

 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

Arrays should not be created for varargs parameters

Analyze your code

 Code Smell

 Minor 

 clumsy

There's no point in creating an array solely for the purpose of passing it as a varargs (...) argument; varargs is an array. Simply pass the elements directly. They will be consolidated into an array automatically. Incidentally passing an array where Object ... is expected makes the intent ambiguous: Is the array supposed to be one object or a collection of objects?

Noncompliant Code Example

```
public void callTheThing() {
    //...
    doTheThing(new String[] { "s1", "s2"}); // Noncompliant:
    doTheThing(new String[12]); // Compliant
    doTheOtherThing(new String[8]); // Noncompliant: ambiguous
    // ...
}

public void doTheThing (String ... args) {
    // ...
}

public void doTheOtherThing(Object ... args) {
    // ...
}
```

Compliant Solution

```
public void callTheThing() {
    //...
    doTheThing("s1", "s2");
    doTheThing(new String[12]);
    doTheOtherThing((Object[]) new String[8]);
    // ...
}

public void doTheThing (String ... args) {
    // ...
}

public void doTheOtherThing(Object ... args) {
    // ...
}
```

Available In:

 |  | 

<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](#)