




 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... 🔍

Code Smell

"Serializable" classes should have a "serialVersionUID"

Code Smell

"switch" statements and expressions should not be nested

Code Smell

Constructors should only call non-overridable methods

Code Smell

Methods should not be too complex

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

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

Code SmellMajor?pitfall

An assert is inappropriate for parameter validation because assertions can be disabled at runtime in the JVM, meaning that a bad operational setting would completely eliminate the intended checks. Further, asserts that fail throw `AssertionErrors`, rather than throwing some type of `Exception`. Throwing `Errors` is completely outside of the normal realm of expected catch/throw behavior in normal programs.

This rule raises an issue when a public method uses one or more of its parameters with asserts.

Noncompliant Code Example

```
public void setPrice(int price) {
    assert price >= 0 && price <= MAX_PRICE;
    // Set the price
}
```

Compliant Solution

```
public void setPrice(int price) {
    if (price < 0 || price > MAX_PRICE) {
        throw new IllegalArgumentException("Invalid price: " + p
    )
    }
    // Set the price
}
```

See

[Programming With Assertions](#)

Available In:




sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-4274

1/2

<div>Classes that don't define "hashCode()" should not be used in hashes</div> <div> Bug</div>
<div>Floating point numbers should not be tested for equality</div> <div> Bug</div>
<div>Increment (++) and decrement (--) operators should not be used in a method call or mixed with other operators in an expression</div> <div> Code Smell</div>
<div>Limited dependence should be placed on operator precedence</div>