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...

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
Code Smell

Tests should be kept in a dedicated source directory
Code Smell

"this" should not be exposed from constructors

Assertion arguments should be passed in the correct order

Analyze your code

Code Smell   Major   Quick Fix   junit tests suspicious

The standard assertions library methods such as `org.junit.Assert.assertEquals`, and `org.junit.Assert.assertSame` expect the first argument to be the expected value and the second argument to be the actual value. For AssertJ, it's the other way around, the argument of `org.assertj.core.api.Assertions.assertThat` is the actual value, and the subsequent calls contain the expected values. Swap them, and your test will still have the same outcome (succeed/fail when it should) but the error messages will be confusing.

This rule raises an issue when the actual argument to an assertions library method is a hard-coded value and the expected argument is not.

Supported frameworks:

- JUnit4
- JUnit5
- AssertJ

Noncompliant Code Example

```
org.junit.Assert.assertEquals(runner.exitCode(), 0, "Unexpec
org.assertj.core.api.Assertions.assertThat(0).isEqualTo(runn
```

Compliant Solution

```
org.junit.Assert.assertEquals(0, runner.exitCode(), "Unexpec
org.assertj.core.api.Assertions.assertThat(runner.exitCode()
```

Available In:
sonarlint sonarcloud sonarqube

Code Smell

Classes should not have too many "static" imports

Code Smell

Escaped Unicode characters should not be used

Code Smell

Inner classes should not have too many lines of code

Code Smell

Inner classes which do not reference their owning classes should be "static"