

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...

Bug

"toArray" should be passed an array of the proper type

Bug

Neither "Math.abs" nor negation should be used on numbers that could be "MIN\_VALUE"

Bug

The value returned from a stream read should be checked

Bug

"@NonNull" values should not be set to null

Bug

"Iterator.next()" methods should throw "NoSuchElementException"

Bug

"compareTo" results should not be checked for specific values

Bug

Math operands should be cast before assignment

Bug

Ints and longs should not be shifted by zero or more than their number of bits-1

Bug

"compareTo" should not return "Integer.MIN\_VALUE"

Bug

Boxing and unboxing should not be immediately reversed

Bug

"equals(Object obj)" should test argument type

## Non-public methods should not be "@Transactional"

Analyze your code

Bug

Major

spring

Marking a non-public method `@Transactional` is both useless and misleading because Spring doesn't "see" non-public methods, and so makes no provision for their proper invocation. Nor does Spring make provision for the methods invoked by the method it called.

Therefore marking a private method, for instance, `@Transactional` can only result in a runtime error or exception if the method is actually written to be `@Transactional`.

### Noncompliant Code Example

```
@Transactional // Noncompliant
private void doTheThing(ArgClass arg) {
    // ...
}
```

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-2230

1/2

 Bug
"Serializable" inner classes of non-serializable classes should be "static"
 Bug
The non-serializable super class of a "Serializable" class should have a no-argument constructor
 Bug
Method parameters, caught exceptions and foreach variables' initial values should not be ignored
 Bug
"equals(Object obj)" and "hashCode()"