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...

"Cloneables" should implement "clone"

Code Smell

Try-with-resources should be used

Code Smell

"readResolve" methods should be inheritable

Code Smell

"for" loop increment clauses should modify the loops' counters

Code Smell

Fields in a "Serializable" class should either be transient or serializable

Code Smell

Package declaration should match source file directory

Code Smell

Generic wildcard types should not be used in return types

Code Smell

"switch" statements should have "default" clauses

Code Smell

Execution of the Garbage Collector should be triggered only by the JVM

Code Smell

Constants should not be defined in interfaces

Code Smell

String literals should not be duplicated

Code Smell

Methods should not be empty

Code Smell

Back references in regular expressions should only refer to capturing groups that are matched before the reference

Analyze your code

Bug

Critical

regex

When a back reference in a regex refers to a capturing group that hasn't been defined yet (or at all), it can never be matched. Named back references throw a `PatternSyntaxException` in that case; numeric back references fail silently when they can't match, simply making the match fail.

When the group is defined before the back reference but on a different control path (like in `(.) | \1` for example), this also leads to a situation where the back reference can never match.

Noncompliant Code Example

```
Pattern.compile("\\1(.)"); // Noncompliant, group 1 is defin
Pattern.compile("(.)\\2"); // Noncompliant, group 2 isn't de
Pattern.compile("(.)|\\1"); // Noncompliant, group 1 and the
Pattern.compile("(?<x>.)|\\k<x>"); // Noncompliant, group x
```

Compliant Solution

```
Pattern.compile("(.)\\1");
Pattern.compile("(?<x>.)\\k<x>");
```

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-6001

1/2

<div><div>"Object.finalize()" should remain protected (versus public) when overriding</div><div> Code Smell</div></div>
<div><div>Exceptions should not be thrown in finally blocks</div><div> Code Smell</div></div>
<div><div>Constant names should comply with a naming convention</div><div> Code Smell</div></div>
<div><div>The Object.finalize() method should not be overridden</div><div> Code Smell</div></div>