

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

Source code should be indented consistently

Analyze your code

Code Smell

Minor

convention

Proper indentation is a simple and effective way to improve the code's readability. Consistent indentation among the developers within a team also reduces the differences that are committed to source control systems, making code reviews easier.

This rule raises an issue when indentation does not match the configured value. Only the first line of a badly indented section is reported.

Noncompliant Code Example

With an indent size of 2:

```
class Foo {
    public int a;
    public int b; // Noncompliant, expected to start at col

    ...

    public void doSomething() {
        if(something) {
            doSomethingElse(); // Noncompliant, expected to s
        } // Noncompliant, expected to start at column 4
    }
}
```

Compliant Solution

```
class Foo {
    public int a;
    public int b;

    ...

    public void doSomething() {
        if(something) {
            doSomethingElse();
        }
    }
}
```

Available In:

sonarlint





sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-1120

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, 1/2

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](#)

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>