




 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

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

Local variable and method parameter names should comply with a naming convention

Analyze your code

Code Smell

Minor

convention

Shared naming conventions allow teams to collaborate effectively. This rule raises an issue when a local variable or function parameter name does not match the provided regular expression.

Noncompliant Code Example

With the default regular expression `^[a-z][a-zA-Z0-9]*$`:

```
public void doSomething(int my_param) {
    int LOCAL;
    ...
}
```

Compliant Solution

```
public void doSomething(int myParam) {
    int local;
    ...
}
```

Exceptions

Loop counters are ignored by this rule.





```
for (int i_1 = 0; i_1 < limit; i_1++) { // Compliant
    // ...
}
```

as well as one-character catch variables:

```
try {
    //...
} catch (Exception e) { // Compliant
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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