




 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... 🔍

Code Smell

The members of an interface or class declaration should appear in a pre-defined order

Code Smell

Code Smell

Abstract class names should comply with a naming convention

Code Smell

Code Smell

Strings literals should be placed on the left side when checking for equality

Code Smell

Code Smell

Files should contain an empty newline at the end

Code Smell

Code Smell

Source code should be indented consistently

Code Smell

Code Smell

A close curly brace should be located at the beginning of a line

Code Smell

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

Code Smell

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line

Code Smell

Code Smell

An open curly brace should be located at the beginning of a line

Code Smell

Code Smell

An open curly brace should be located at the end of a line

Code Smell

Code Smell

Tabulation characters should not be

Code Smell

Assignments should not be made from within sub-expressions

Analyze your code

Code Smell

Major

cwe cert suspicious

Assignments within sub-expressions are hard to spot and therefore make the code less readable. Ideally, sub-expressions should not have side-effects.

Noncompliant Code Example

```
if ((str = cont.substring(pos1, pos2)).isEmpty()) { // Noncompliant
    //...
}
```

Compliant Solution

```
str = cont.substring(pos1, pos2);
if (str.isEmpty()) {
    //...
}
```

Exceptions

Assignments in while statement conditions, and assignments enclosed in relational expressions are ignored.

```
BufferedReader br = new BufferedReader(/* ... */);
String line;
while ((line = br.readLine()) != null) {...}
```




Chained assignments, including compound assignments, are ignored.

```
int i = j = 0;
int k = (j += 1);
result = (bresult = new byte[len]);
```

See

- [MITRE, CWE-481](#) - Assigning instead of Comparing
- [CERT, EXP45-C](#) - Do not perform assignments in selection statements
- [CERT, EXP51-J](#) - Do not perform assignments in conditional expressions





Available In:

 |  | 

© 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](#)

https://rules.sonarsource.com/java/RSPEC-1121

1/2

<div>used</div> <div> Code Smell</div>
<div>Functions should not be defined with a variable number of arguments</div> <div> Code Smell</div>
<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>