




 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

Abstract class names should comply with a naming convention	Code Smell
Strings literals should be placed on the left side when checking for equality	Code Smell
Files should contain an empty newline at the end	Code Smell
Source code should be indented consistently	Code Smell
A close curly brace should be located at the beginning of a line	Code Smell
Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines	Code Smell
Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line	Code Smell
An open curly brace should be located at the beginning of a line	Code Smell
An open curly brace should be located at the end of a line	Code Smell
Tabulation characters should not be used	Code Smell
Functions should not be defined with a variable number of arguments	Code Smell

"if ... else if" constructs should end with "else" clauses

Analyze your code

- Code Smell
- Critical
- cert

This rule applies whenever an `if` statement is followed by one or more `else if` statements; the final `else if` should be followed by an `else` statement.

The requirement for a final `else` statement is defensive programming.

The `else` statement should either take appropriate action or contain a suitable comment as to why no action is taken. This is consistent with the requirement to have a final default clause in a `switch` statement.

Noncompliant Code Example

```
if (x == 0) {
    doSomething();
} else if (x == 1) {
    doSomethingElse();
}
```

Compliant Solution

```
if (x == 0) {
    doSomething();
} else if (x == 1) {
    doSomethingElse();
} else {
    throw new IllegalStateException();
}
```

See

- CERT, MSC01-C. - Strive for logical completeness
- CERT, MSC57-J. - Strive for logical completeness

Available In:

sonarlint | sonarcloud | sonarqube

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>