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...

HTTP response headers should not be vulnerable to injection attacks
Vulnerability

Members of Spring components should be injected
Vulnerability

Classes should not be loaded dynamically
Vulnerability

Equality operators should not be used in "for" loop termination conditions
Code Smell

"Bean Validation" (JSR 380) should be properly configured
Code Smell

Spring beans should be considered by "@ComponentScan"
Code Smell

Number patterns should be regular
Code Smell

Lazy initialization of "static" fields should be "synchronized"
Code Smell

Wildcard imports should not be used
Code Smell

Modulus results should not be checked for direct equality
Code Smell

Comparators should be "Serializable"
Code Smell

"Serializable" classes should have a "serialVersionUID"
Code Smell

"else" statements should be clearly matched with an "if"

Analyze your code

Code Smell | Major | confusing

The dangling `else` problem appears when nested `if/else` statements are written without curly braces. In this case, `else` is associated with the nearest `if` but that is not always obvious and sometimes the indentation can also be misleading.

This rules reports `else` statements that are difficult to understand, because they are inside nested `if` statements without curly braces.

Adding curly braces can generally make the code clearer (see rule {rule:java:S121}), and in this situation of dangling `else`, it really clarifies the intention of the code.

Noncompliant Code Example

```
if (a)
  if (b)
    d++;
else     // Noncompliant, is the "else" associated with "if
  e++;
```

Compliant Solution

```
if (a) {
  if (b) {
    d++;
  }
} else { // Compliant, there is no doubt the "else" is asso
  e++;
}
```

See

- https://en.wikipedia.org/wiki/Dangling_else

Available In:
sonarlint sonarcloud sonarqube

"switch" statements and expressions should not be nested

Code Smell

Constructors should only call non-overridable methods

Code Smell

Methods should not be too complex

Code Smell

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

Code Smell