Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Conditionally executed code should be reachable

Analyze your code

Bug Major cwe cert unused suspicious pitfall

Conditional expressions which are always `true` or `false` can lead to dead code. Such code is always buggy and should never be used in production.

Noncompliant Code Example

```
a = false;
if (a) { // Noncompliant
  doSomething(); // never executed
}

if (!a || b) { // Noncompliant; "!a" is always "true", "b" i
  doSomething();
} else {
  doSomethingElse(); // never executed
}
```

Exceptions

This rule will not raise an issue in either of these cases:

- When the condition is a single `final boolean`

```
final boolean debug = false;
//...
if (debug) {
  // Print something
}
```

- When the condition is literally `true` or `false`.

```
if (true) {
  // do something
}
```

In these cases it is obvious the code is as intended.

See

- MITRE, CWE-570 - Expression is Always False
- MITRE, CWE-571 - Expression is Always True
- CERT, MSC12-C. - Detect and remove code that has no effect or is never executed

Available In:

sonarlint sonarcloud sonarqube

Repeated patterns in regular expressions should not match the empty string

Bug

AssertJ assertions "allMatch" and "doesNotContains" should also test for emptiness

Bug

Double Brace Initialization should not be used

Bug

Non-primitive fields should not be