




 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... 🔍

Limited dependence should be placed on operator precedence

Code Smell

Custom getter method should not be used to override record's getter behavior

Code Smell

Tests should use fixed data instead of randomized data

Code Smell

Spring's ModelAndViewAssert assertions should be used instead of other assertions

Code Smell

Lambdas should not have too many lines

Code Smell

"@EnableAutoConfiguration" should be fine-tuned

Code Smell

Enum values should be compared with "=="

Code Smell

Spring components should use constructor injection

Code Smell

Regex patterns should not be created needlessly

Code Smell

Track uses of disallowed constructors

Code Smell

Java 8's "Files.exists" should not be used

Code Smell

"Optional" should not be used for parameters

Raw types should not be used

Analyze your code

Code Smell

Major ?

pitfall

Generic types shouldn't be used raw (without type parameters) in variable declarations or return values. Doing so bypasses generic type checking, and defers the catch of unsafe code to runtime.

Noncompliant Code Example

```
List myList; // Noncompliant
Set mySet; // Noncompliant
```

Compliant Solution

```
List<String> myList;
Set<? extends Number> mySet;
```





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-3740

1/2

 Code Smell
Tests should be kept in a dedicated source directory  Code Smell
"this" should not be exposed from constructors  Code Smell
Classes should not have too many "static" imports  Code Smell
Escaped Unicode characters should not be used