

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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags

Search by name...

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

Exception classes should be immutable

Analyze your code

Code Smell

Minor

error-handling

Exceptions are meant to represent the application's state at the point at which an error occurred.

Making all fields in an Exception class final ensures that this state:

- Will be fully defined at the same time the Exception is instantiated.
- Won't be updated or corrupted by a questionable error handler.

This will enable developers to quickly understand what went wrong.

Noncompliant Code Example

```
public class MyException extends Exception {

    private int status;                                // Nonco

    public MyException(String message) {
        super(message);
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }

}
```

Compliant Solution

```
public class MyException extends Exception {

    private final int status;


    public MyException(String message, int status) {
        super(message);
        this.status = status;
    }

    public int getStatus() {
        return status;
    }


}
```

Available In:


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

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](#)