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 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...

Spring beans should be considered by "@ComponentScan"

Code Smell

Number patterns should be regular

Code Smell

Lazy initialization of "static" fields should be "synchronized"

Code Smell

Wildcard imports should not be used

Code Smell

Modulus results should not be checked for direct equality

Code Smell

Comparators should be "Serializable"

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

Java features should be preferred to Guava

Analyze your code

Code SmellMajor?java9 java8

Some Guava features were really useful for Java 7 application because Guava was bringing APIs missing in the JDK. Java 8 fixed some of these limitations. When migrating an application to Java 8 or even when starting a new one, it's recommended to prefer Java 8 APIs over Guava ones to ease its maintenance: developers don't need to learn how to use two APIs and can stick to the standard one.

Java 9 brought even more useful methods to the standard Java library and if Java version is equal to or higher than 9, these standard methods should be used.

This rule raises an issue when the following Guava APIs are used:

Guava API	Java 8 API
com.google.common.io.BaseEncoding#base64()	java.util.Base64
com.google.common.io.BaseEncoding#base64Url()	java.util.Base64
com.google.common.base.Joiner.on()	java.lang.String#join() or java.util.stream.Collectors#joining()
com.google.common.base.Optional#of()	java.util.Optional#of()
com.google.common.base.Optional#absent()	java.util.Optional#empty()
com.google.common.base.Optional#fromNullable()	java.util.Optional#ofNullable()
com.google.common.base.Optional	java.util.Optional
com.google.common.base.Predicate	java.util.function.Predicate
com.google.common.base.Function	java.util.function.Function
com.google.common.base.Supplier	java.util.function.Supplier
com.google.common.io.Files.createTempDir	java.nio.file.Files.createTempDirectory

Guava API	Java 9 API
com.google.common.collect.ImmutableSet#of()	java.util.Set#of()
com.google.common.collect.ImmutableList#of()	java.util.List#of()
com.google.common.collect.ImmutableMap#of()	java.util.Map#of() or java.util.Map#ofEntries()

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-4738

1/2

<p>Control structures should use curly braces</p> <p> Code Smell</p>
<p>Expressions should not be too complex</p> <p> Code Smell</p>
<p>Mockito argument matchers should be used on all parameters</p> <p> Bug</p>
<p>Spring "@Controller" classes should not use "@Scope"</p>