

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

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... 🔍

without a cryptographic solution is security-sensitive

Security Hotspot

Using unencrypted databases in mobile applications is security-sensitive

Security Hotspot

Authorizing non-authenticated users to use keys in the Android KeyStore is security-sensitive

Security Hotspot

Allowing user enumeration is security-sensitive

Security Hotspot

Allowing requests with excessive content length is security-sensitive

Security Hotspot

Disabling auto-escaping in template engines is security-sensitive

Security Hotspot

Allowing deserialization of LDAP objects is security-sensitive

Security Hotspot

Setting loose POSIX file permissions is security-sensitive

Security Hotspot

Formatting SQL queries is security-sensitive

Security Hotspot

Deprecated annotations should include explanations

Code Smell

Restricted Identifiers should not be used as Identifiers

Code Smell

Redundant constructors/methods

"Object.finalize()" should remain protected (versus public) when overriding

Analyze your code

Code SmellCritical?cwe cert

The contract of the `Object.finalize()` method is clear: only the Garbage Collector is supposed to call this method.

Making this method public is misleading, because it implies that any caller can use it.

Noncompliant Code Example

```
public class MyClass {

    @Override
    public void finalize() {    // Noncompliant
        /* ... */
    }
}
```

See

- MITRE, CWE-583 - finalize() Method Declared Public
- CERT, MET12-J. - Do not use finalizers





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](#)

https://rules.sonarsource.com/java/RSPEC-1174

1/2

<div>should be avoided in records</div> <div> Code Smell</div>
<div>Records should be used instead of ordinary classes when representing immutable data structure</div> <div> Code Smell</div>
<div>"Stream.toList()" method should be used instead of "collectors" when unmodifiable list needed</div> <div> Code Smell</div>
<div>Operator "instanceof" should be used instead of "A.class.isInstance()"</div> <div> Code Smell</div>