




 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

Public methods should not contain selector arguments

Analyze your code

 Code Smell

 Major

 design

A selector argument is a boolean argument that's used to determine which of two paths to take through a method. Specifying such a parameter may seem innocuous, particularly if it's well named.

Unfortunately, the maintainers of the code calling the method won't see the parameter name, only its value. They'll be forced either to guess at the meaning or to take extra time to look the method up.

Instead, separate methods should be written.

This rule finds methods with a boolean that's used to determine which path to take through the method.

Noncompliant Code Example

```
public String tempt(String name, boolean ofAge) {
    if (ofAge) {
        offerLiquor(name);
    } else {
        offerCandy(name);
    }
}

// ...
public void corrupt() {
    tempt("Joe", false); // does this mean not to tempt Joe?
}
```

Compliant Solution

```
public void temptAdult(String name) {
    offerLiquor(name);
}

public void temptChild(String name) {
    offerCandy(name);
}

// ...
public void corrupt() {
    age < legalAge ? temptChild("Joe") : temptAdult("Joe");
}
```

Available In:

sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-2301

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>

© 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](#)