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...

"static" base class members should not be accessed via derived types
Code Smell

Instance methods should not write to "static" fields
Code Smell

"indexOf" checks should not be for positive numbers
Code Smell

Method overrides should not change contracts
Code Smell

Whitespace and control characters in literals should be explicit
Code Smell

Null should not be returned from a "Boolean" method
Code Smell

Classes should not access their own subclasses during initialization
Code Smell

"Object.wait(...)" and "Condition.await(...)" should be called inside a "while" loop
Code Smell

IllegalMonitorStateException should not be caught
Code Smell

JUnit assertions should not be used in "run" methods
Code Smell

Class names should not shadow interfaces or superclasses
Code Smell

"Cloneables" should implement "clone"

Calls to methods should not trigger an IllegalArgumentException

Analyze your code

Bug Critical

It's common for methods to check the value of their parameters and throw an `IllegalArgumentException` when one of them doesn't match a given condition. Those conditions are usually mentioned in the javadoc of the method.

This rule raises an issue when it detects that a method call has an argument which will trigger an `IllegalArgumentException`.

Noncompliant Code Example

```
private void addFirst(Collection<String> collection, Strin
  if (collection instanceof List) {
    ((List<String>) collection).add(0, element);
  } else {
    throw new IllegalArgumentException();
  }
}

public void caller() {
  // Noncompliant: triggers an IllegalArgumentException be
  addFirst(new HashSet<>(), "hello");
}
```

Available In:
sonarcloud

Code Smell

Try-with-resources should be used

Code Smell

"readResolve" methods should be inheritable

Code Smell

"for" loop increment clauses should modify the loops' counters

Code Smell

Fields in a "Serializable" class should either be transient or serializable

Code Smell