




 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... 🔍


Non-public methods should not be "@Transactional"

 Bug


Servlets should not have mutable instance fields

 Bug


"toString()" and "clone()" methods should not return null

 Bug


".equals()" should not be used to test the values of "Atomic" classes

 Bug


Return values from functions without side effects should not be ignored

 Bug


Child class methods named for parent class methods should be overrides

 Bug


Inappropriate "Collection" calls should not be made

 Bug


Silly equality checks should not be made

 Bug


Dissimilar primitive wrappers should not be used with the ternary operator without explicit casting

 Bug


"InterruptedException" should not be ignored

 Bug

Classes extending java.lang.Thread should override the "run" method

 Bug

"Double.longBitsToDouble" should not

 Bug

"static" base class members should not be accessed via derived types

Analyze your code

Code SmellCriticalQuick Fixconfusing

In the interest of code clarity, static members of a base class should never be accessed using a derived type's name. Doing so is confusing and could create the illusion that two different static members exist.

Noncompliant Code Example

```
class Parent {
    public static int counter;
}

class Child extends Parent {
    public Child() {
        Child.counter++; // Noncompliant
    }
}
```

Compliant Solution

```
class Parent {
    public static int counter;
}

class Child extends Parent {
    public Child() {
        Parent.counter++;
    }
}
```





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-3252

1/2

be used for "int"
 Bug
Values should not be uselessly incremented
 Bug
Silly String operations should not be made
 Bug
Non-serializable classes should not be written
 Bug
"hashCode" and "toString" should not