
































-  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... 

"BigDecimal(double)" should not be used



Invalid "Date" values should not be used



Reflection should not be used to check non-runtime annotations



Custom serialization method signatures should meet requirements



"Externalizable" classes should have no-arguments constructors



Classes should not be compared by name



Related "if/else if" statements should not have the same condition



Synchronization should not be done on instances of value-based classes



"Iterator.hasNext()" should not call "Iterator.next()"



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



Loops with at most one iteration should be refactored



JUnit assertions should not be used in "run" methods

Analyze your code

-  Code Smell
-  Critical
-  junit tests




JUnit assertions should not be made from the run method of a Runnable, because failed assertions result in `AssertionErrors` being thrown. If the error is thrown from a thread other than the one that ran the test, the thread will exit but the test won't fail.

Noncompliant Code Example

```
public void run() {  
    // ...  
    Assert.assertEquals(expected, actual); // Noncompliant  
}
```

Available In:
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](#)

<div>Variables should not be self-assigned</div> <div> Bug</div>
<div>"StringBuilder" and "StringBuffer" should not be instantiated with a character</div> <div> Bug</div>
<div>Methods should not be named "toString", "hashCode" or "equal"</div> <div> Bug</div>
<div>"Thread.run()" should not be called directly</div> <div> Bug</div>