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...

naming convention
Code Smell

Value-based objects should not be serialized
Code Smell

Default annotation parameter values should not be passed as arguments
Code Smell

Method parameters should be declared with base types
Code Smell

Fields should not be initialized to default values
Code Smell

Multiple loops over the same set should be combined
Code Smell

Classes without "public" constructors should be "final"
Code Smell

Unnecessary semicolons should be omitted
Code Smell

Literal boolean values and nulls should not be used in assertions
Code Smell

Test assertions should include messages
Code Smell

Mutable members should not be stored or returned directly
Code Smell

Redundant modifiers should not be used

JUnit4 @Ignored and JUnit5 @Disabled annotations should be used to disable tests and should provide a rationale

Analyze your code

Code Smell | Major | junit tests bad-practice confusing suspicious

When a test fails due, for example, to infrastructure issues, you might want to ignore it temporarily. But without some kind of notation about why the test is being ignored, it may never be reactivated. Such tests are difficult to address without comprehensive knowledge of the project, and end up polluting their projects.

This rule raises an issue for each ignored test that does not have any comment about why it is being skipped.

- For Junit4, this rule targets the @Ignore annotation.
- For Junit5, this rule targets the @Disabled annotation.
- Cases where assumeTrue(false) or assumeFalse(true) are used to skip tests are targeted as well.

Noncompliant Code Example

```
@Ignore  // Noncompliant
@Test
public void testDoTheThing() {
  // ...
```

or

```
@Test
public void testDoTheThing() {
  Assume.assumeFalse(true); // Noncompliant
  // ...
```

Compliant Solution

```
@Test
@Ignore("See Ticket #1234")
public void testDoTheThing() {
  // ...
```

Available In:
sonarlint | sonarcloud | sonarqube

Code Smell

"private" and "final" methods that don't access instance data should be "static"

Code Smell

Files should not be empty

Code Smell

Collection methods with O(n) performance should be used carefully

Code Smell

"Exception" should not be caught when not required by called methods