




 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... 🔍

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

"Preconditions" and logging arguments should not require evaluation

Code Smell

Boolean expressions should not be gratuitous

Code Smell

"Lock" objects should not be

AssertJ configuration should be applied

Analyze your code

Bug

Major

tests assertj

A org.assertj.core.configuration.Configuration will be effective only once you call Configuration.apply() or Configuration.applyAndDisplay().

This rule raises an issue when configurations are set without the appropriate call to apply them.

Noncompliant Code Example

```
Configuration configuration = new Configuration(); // Noncom
configuration.setComparingPrivateFields(true); {code}
```

Compliant Solution

```
Configuration configuration = new Configuration();
configuration.setComparingPrivateFields(true);
configuration.applyAndDisplay();
// Alternatively: configuration.apply();
```

See

• [AssertJ configuration documentation](#)

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-5831

1/2

| |
|---|
| <div>lock objects created with "synchronized"</div> <div> Code Smell</div> |
| <div>Classes with only "static" methods should not be instantiated</div> <div> Code Smell</div> |
| <div>"Threads" should not be used where "Runnables" are expected</div> <div> Code Smell</div> |
| <div>Inner class calls to super class methods should be unambiguous</div> <div> Code Smell</div> |
| <div>Unused type parameters should be removed</div> |