




 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

Tags ▾

Search by name... 🔍

⚙️ Code Smell

Only static class initializers should be used

⚙️ Code Smell

⚙️ Code Smell

Empty arrays and collections should be returned instead of null

⚙️ Code Smell

⚙️ Code Smell

"@Override" should be used on overriding and implementing methods

⚙️ Code Smell

⚙️ Code Smell

Enumeration should not be implemented

⚙️ Code Smell

⚙️ Code Smell

Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used

⚙️ Code Smell

⚙️ Code Smell

Unused "private" methods should be removed

⚙️ Code Smell

⚙️ Code Smell

Try-catch blocks should not be nested

⚙️ Code Smell

⚙️ Code Smell

Track uses of "FIXME" tags

⚙️ Code Smell

⚙️ Code Smell

Deprecated elements should have both the annotation and the Javadoc tag

⚙️ Code Smell

⚙️ Code Smell

Assignments should not be made from within sub-expressions

⚙️ Code Smell

⚙️ Code Smell

Generic exceptions should never be thrown

⚙️ Code Smell

Assignment of lazy-initialized members should be the last step with double-checked locking

Analyze your code

🐞 Bug

🔴 Major ?

🔗 multi-threading cert

Double-checked locking can be used for lazy initialization of `volatile` fields, but only if field assignment is the last step in the `synchronized` block. Otherwise you run the risk of threads accessing a half-initialized object.

Noncompliant Code Example

```
public class MyClass {

    private volatile List<String> strings;

    public List<String> getStrings() {
        if (strings == null) { // check#1
            synchronized(MyClass.class) {
                if (strings == null) {
                    strings = new ArrayList<>(); // Noncompliant
                    strings.add("Hello"); //When threadA gets here, t
                    strings.add("World");
                }
            }
        }
        return strings;
    }
}
```

Compliant Solution

```
public class MyClass {

    private volatile List<String> strings;





    public List<String> getStrings() {
        if (strings == null) { // check#1
            synchronized(MyClass.class) {
                if (strings == null) {
                    List<String> tmpList = new ArrayList<>();
                    tmpList.add("Hello");
                    tmpList.add("World");
                    strings = tmpList;
                }
            }
        }
        return strings;
    }
}
```

See

- [CERT, LCK10-J](#) - Use a correct form of the double-checked locking idiom

https://rules.sonarsource.com/java/RSPEC-3064

1/2

Labels should not be used  Code Smell
Utility classes should not have public constructors  Code Smell
Local variables should not shadow class fields  Code Smell
Redundant pairs of parentheses should be removed  Code Smell

See Also

- {rule:java:S2168} - Double-checked locking should not be used

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](#)