




 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... 🔍

"Class.forName()" should not load JDBC 4.0+ drivers

Code Smell

Java features should be preferred to Guava

Code Smell

Nullness of parameters should be guaranteed

Code Smell

"Integer.toHexString" should not be used to build hexadecimal strings

Code Smell

Asserts should not be used to check the parameters of a public method

Code Smell

Assignments should not be redundant

Code Smell

Methods should not have identical implementations

Code Smell

"java.nio.Files#delete" should be preferred

Code Smell

Unused "private" classes should be removed

Code Smell

"Stream.peek" should be used with caution

Code Smell

"Map.get" and value test should be replaced with single method call

Code Smell

"@RequestMapping" methods should

Assertions should not be used in production code

Analyze your code

BugMajor?

Assertions are intended to be used in **test** code, but not in **production** code. It is confusing, and might lead to `ClassNotFoundException` when the build tools only provide the required dependency in test scope.

In addition, assertions will throw a sub-class of `Error`: `AssertionError`, which should be avoided in production code.

This rule raises an issue when any assertion intended to be used in test is used in production code.

Supported frameworks:

- JUnit
- FestAssert
- AssertJ

Note: this does not apply for `assert` from Java itself or if the source code package name is related to tests (contains: `test` or `assert` or `junit`).

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-5960

1/2

<div>not be "private"</div> <div> Code Smell</div>
<div>Raw types should not be used</div> <div> Code Smell</div>
<div>"Arrays.stream" should be used for primitive arrays</div> <div> Code Smell</div>
<div>Printf-style format strings should be used correctly</div> <div> Code Smell</div>
<div>Assertion arguments should be passed in the correct order</div>