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

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

Switch arrow labels should not use redundant keywords

Analyze your code

Code Smell Minor java14

In Switch Expressions, an arrow label consisting of a block with a single `yield` can be simplified to directly return the value, resulting in cleaner code.

Similarly, for Switch Statements and arrow labels, a `break` in a block is always redundant and should not be used. Furthermore, if the resulting block contains only one statement, the curly braces of that block can also be omitted.

This rule reports an issue when a case of a Switch Expression contains a block with a single `yield` or when a Switch Statement contains a block with a `break`.

Noncompliant Code Example

```
int i = switch (mode) {
  case "a" -> {         // Noncompliant: Remove the redundant
    yield 1;
  }
  default -> {          // Noncompliant: Remove the redundant
    yield 2;
  }
};

switch (mode) {
  case "a" -> {         // Noncompliant: Remove the redundant
    result = 1;
    break;
  }
  default -> {          // Noncompliant: Remove the redundant
    doSomethingElse();
    result = 2;
    break;
  }
}
```

Compliant Solution

```
int i = switch (mode) {
  case "a" -> 1;
  default -> 2;
};

switch (mode) {
  case "a" -> result = 1;
  default -> {
   doSomethingElse();
   result = 2;
 }
}
```

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell

See

- JEP 361: Switch Expressions

Available In:

sonarlint sonarcloud sonarqube