












































-  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

 Code Smell
Return of boolean expressions should not be wrapped into an "if-then-else" statement
 Code Smell
Boolean literals should not be redundant
 Code Smell
Modifiers should be declared in the correct order
 Code Smell
Empty statements should be removed
 Code Smell
Class variable fields should not have public accessibility
 Code Smell
URIs should not be hardcoded
 Code Smell
Class names should comply with a naming convention
 Code Smell
Method names should comply with a naming convention
 Code Smell
Comma-separated labels should be used in Switch with colon case
 Code Smell
JUnit5 test classes and methods should have default package visibility
 Code Smell
Track uses of "TODO" tags
 Code Smell
Deprecated code should be removed

Tags ▾

Search by name... 

Regexes containing characters subject to normalization should use the CANON_EQ flag

Analyze your code

 Code Smell

 Major 

 regex

Characters like 'é' can be expressed either as a single code point or as a cluster of the letter 'e' and a combining accent mark. Without the CANON_EQ flag, a regex will only match a string in which the characters are expressed in the same way.

Noncompliant Code Example

```
String s = "e\u0300";
Pattern p = Pattern.compile("é|ë|è"); // Noncompliant
System.out.println(p.matcher(s).replaceAll("e")); // print 'e'
```





Compliant Solution

```
String s = "e\u0300";
Pattern p = Pattern.compile("é|ë|è", Pattern.CANON_EQ);
System.out.println(p.matcher(s).replaceAll("e")); // print 'e'
```

Available In:

 |  | 

© 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
Annotated Mockito objects should be initialized  Bug
Custom resources should be closed  Bug
Threads should not be started in constructors  Code Smell
"main" should not "throw" anything