sonar RULES Products

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 rules 632 | Vulnerability 53 | Bug 154 | Security Hotspot 36 | Code Smell 389 | Quick Fix 42

Tags | Search by name...

Code Smell

Printf-style format strings should be used correctly
Code Smell

Assertion arguments should be passed in the correct order
Code Smell

Ternary operators should not be 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"

AssertJ methods setting the assertion context should come before an assertion

Analyze your code

Bug Major tests assertj

Describing, setting error message or adding a comparator in AssertJ must be done before calling the assertion, otherwise, settings will not be taken into account.

This rule raises an issue when one of the method (with all similar methods):

- `as`
- `describedAs`
- `withFailMessage`
- `overridingErrorMessage`
- `usingComparator`
- `usingElementComparator`
- `extracting`
- `filteredOn`

is called without calling an AssertJ assertion afterward.

Noncompliant Code Example

```
assertThat(actual).isEqualTo(expected).as("Description"); //
assertThat(actual).isEqualTo(expected).withFailMessage("Fail
assertThat(actual).isEqualTo(expected).usingComparator(new C
```

Compliant Solution

```
assertThat(actual).as("Description").isEqualTo(expected);
assertThat(actual).withFailMessage("Fail message").isEqualTo
assertThat(actual).usingComparator(new CustomComparator()).i
```

See

- AssertJ incorrect usage documentation

Available In:
sonarlint sonarcloud sonarqube

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 "synchronized"

Code Smell

Classes with only "static" methods should not be instantiated