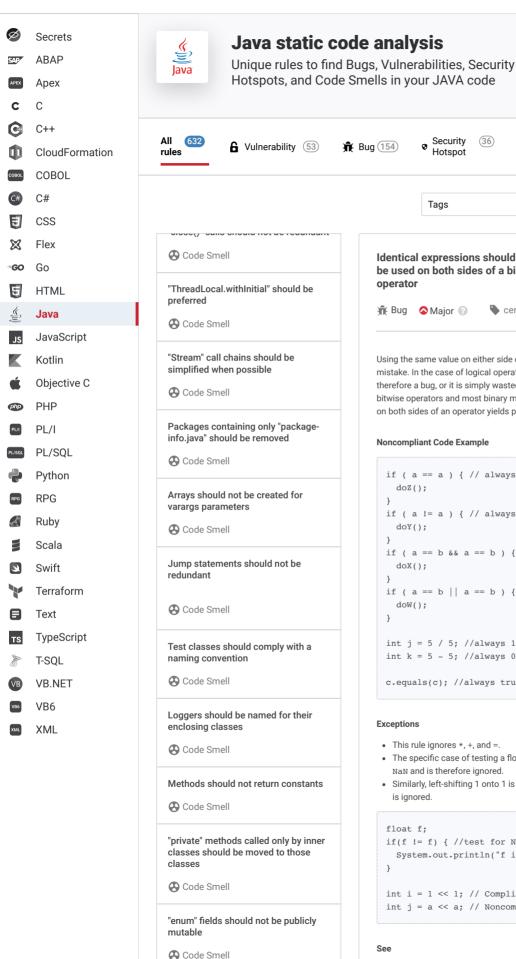


Products >



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

Security

Hotspot

Tags

👬 Bug 🔷 Major 🕝

(36)

Analyze your code

O Quick 42

Q

(389)

Search by name.

Using the same value on either side of a binary operator is almost always a mistake. In the case of logical operators, it is either a copy/paste error and therefore a bug, or it is simply wasted code, and should be simplified. In the case of bitwise operators and most binary mathematical operators, having the same value on both sides of an operator yields predictable results, and should be simplified.

Noncompliant Code Example

```
if ( a == a ) { // always true
  doZ();
if ( a != a ) { // always false
  doY();
if ( a == b \&\& a == b ) { // if the first one is true, the s
  doX();
if ( a == b \mid \mid a == b ) { // if the first one is true, the s
  doW();
int j = 5 / 5; //always 1
int k = 5 - 5; //always 0
c.equals(c); //always true
```

Exceptions

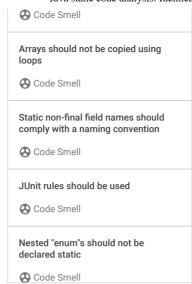
- This rule ignores *, +, and =.
- The specific case of testing a floating point value against itself is a valid test for Nan and is therefore ignored.
- . Similarly, left-shifting 1 onto 1 is common in the construction of bit masks, and is ignored.

```
float f;
if(f != f) { //test for NaN value
  System.out.println("f is NaN");
int i = 1 \ll 1; // Compliant
int j = a << a; // Noncompliant</pre>
```

- CERT, MSC12-C. Detect and remove code that has no effect or is never executed
- {rule:java:S1656} Implements a check on =.

Abstract methods should not be

redundant





© 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