




 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

Classes and enums with private members should have a constructor	Code Smell
Track comments matching a regular expression	Code Smell
Statements should be on separate lines	Code Smell
Classes should not be coupled to too many other classes (Single Responsibility Principle)	Code Smell
"java.lang.Error" should not be extended	Code Smell
Anonymous classes should not have too many lines	Code Smell
Public types, methods and fields (API) should be documented with Javadoc	Code Smell
Exception handlers should preserve the original exceptions	Code Smell
Checked exceptions should not be thrown	Code Smell
Public methods should throw at most one checked exception	Code Smell
"switch case" clauses should not have too many lines of code	Code Smell
Methods should not have too many	

Tags ▾

Search by name... 🔍

Parameters should be passed in the correct order

Analyze your code

Code Smell

Major ?

When the names of parameters in a method call match the names of the method arguments, it contributes to clearer, more readable code. However, when the names match, but are passed in a different order than the method arguments, it indicates a mistake in the parameter order which will likely lead to unexpected results.

Noncompliant Code Example

```
public double divide(int divisor, int dividend) {
    return divisor/dividend;
}

public void doTheThing() {
    int divisor = 15;
    int dividend = 5;

    double result = divide(dividend, divisor); // Noncompliant
    //...
}
```

Compliant Solution





```
public double divide(int divisor, int dividend) {
    return divisor/dividend;
}

public void doTheThing() {
    int divisor = 15;
    int dividend = 5;

    double result = divide(divisor, dividend);
    //...
}
```

Available In:

sonarlint | sonarcloud | sonarqube

<div>Methods should not have too many return statements</div> <div> Code Smell</div>
<div>Magic numbers should not be used</div> <div> Code Smell</div>
<div>Files should not have too many lines of code</div> <div> Code Smell</div>
<div>Lines should not be too long</div> <div> Code Smell</div>
<div>JEE applications should not "getClassLoader"</div>