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...

Primitive wrappers should not be instantiated only for "toString" or "compareTo" calls
Code Smell

Case insensitive string comparisons should be made without intermediate upper or lower casing
Code Smell

Collection.isEmpty() should be used to test for emptiness
Code Smell

String.valueOf() should not be appended to a String
Code Smell

Interface names should comply with a naming convention
Code Smell

"throws" declarations should not be superfluous
Code Smell

Unnecessary imports should be removed
Code Smell

Return of boolean expressions should not be wrapped into an "if-then-else" statement
Code Smell

Boolean literals should not be redundant
Code Smell

Modifiers should be declared in the correct order
Code Smell

Empty statements should be removed
Code Smell

Test methods should not contain too many assertions

Analyze your code

Code Smell | Major | tests

A common good practice is to write test methods targeting only one logical concept, that can only fail for one reason.

While it might make sense to have more than one assertion to test one concept, having too many is a sign that a test became too complex and should be refactored to multiples ones.

This rule will report any test method containing more than a given number of assertion.

Noncompliant Code Example

With a parameter of 2.

```
@Test
void test() { // Refactor this method.
  assertEquals(1, f(1));
  assertEquals(2, f(2));
  assertEquals(3, g(1));
}
```

Compliant Solution

```
void test_f() {
  assertEquals(1, f(1));
  assertEquals(2, f(2));
}
void test_g() {
  assertEquals(3, g(1));
}
```

Available In:

sonarlint | sonarcloud | sonarqube

Class variable fields should not have public accessibility

Code Smell

URIs should not be hardcoded

Code Smell

Class names should comply with a naming convention

Code Smell

Method names should comply with a naming convention

Code Smell

Comma-separated labels should be