
































-  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 rules

632
- Vulnerability

53
- Bug

154
- Security Hotspot













36
- Code Smell

389
- Quick Fix

42

Tags ▾

Search by name... 🔍

 Bug
Classes extending java.lang.Thread should override the "run" method
 Bug
"Double.longBitsToDouble" should not be used for "int"
 Bug
Values should not be uselessly incremented
 Bug
Silly String operations should not be made
 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
 Bug
Invalid "Date" values should not be used
 Bug
Reflection should not be used to check non-runtime annotations
 Bug
Custom serialization method signatures should meet requirements
 Bug

Null should not be returned from a "Boolean" method

Analyze your code

-  Code Smell
-  Critical
-  cwe cert pitfall

While `null` is technically a valid `Boolean` value, that fact, and the distinction between `Boolean` and `boolean` is easy to forget. So returning `null` from a `Boolean` method is likely to cause problems with callers' code.





Noncompliant Code Example

```
public Boolean isUsable() {  
    // ...  
    return null; // Noncompliant  
}
```

- See
- MITRE, CWE-476 - NULL Pointer Dereference
 - CERT, EXP01-J. - Do not use a null in a case where an object is required

Available In:

sonarlint  | sonarcloud  | sonarqube 

<div><div>"Externalizable" classes should have no-arguments constructors</div><div> Bug</div></div>
<div><div>Classes should not be compared by name</div><div> Bug</div></div>
<div><div>Related "if/else if" statements should not have the same condition</div><div> Bug</div></div>
<div><div>Synchronization should not be done on instances of value-based classes</div><div> Bug</div></div>