Java static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your JAVA code

Test classes should comply with a naming convention
Code Smell

Loggers should be named for their enclosing classes
Code Smell

Methods should not return constants
Code Smell

"private" methods called only by inner classes should be moved to those classes
Code Smell

"enum" fields should not be publicly mutable
Code Smell

Abstract methods should not be redundant
Code Smell

Arrays should not be copied using loops
Code Smell

Static non-final field names should comply with a naming convention
Code Smell

JUnit rules should be used
Code Smell

Nested "enum"s should not be declared static
Code Smell

"catch" clauses should do more than rethrow
Code Smell

Mutable fields should not be "public static"

"StringBuilder" and "StringBuffer" should not be instantiated with a character

Analyze your code

Bug Major pitfall

Instantiating a `StringBuilder` or a `StringBuffer` with a character is misleading because most Java developers would expect the character to be the initial value of the `StringBuffer`.

What actually happens is that the int representation of the character is used to determine the initial size of the `StringBuffer`.

Noncompliant Code Example

```
StringBuffer foo = new StringBuffer('x');   //equivalent to
```

Compliant Solution

```
StringBuffer foo = new StringBuffer("x");
```

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

Code Smell

The diamond operator ("<>") should be used

Code Smell

"finalize" should not set fields to "null"

Code Smell

Subclasses that add fields should override "equals"

Code Smell

Catches should be combined

Code Smell