


-  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

"StandardCharsets" constants should be preferred

Analyze your code

Code Smell

Minor

java7 clumsy

JDK7 introduced the class `java.nio.charset.StandardCharsets`. It provides constants for all charsets that are guaranteed to be available on every implementation of the Java platform.

- ISO_8859_1
- US_ASCII
- UTF_16
- UTF_16BE
- UTF_16LE
- UTF_8

These constants should be preferred to:

- the use of a String such as "UTF-8" which has the drawback of requiring the catch/throw of an `UnsupportedEncodingException` that will never actually happen
- the use of Guava's `Charsets` class, which has been obsolete since JDK7

Noncompliant Code Example

```
try {
    byte[] bytes = string.getBytes("UTF-8"); // Noncompliant;
} catch (UnsupportedEncodingException e) {
    throw new AssertionError(e);
}
// ...
byte[] bytes = string.getBytes(Charsets.UTF_8); // Noncompliant
```

Compliant Solution

```
byte[] bytes = string.getBytes(StandardCharsets.UTF_8)
```





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-4719

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>