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

Inappropriate "Collection" calls should not be made
Bug

Silly equality checks should not be made
Bug

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

"InterruptedException" should not be ignored
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

## Whitespace and control characters in literals should be explicit

Analyze your code

Code Smell | Critical | pitfall

Non-encoded control characters and whitespace characters are often injected in the source code because of a bad manipulation. They are either invisible or difficult to recognize, which can result in bugs when the string is not what the developer expects. If you actually need to use a control character use their encoded version (ex: ASCII `\n,\t,…` or Unicode `U+000D, U+0009,…`).

This rule raises an issue when the following characters are seen in a literal string:

- ASCII control character. (character index < 32 or = 127)
- Unicode whitespace characters.
- Unicode C0 control characters
- Unicode characters `U+200B, U+200C, U+200D, U+2060, U+FEFF, U+2028, U+2029`

No issue will be raised on the simple space character. Unicode `U+0020`, ASCII 32.

### Noncompliant Code Example

```
String tabInside = "A	B";  // Noncompliant, contains a tab
String zeroWidthSpaceInside = "foobar"; // Noncompliant, it
char tab = '	';
```

### Compliant Solution

```
String tabInside = "A\tB";  // Compliant, uses escaped value
String zeroWidthSpaceInside = "foo\u200Bbar";  // Compliant,
char tab = '\t';
```

Available In:

sonarlint | sonarcloud | sonarqube

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