


-  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

Anonymous classes should not have too many lines

Code Smell

Code Smell

Public types, methods and fields (API) should be documented with Javadoc

Code Smell

Code Smell

Exception handlers should preserve the original exceptions

Code Smell

Code Smell

Checked exceptions should not be thrown

Code Smell

Code Smell

Public methods should throw at most one checked exception

Code Smell

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Code Smell

Methods should not have too many return statements

Code Smell

Code Smell

Magic numbers should not be used

Code Smell

Code Smell

Files should not have too many lines of code

Code Smell

Code Smell

Lines should not be too long

Code Smell

Bug

JEE applications should not "getClassLoader"

Code Smell

Math should not be performed on floats

"static" members should be accessed statically

Analyze your code

Code Smell

Major

Quick Fix

pitfall

While it is *possible* to access static members from a class instance, it's bad form, and considered by most to be misleading because it implies to the readers of your code that there's an instance of the member per class instance.

Noncompliant Code Example

```
public class A {
    public static int counter = 0;
}

public class B {
    private A first = new A();
    private A second = new A();

    public void runUpTheCount() {
        first.counter++; // Noncompliant
        second.counter++; // Noncompliant. A.counter is now 2,
    }
}
```

Compliant Solution

```
public class A {
    public static int counter = 0;
}

public class B {
    private A first = new A();
    private A second = new A();

    public void runUpTheCount() {
        A.counter++; // Compliant
        A.counter++; // Compliant
    }
}
```






Available In:

sonarlint | sonarcloud | sonarqube

© 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-2209

1/2

 Bug
"equals" methods should be symmetric and work for subclasses  Bug
Literal suffixes should be upper case  Code Smell
Unicode-aware versions of character classes should be preferred  Code Smell
Use Java 12 "switch" expression  Code Smell