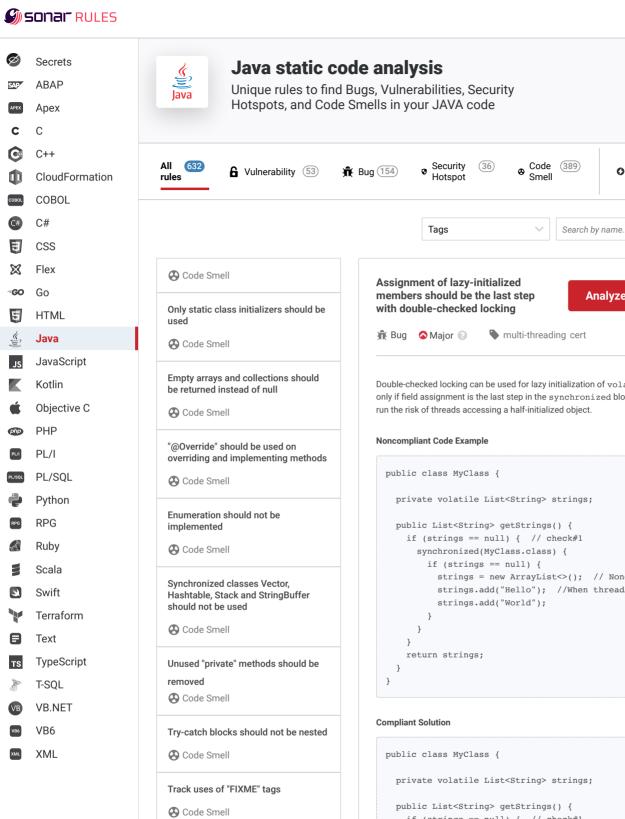
Products ✓



Deprecated elements should have both the annotation and the Javadoc

Assignments should not be made from within sub-expressions

Generic exceptions should never be

Code Smell

Code Smell

Code Smell

thrown

Analyze your code

O Quick 42 Fix

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

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

```
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;
```

• CERT, LCK10-J. - Use a correct form of the double-checked locking idiom

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