




 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

Printf-style format strings should be used correctly

Code Smell

Assertion arguments should be passed in the correct order

Code Smell

Ternary operators should not be nested

Code Smell

"writeObject" should not be the only "synchronized" code in a class

Code Smell

Reflection should not be used to increase accessibility of classes, methods, or fields

Code Smell

Static fields should not be updated in constructors

Code Smell

"Thread.sleep" should not be used in tests

Code Smell

"entrySet()" should be iterated when both the key and value are needed

Code Smell

"DateUtils.truncate" from Apache Commons Lang library should not be used

Code Smell

Multiline blocks should be enclosed in curly braces

Code Smell

"readObject" should not be "synchronized"

Code Smell

Alternatives in regular expressions should be grouped when used with anchors

Bug

Major

regex

In regular expressions, anchors (^, \$, \A, \Z and \z) have higher precedence than the | operator. So in a regular expression like ^a1t1|a1t2|a1t3\$, a1t1 would be anchored to the beginning, a1t3 to the end and a1t2 wouldn't be anchored at all. Usually the intended behavior is that all alternatives are anchored at both ends. To achieve this, a non-capturing group should be used around the alternatives.

In cases where it is intended that the anchors only apply to one alternative each, adding (non-capturing) groups around the anchors and the parts that they apply to will make it explicit which parts are anchored and avoid readers misunderstanding the precedence or changing it because they mistakenly assume the precedence was not intended.

Noncompliant Code Example

```
^a|b|c$
```

Compliant Solution

```
^(?:a|b|c)$
```

or

```
^a$|^b$|^c$
```

or, if you do want the anchors to only apply to a and c respectively:

```
(?:^a)|b|(?:c$)
```

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-5850

1/2

 Code Smell
"Preconditions" and logging arguments should not require evaluation  Code Smell
Boolean expressions should not be gratuitous  Code Smell
"Lock" objects should not be "synchronized"  Code Smell
Classes with only "static" methods should not be instantiated