




 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... 🔍

Code Smell

Control flow statements "if", "for", "while", "switch" and "try" should not be nested too deeply

Code Smell

"if ... else if" constructs should end with "else" clauses

Code Smell

Control structures should use curly braces

Code Smell

Expressions should not be too complex

Code Smell

Mockito argument matchers should be used on all parameters

Bug

Spring "@Controller" classes should not use "@Scope"

Bug

Constructor injection should be used instead of field injection

Bug

Classes that don't define "hashCode()" should not be used in hashes

Bug

Floating point numbers should not be tested for equality

Bug

Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression

Code Smell

Limited dependence should be placed on operator precedence

Methods should not have identical implementations

Analyze your code

Code Smell

Major ?

confusing duplicate suspicious

When two methods have the same implementation, either it was a mistake - something else was intended - or the duplication was intentional, but may be confusing to maintainers. In the latter case, one implementation should invoke the other. Numerical and string literals are not taken into account.

Noncompliant Code Example

```
private final static String CODE = "bounteous";

public String calculateCode() {
    doTheThing();
    return CODE;
}

public String getName() { // Noncompliant
    doTheThing();
    return CODE;
}
```

Compliant Solution

```
private final static String CODE = "bounteous";

public String getCode() {
    doTheThing();
    return CODE;
}

public String getName() {
    return getCode();
}
```

Exceptions

Methods that are not accessors (getters and setters), with fewer than 2 statements are ignored.

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-4144

1/2

on operator precedence
 Code Smell
Custom getter method should not be used to override record's getter behavior
 Code Smell
Tests should use fixed data instead of randomized data
 Code Smell
Spring's ModelAndViewAssert assertions should be used instead of other assertions
 Code Smell