




 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... 🔍


 Bug


"equals()" should not be used to test the values of "Atomic" classes







Return values from functions without side effects should not be ignored







Child class methods named for parent class methods should be overrides







Inappropriate "Collection" calls should not be made







Silly equality checks should not be made





Dissimilar primitive wrappers should not be used with the ternary operator without explicit casting





"InterruptedException" should not be ignored

Classes extending java.lang.Thread should override the "run" method


"Double.longBitsToDouble" should not be used for "int"


Values should not be uselessly incremented


Silly String operations should not be made

"indexOf" checks should not be for positive numbers

Analyze your code

 Code Smell

 Critical

 suspicious

Most checks against an indexOf value compare it with -1 because 0 is a valid index. Any checks which look for values >0 ignore the first element, which is likely a bug. If the intent is merely to check inclusion of a value in a String or a List, consider using the contains method instead.

This rule raises an issue when an indexOf value retrieved either from a String or a List is tested against >0.

Noncompliant Code Example

```
String color = "blue";
String name = "ishmael";

List<String> strings = new ArrayList<String> ();
strings.add(color);
strings.add(name);

if (strings.indexOf(color) > 0) { // Noncompliant
    // ...
}
if (name.indexOf("ish") > 0) { // Noncompliant
    // ...
}
if (name.indexOf("ae") > 0) { // Noncompliant
    // ...
}
```


Compliant Solution

```
String color = "blue";
String name = "ishmael";


List<String> strings = new ArrayList<String> ();
strings.add(color);
strings.add(name);

if (strings.indexOf(color) > -1) {
    // ...
}
if (name.indexOf("ish") >= 0) {
    // ...
}
if (name.contains("ae") {
    // ...
}
```


Available In:

 Bug


Non-serializable classes should not be written

 Bug



"hashCode" and "toString" should not be called on array instances

 Bug

Collections should not be passed as arguments to their own methods

 Bug

"BigDecimal(double)" should not be used

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](#)