




 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... 🔍

"URL.hashCode" and "URL.equals" should be avoided

Code Smell

Two branches in a conditional structure should not have exactly the same implementation

Code Smell

Unused assignments should be removed

Code Smell

"Object.wait(...)" should never be called on objects that implement "java.util.concurrent.locks.Condition"

Code Smell

A field should not duplicate the name of its containing class

Code Smell

JUnit4 @Ignored and JUnit5 @Disabled annotations should be used to disable tests and should provide a rationale

Code Smell

Anonymous inner classes containing only one method should become lambdas

Code Smell

"switch" statements should not have too many "case" clauses

Code Smell

"for" loop stop conditions should be invariant

Code Smell

Sections of code should not be commented out

Code Smell

Non-constructor methods should not be commented out

Code Smell

Overrides should match their parent class methods in synchronization

Analyze your code

BugMajormulti-threading cert

When @Overrides of synchronized methods are not themselves synchronized, the result can be improper synchronization as callers rely on the thread-safety promised by the parent class.

Noncompliant Code Example

```
public class Parent {

    synchronized void foo() {
        //...
    }
}

public class Child extends Parent {

    @Override
    public void foo () { // Noncompliant
        // ...
        super.foo();
    }
}
```

Compliant Solution

```
public class Parent {

    synchronized void foo() {
        //...
    }
}


public class Child extends Parent {

    @Override
    synchronized void foo () {
        // ...
        super.foo();
    }
}
```

See

- [CERT, TSM00-J](#) - Do not override thread-safe methods with methods that are not thread-safe

Available In:



https://rules.sonarsource.com/java/RSPEC-3551

1/2

have the same name as the enclosing class

 Code Smell

Exception types should not be tested using "instanceof" in catch blocks

 Code Smell

Classes from "sun.\*" packages should not be used

 Code Smell

Throwable and Error should not be caught

 Code Smell

 |  | 

© 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](#)