


-  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...

Inner classes which do not reference their owning classes should be "static"

Code Smell

"deleteOnExit" should not be used

Code Smell

Public methods should not contain selector arguments

Code Smell

Java parser failure

Code Smell

Track uses of disallowed methods

Code Smell

Types should be used in lambdas

Code Smell

"java.time" classes should be used for dates and times

Code Smell

The names of methods with boolean return values should start with "is" or "has"

Code Smell

Files should contain only one top-level class or interface each

Code Smell

Classes should not have too many fields

Code Smell

The ternary operator should not be used

Code Smell

Standard functional interfaces should not be redefined

Code Smell

Multiline blocks should be enclosed in curly braces

Analyze your code

Code SmellMajor?cwe cert

Curly braces can be omitted from a one-line block, such as with an `if` statement or for loop, but doing so can be misleading and induce bugs.

This rule raises an issue when the whitespacing of the lines after a one line block indicates an intent to include those lines in the block, but the omission of curly braces means the lines will be unconditionally executed once.

Note that this rule considers tab characters to be equivalent to 1 space. If you mix spaces and tabs you will sometimes see issues in code which look fine in your editor but are confusing when you change the size of tabs.

Noncompliant Code Example

```
if (condition)
    firstActionInBlock();
    secondAction(); // Noncompliant; executed unconditionally
    thirdAction();

if (condition) firstActionInBlock(); secondAction(); // Non

if (condition) firstActionInBlock(); // Noncompliant
    secondAction(); // Executed unconditionally

if (condition); secondAction(); // Noncompliant; secondActi

String str = null;
for (int i = 0; i < array.length; i++)
    str = array[i];
    doTheThing(str); // Noncompliant; executed only on last a
```








Compliant Solution

```
if (condition) {
    firstActionInBlock();
    secondAction();
}
thirdAction();

String str = null;
for (int i = 0; i < array.length; i++) {
    str = array[i];
    doTheThing(str);
}
```

See

- [MITRE, CWE-483](#) - Incorrect Block Delimitation
- [CERT, EXP52-J](#) - Use braces for the body of an if, for, or while statement

 Code Smell	<div>Available In:</div> <div>  </div>
"NullPointerException" should not be caught	
 Code Smell	
"NullPointerException" should not be explicitly thrown	
 Code Smell	
Classes should not have too many methods	<div>© 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</div>
 Code Smell	
Methods should not have too many lines	