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...

"ThreadLocal.withInitial" should be preferred
Code Smell

"Stream" call chains should be simplified when possible
Code Smell

Packages containing only "package-info.java" should be removed
Code Smell

Arrays should not be created for varargs parameters
Code Smell

Jump statements should not be redundant
Code Smell

Test classes should comply with a naming convention
Code Smell

Loggers should be named for their enclosing classes
Code Smell

Methods should not return constants
Code Smell

"private" methods called only by inner classes should be moved to those classes
Code Smell

"enum" fields should not be publicly mutable
Code Smell

Abstract methods should not be redundant
Code Smell

Arrays should not be copied using loops

Loops with at most one iteration should be refactored

Analyze your code

Bug Major

A loop with at most one iteration is equivalent to the use of an `if` statement to conditionally execute one piece of code. No developer expects to find such a use of a loop statement. If the initial intention of the author was really to conditionally execute one piece of code, an `if` statement should be used instead.

At worst that was not the initial intention of the author and so the body of the loop should be fixed to use the nested `return`, `break` or `throw` statements in a more appropriate way.

Noncompliant Code Example

```
for (int i = 0; i < 10; i++) { // noncompliant, loop only ex
  printf("i is %d", i);
  break;
}
...
for (int i = 0; i < 10; i++) { // noncompliant, loop only ex
  if (i == x) {
    break;
  } else {
    printf("i is %d", i);
    return;
  }
}
```

Compliant Solution

```
for (int i = 0; i < 10; i++) {
  printf("i is %d", i);
}
...
for (int i = 0; i < 10; i++) {
  if (i == x) {
    break;
  } else {
    printf("i is %d", i);
  }
}
```

Available In:

sonarlint sonarcloud sonarqube

Code Smell

Static non-final field names should comply with a naming convention

Code Smell

JUnit rules should be used

Code Smell

Nested "enum"s should not be declared static

Code Smell

"catch" clauses should do more than rethrow