




 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



Scala static code analysis

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

All rules 41

Bug 6

Security Hotspot 2

Code Smell 33

Tags ▾

Search by name... 🔍

Boolean literals should not be redundant

Code Smell

Class names should comply with a naming convention

Code Smell

Method names should comply with a naming convention

Code Smell

Track uses of "TODO" tags

Code Smell

Track lack of copyright and license headers

Code Smell

"match" statements should not be nested

Code Smell

Control flow statements "if", "for", "while", "match" and "try" should not be nested too deeply

Code Smell

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

Code Smell

Expressions should not be too complex

Code Smell

Scala parser failure

Code Smell

Methods should not have too many lines of code

Code Smell

Statements should be on separate lines

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

Analyze your code

Code Smell Critical ?

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 `case _` clause in a `match`.

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
}
```

Exceptions

When all branches of an `if-else if` end with `return`, `break` or `throw`, the code that comes after the `if` implicitly behaves as if it was in an `else` clause. This rule will therefore ignore that case.

Available In:






sonarlint sonarcloud sonarqube

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.

[Privacy Policy](#)

https://rules.sonarsource.com/scala/RSPEC-126

1/2

 Code Smell
"match case" clauses should not have too many lines of code  Code Smell
Files should not have too many lines of code  Code Smell
Lines should not be too long  Code Smell
Tabulation characters should not be used  Code Smell