




 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... 🔍

Enum values should be compared with "=="

Code Smell

Spring components should use constructor injection

Code Smell

Regex patterns should not be created needlessly

Code Smell

Track uses of disallowed constructors

Code Smell

Java 8's "Files.exists" should not be used

Code Smell

"Optional" should not be used for parameters

Code Smell

Tests should be kept in a dedicated source directory

Code Smell

"this" should not be exposed from constructors

Code Smell

Classes should not have too many "static" imports

Code Smell

Escaped Unicode characters should not be used

Code Smell

Inner classes should not have too many lines of code

Code Smell

Inner classes which do not reference their owning classes should be "static"

Code Smell

### Ternary operators should not be nested

Analyze your code

Code Smell

Major ?

confusing

Just because you *can* do something, doesn't mean you should, and that's the case with nested ternary operations. Nesting ternary operators results in the kind of code that may seem clear as day when you write it, but six months later will leave maintainers (or worse - future you) scratching their heads and cursing.

Instead, err on the side of clarity, and use another line to express the nested operation as a separate statement.

#### Noncompliant Code Example

```
public String getReadableStatus(Job j) {
    return j.isRunning() ? "Running" : j.hasErrors() ? "Failed"
}
```

#### Compliant Solution

```
public String getReadableStatus(Job j) {
    if (j.isRunning()) {
        return "Running";
    }
    return j.hasErrors() ? "Failed" : "Succeeded";
}
```

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](#)

https://rules.sonarsource.com/java/RSPEC-3358

1/2

 Code Smell
<b>"deleteOnExit" should not be used</b>  Code Smell
<b>Public methods should not contain selector arguments</b>  Code Smell
<b>Java parser failure</b>  Code Smell
<b>Track uses of disallowed methods</b>  Code Smell