




 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... 🔍

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

"Object.finalize()" should remain

### Regex boundaries should not be used in a way that can never be matched

Analyze your code

Bug

Critical

?

regex

In regular expressions the boundaries `^` and `\A` can only match at the beginning of the input (or, in case of `^` in combination with the `MULTILINE` flag, the beginning of the line) and `$`, `\z` and `\Z` only at the end.

These patterns can be misused, by accidentally switching `^` and `$` for example, to create a pattern that can never match.

#### Noncompliant Code Example

```
// This can never match because $ and ^ have been switched a
Pattern.compile("$[a-z]^"); // Noncompliant
```

#### Compliant Solution

```
Pattern.compile("^ [a-z]+$");
```

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-5996

1/2

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