




 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... 🔍

Redundant pairs of parentheses should be removed

Code Smell

Inheritance tree of classes should not be too deep

Code Smell

Nested blocks of code should not be left empty

Code Smell

Methods should not have too many parameters

Code Smell

Unused "private" fields should be removed

Code Smell

Collapsible "if" statements should be merged

Code Smell

Unused labels should be removed

Code Smell

Standard outputs should not be used directly to log anything

Code Smell

OS commands should not be vulnerable to argument injection attacks

Vulnerability

"ActiveMQConnectionFactory" should not be vulnerable to malicious code deserialization

Vulnerability

Logging should not be vulnerable to injection attacks

Vulnerability

Exceptions should not be thrown from

"="+ should not be used instead of "+="

Analyze your code

Bug Major ?

The use of operators pairs ( +=, -= or != ) where the reversed, single operator was meant ( +=, -= or != ) will compile and run, but not produce the expected results.

This rule raises an issue when +=, -=, or != is used without any spacing between the two operators and when there is at least one whitespace character after.

Noncompliant Code Example

```
int target = -5;
int num = 3;

target -= num; // Noncompliant; target = -3. Is that really
target += num; // Noncompliant; target = 3
```

Compliant Solution

```
int target = -5;
int num = 3;

target = -num; // Compliant; intent to assign inverse value
target += num;
```




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-2757

1/2

|   |
|---|
| <div>Exceptions checks not be ignored from<br/>servlet methods</div> <div> Vulnerability</div>                     |
| <div>Return values should not be ignored<br/>when they contain the operation<br/>status code</div> <div> Bug</div> |
| <div>Repeated patterns in regular<br/>expressions should not match the<br/>empty string</div> <div> Bug</div>      |
| <div>AssertJ assertions "allMatch" and<br/>"doesNotContains" should also test<br/>for emptiness</div>   |