




 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... 🔍

synchronized in pairs

Bug

Non-thread-safe fields should not be static

Bug

"null" should not be used with "Optional"

Bug

Unary prefix operators should not be repeated

Bug

"==" should not be used instead of "!="

Bug

"read" and "readLine" return values should be used

Bug

Inappropriate regular expressions should not be used

Bug

Conditionally executed code should be reachable

Bug

"notifyAll" should be used

Bug

Blocks should be synchronized on "private final" fields

Bug

Non-serializable objects should not be stored in "HttpSession" objects

Bug

"wait", "notify" and "notifyAll" should only be called when a lock is obviously held on an object

Bug

Derived exceptions should not hide their parents' catch blocks

Analyze your code

Code Smell

Critical

The catch block of a checked exception "E" may be hidden because the corresponding try block only throws exceptions derived from E.

These derived exceptions are handled in dedicated catch blocks prior to the catch block of the base exception E.

The catch block of E is unreachable and should be considered dead code. It should be removed, or the entire try-catch structure should be refactored.

It is also possible that a single exception type in a multi-catch block may be hidden while the catch block itself is still reachable. In that case it is enough to only remove the hidden exception type or to replace it with another type.

Noncompliant Code Example

```
public class HiddenCatchBlock {

    public static class CustomException extends Exception {
    }

    public static class CustomDerivedException extends CustomException {
    }

    public static void main(String[] args) {
        try {
            throwCustomDerivedException();
        } catch (CustomDerivedException e) {
            // ...
        } catch (CustomException e) { // Noncompliant; this code
            // ...
        }
    }

    private static void throwCustomDerivedException() throws CustomException {
        throw new CustomDerivedException();
    }
}
```

Compliant Solution

```
public class HiddenCatchBlock {

    public static class CustomException extends Exception {
    }

    public static class CustomDerivedException extends CustomException {
    }

    public static void main(String[] args) {
        try {
```

https://rules.sonarsource.com/java/RSPEC-4970

1/2

Null pointers should not be dereferenced

Bug

Loop conditions should be true at least once

Bug

A "for" loop update clause should move the counter in the right direction

Bug

Non-public methods should not be "@Transactional"

Bug

```
throwCustomDerivedException();
} catch(CustomDerivedException e) { // Compliant; try-ca
//...
}
}
}
```

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](#)