sonar RULES Products

- 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...

Abstract class names should comply with a naming convention
Code Smell

Strings literals should be placed on the left side when checking for equality
Code Smell

Files should contain an empty newline at the end
Code Smell

Source code should be indented consistently
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

An open curly brace should be located at the end of a line
Code Smell

Tabulation characters should not be used
Code Smell

Functions should not be defined with a variable number of arguments
Code Smell

## Equality operators should not be used in "for" loop termination conditions

Analyze your code

Code Smell Critical cwe cert suspicious

Testing `for` loop termination using an equality operator (`==` and `!=`) is dangerous, because it could set up an infinite loop. Using a broader relational operator instead casts a wider net, and makes it harder (but not impossible) to accidentally write an infinite loop.

### Noncompliant Code Example

```
for (int i = 1; i != 10; i += 2)  // Noncompliant. Infinite;
{
  //...
}
```

### Compliant Solution

```
for (int i = 1; i <= 10; i += 2)  // Compliant
{
  //...
}
```

### Exceptions

Equality operators are ignored if the loop counter is not modified within the body of the loop and either:

- starts below the ending value and is incremented by 1 on each iteration.
- starts above the ending value and is decremented by 1 on each iteration.

Equality operators are also ignored when the test is against `null`.

```
for (int i = 0; arr[i] != null; i++) {
  // ...
}

for (int i = 0; (item = arr[i]) != null; i++) {
  // ...
}
```

### See

- MITRE, CWE-835 - Loop with Unreachable Exit Condition ('Infinite Loop')
- CERT, MSC21-C. - Use robust loop termination conditions

Available In:
sonarlint sonarcloud sonarqube

Local-Variable Type Inference should be used

Code Smell

Migrate your tests from JUnit4 to the new JUnit5 annotations

Code Smell

Track uses of disallowed classes

Code Smell

Track uses of "@SuppressWarnings" annotations

Code Smell