 sonar RULES

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... 🔍

Abstract class names should comply with a naming convention

Code Smell

Strings literals should be placed on the left side when checking for equality

Code Smell

Files should contain an empty newline at the end

Code Smell

Source code should be indented consistently

Code Smell

A close curly brace should be located at the beginning of a line

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line

Code Smell

An open curly brace should be located at the beginning of a line

Code Smell

An open curly brace should be located at the end of a line

Code Smell

Tabulation characters should not be used

Code Smell

Functions should not be defined with a variable number of arguments

Code Smell

"Serializable" classes should have a "serialVersionUID"

Analyze your code

Code Smell

Critical

serialization cert pitfall

A serialVersionUID field is strongly recommended in all Serializable classes. If you do not provide one, one will be calculated for you by the compiler. The danger in not explicitly choosing the value is that when the class changes, the compiler will generate an entirely new id, and you will be suddenly unable to deserialize (read from file) objects that were serialized with the previous version of the class.

serialVersionUID's should be declared with all of these modifiers: static final long.

Noncompliant Code Example

```
public class Raspberry extends Fruit // Noncompliant; no se
    implements Serializable {
    private String variety;

    public Raspberry(Season ripe, String variety) { ...}
    public void setVariety(String variety) {...}
    public String getVariety() {...}
}

public class Raspberry extends Fruit
    implements Serializable {
    private final int serialVersionUID = 1; // Noncompliant; n
```

Compliant Solution

```
public class Raspberry extends Fruit
    implements Serializable {
    private static final long serialVersionUID = 1;
    private String variety;

    public Raspberry(Season ripe, String variety) { ...}
    public void setVariety(String variety) {...}
    public String getVariety() {...}
}
```

Exceptions

Records, Swing and AWT classes, abstract classes, Throwable and its subclasses (Exceptions and Errors), and classes marked with @SuppressWarnings("serial") are ignored.

See

- [CERT, SER00-J](#). - Enable serialization compatibility during class evolution
- [Record Serialization](#) - Serialization of Records

https://rules.sonarsource.com/java/RSPEC-2057

1/2


Local-Variable Type Inference should be used

 Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

 Code Smell

Track uses of disallowed classes

 Code Smell

Track uses of "@SuppressWarnings" annotations

 Code Smell

Available In:
 |  | 

© 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](#)