




 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... 🔍

"Externalizable" classes should have no-arguments constructors

Bug

Classes should not be compared by name

Bug

Related "if/else if" statements should not have the same condition

Bug

Synchronization should not be done on instances of value-based classes

Bug

"Iterator.hasNext()" should not call "Iterator.next()"

Bug

Identical expressions should not be used on both sides of a binary operator

Bug

Loops with at most one iteration should be refactored

Bug

Variables should not be self-assigned

Bug

"StringBuilder" and "StringBuffer" should not be instantiated with a character

Bug

Methods should not be named "toString", "hashCode" or "equal"

Bug

"Thread.run()" should not be called directly

Bug

Try-with-resources should be used

Analyze your code

Code Smell

Critical

java8 cert pitfall

Java 7 introduced the try-with-resources statement, which guarantees that the resource in question will be closed. Since the new syntax is closer to bullet-proof, it should be preferred over the older try/catch/finally version.

This rule checks that close-able resources are opened in a try-with-resources statement.

Note that this rule is automatically disabled when the project's sonar.java.source is lower than 7.

Noncompliant Code Example

```
FileReader fr = null;
BufferedReader br = null;
try {
    fr = new FileReader(fileName);
    br = new BufferedReader(fr);
    return br.readLine();
} catch (...) {}
} finally {
    if (br != null) {
        try {
            br.close();
        } catch (IOException e){...}
    }
    if (fr != null ) {
        try {
            br.close();
        } catch (IOException e){...}
    }
}
```

Compliant Solution








```
try (
    FileReader fr = new FileReader(fileName);
    BufferedReader br = new BufferedReader(fr)
) {
    return br.readLine();
}
catch (...) {}
```

or

```
try (BufferedReader br =
    new BufferedReader(new FileReader(fileName))) { // n
    return br.readLine();
}
catch (...) {}
```

https://rules.sonarsource.com/java/RSPEC-2093

1/2

<p>"equals" method overrides should accept "Object" parameters</p> <p> Bug</p>	<div><p>See</p><ul style="list-style-type: none">• CERT, ERR54-J. - Use a try-with-resources statement to safely handle closeable resources<p>Available In:</p><p>sonarlint  sonarcloud  sonarqube </p></div>
<p>The Object.finalize() method should not be called</p> <p> Bug</p>	
<p>Enabling file access for WebViews is security-sensitive</p> <p> Security Hotspot</p>	
<p>Enabling JavaScript support for WebViews is security-sensitive</p> <p> Security Hotspot</p>	

© 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](#)