sonar

RULES

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

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...

should be documented with Javadoc

Code Smell

Exception handlers should preserve the original exceptions

Code Smell

Checked exceptions should not be thrown

Code Smell

Public methods should throw at most one checked exception

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Methods should not have too many return statements

Code Smell

Magic numbers should not be used

Code Smell

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

JEE applications should not "getClassLoader"

Bug

Math should not be performed on floats

Bug

"equals" methods should be symmetric and work for subclasses

Bug

Silly math should not be performed

Analyze your code

Code Smell

Major

clumsy

Certain math operations are just silly and should not be performed because their results are predictable.

In particular, `anyValue % 1` is silly because it will always return 0.

Casting a non-floating-point value to floating-point and then passing it to `Math.round`, `Math.ceil`, or `Math.floor` is silly because the result will always be the original value.

These operations are silly with any constant value: `Math.abs`, `Math.ceil`, `Math.floor`, `Math rint`, `Math.round`.

And these oprations are silly with certain constant values:

Operation	Value
<code>acos</code>	0.0 or 1.0
<code>asin</code>	0.0 or 1.0
<code>atan</code>	0.0 or 1.0
<code>atan2</code>	0.0
<code>cbrt</code>	0.0 or 1.0
<code>cos</code>	0.0
<code>cosh</code>	0.0
<code>exp</code>	0.0 or 1.0
<code>expm1</code>	0.0
<code>log</code>	0.0 or 1.0
<code>log10</code>	0.0 or 1.0
<code>sin</code>	0.0
<code>sinh</code>	0.0
<code>sqrt</code>	0.0 or 1.0
<code>tan</code>	0.0
<code>tanh</code>	0.0
<code>toDegrees</code>	0.0 or 1.0
<code>toRadians</code>	0.0

Noncompliant Code Example

```
public void doMath(int a) {
    double floor = Math.floor((double)a); // Noncompliant
    double ceiling = Math.ceil(4.2); // Noncompliant
    double arcTan = Math.atan(0.0); // Noncompliant
}
```

Available In:


sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-2185

1/2

Literal suffixes should be upper case  Code Smell
Unicode-aware versions of character classes should be preferred  Code Smell
Use Java 12 "switch" expression  Code Smell
"serialVersionUID" should not be declared blindly  Code Smell
"Stream.collect()" calls should not be redundant

© 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](#)