot, then consider moving the inner switch to another method.

Noncompliant Code Example

```
void foo(int n, int m) {
    switch (n) {
        case 0:
            switch (m) { // Noncompliant; nested switch
                // ...
            }
        case 1:
            // ...
        default:
            // ...
    }
}
```

Compliant Solution

```
void foo(int n, int m) {
    switch (n) {
        case 0:
            bar(m);
        case 1:
            // ...
        default:
            // ...
    }
}

void bar(int m){
    switch(m) {
        // ...
    }
}
```

Available In:

sonarlint | sonarcloud | sonarqube


Local-Variable Type Inference should be used

 Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

 Code Smell

Track uses of disallowed classes

 Code Smell

Track uses of "@SuppressWarnings" annotations

 Code Smell

© 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](#)