

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...

Subject to normalization should use the CANON\_EQ flag

Code Smell

Regular expressions should not be too complicated

Code Smell

JUnit assertTrue/assertFalse should be simplified to the corresponding dedicated assertion

Code Smell

Only one method invocation is expected when testing runtime exceptions

Code Smell

Exception testing via JUnit ExpectedException rule should not be mixed with other assertions

Code Smell

"@Deprecated" code marked for removal should never be used

Code Smell

Vararg method arguments should not be confusing

Code Smell

Whitespace for text block indent should be consistent

Code Smell

'List.remove()' should not be used in ascending 'for' loops

Code Smell

Collection constructors should not be used as java.util.function.Function

Code Smell

"else" statements should be clearly matched with an "if"

Code Smell

Equals method should be overridden in records containing array fields

Analyze your code

BugMajorjava16

In records, the default behavior of the equals() method is to check the equality by field values. This works well for primitive fields or fields, whose type overrides equals(), but this behavior doesn't work as expected for array fields.

By default, array fields are compared by their reference, and overriding equals() is highly appreciated to achieve the deep equality check. The same strategy applies to hashCode() and toString() methods.

This rule reports an issue if a record class has an array field and is not overriding equals(), hashCode() or toString() methods.

Noncompliant Code Example

```
record Person(String[] names, int age) {} // Noncompliant
```

Compliant Solution

```
record Person(String[] names, int age) {
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
            Person person = (Person) o;
        return age == person.age && Arrays.equals(names, per
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(age);
        result = 31 * result + Arrays.hashCode(names);
        return result;
    }

    @Override
    public String toString() {
        return "Person{" +
            "names=" + Arrays.toString(names) +
            ", age=" + age +
            '}';
    }
}
```

See

- [Records specification](#)

Available In:

https://rules.sonarsource.com/java/RSPEC-6218

1/2

"Class.forName()" should not load JDBC 4.0+ drivers

 Code Smell

Java features should be preferred to

Guava

 Code Smell

Nullness of parameters should be guaranteed

 Code Smell

"Integer.toHexString" should not be used to build hexadecimal strings

 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](#)