




 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... 🔍

Sections of code should not be commented out

Code Smell

Non-constructor methods should not have the same name as the enclosing class

Code Smell

Exception types should not be tested using "instanceof" in catch blocks

Code Smell

Classes from "sun.*" packages should not be used

Code Smell

Throwable and Error should not be caught

Code Smell

Unused method parameters should be removed

Code Smell

Only static class initializers should be used

Code Smell

Empty arrays and collections should be returned instead of null

Code Smell

"@Override" should be used on overriding and implementing methods

Code Smell

Enumeration should not be implemented

Code Smell

Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used

Code Smell

Unused "private" methods should be

"getClass" should not be used for synchronization

Analyze your code

BugMajor?multi-threading cert

getClass should not be used for synchronization in non-final classes because child classes will synchronize on a different object than the parent or each other, allowing multiple threads into the code block at once, despite the synchronized keyword.

Instead, hard code the name of the class on which to synchronize or make the class final.

Noncompliant Code Example

```
public class MyClass {
    public void doSomethingSynchronized(){
        synchronized (this.getClass()) { // Noncompliant
            // ...
        }
    }
}
```

Compliant Solution

```
public class MyClass {
    public void doSomethingSynchronized(){
        synchronized (MyClass.class) {
            // ...
        }
    }
}
```

See

- [CERT, LCK02-J](#) - Do not synchronize on the class object returned by getClass()

Available In:





sonarlint | sonarcloud | sonarqube

© 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-3067

1/2

<div>Unused private methods should be removed</div> <div> Code Smell</div>
<div>Try-catch blocks should not be nested</div> <div> Code Smell</div>
<div>Track uses of "FIXME" tags</div> <div> Code Smell</div>
<div>Deprecated elements should have both the annotation and the Javadoc tag</div> <div> Code Smell</div>
<div>Assignments should not be made from within sub-expressions</div>