


-  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... 🔍

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

## Enum values should be compared with "=="

Analyze your code

Code SmellMajor ?

Testing equality of an enum value with `equals` is perfectly valid because an enum is an Object and every Java developer knows `=="` should not be used to compare the content of an Object. At the same time, using `=="` on enums:

- provides the same expected comparison (content) as `equals`
- is more null-safe than `equals()`
- provides compile-time (static) checking rather than runtime checking

For these reasons, use of `=="` should be preferred to `equals`.

### Noncompliant Code Example

```
public enum Fruit {
    APPLE, BANANA, GRAPE
}

public enum Cake {
    LEMON_TART, CHEESE_CAKE
}

public boolean isFruitGrape(Fruit candidateFruit) {
    return candidateFruit.equals(Fruit.GRAPE); // Noncompliant
}

public boolean isFruitGrape(Cake candidateFruit) {
    return candidateFruit.equals(Fruit.GRAPE); // Noncompliant
}
```

### Compliant Solution

```
public boolean isFruitGrape(Fruit candidateFruit) {
    return candidateFruit == Fruit.GRAPE; // Compliant; there
}

public boolean isFruitGrape(Cake candidateFruit) {
    return candidateFruit == Fruit.GRAPE; // Compliant; compil
}
```

See

- [Use == \(or !=\) to Compare Java Enums](#)

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-4551

1/2

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>

© 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](#)