




 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

Tags ▾

Search by name... 🔍

Bug

Non-thread-safe fields should not be static

Bug

"null" should not be used with "Optional"

Bug

Unary prefix operators should not be repeated

Bug

"+=" should not be used instead of "+="

Bug

"read" and "readLine" return values should be used

Bug

Inappropriate regular expressions should not be used

Bug

Conditionally executed code should be reachable

Bug

"notifyAll" should be used

Bug

Blocks should be synchronized on "private final" fields

Bug

Non-serializable objects should not be stored in "HttpSession" objects

Bug

"wait", "notify" and "notifyAll" should only be called when a lock is obviously held on an object

Bug

String offset-based methods should be preferred for finding substrings from offsets

Analyze your code

Code Smell

Critical

performance

Looking for a given substring starting from a specified offset can be achieved by such code: `str.substring(beginIndex).indexOf(char1)`. This works well, but it creates a new `String` for each call to the `substring` method. When this is done in a loop, a lot of `Strings` are created for nothing, which can lead to performance problems if `str` is large.

To avoid performance problems, `String.substring(beginIndex)` should not be chained with the following methods:

- `indexOf(int ch)`
- `indexOf(String str)`
- `lastIndexOf(int ch)`
- `lastIndexOf(String str)`
- `startsWith(String prefix)`

For each of these methods, another method with an additional parameter is available to specify an offset.

Using these methods will avoid the creation of additional `String` instances. For `indexOf` methods, adjust the returned value by subtracting the substring index parameter to obtain the same result.

Noncompliant Code Example

```
str.substring(beginIndex).indexOf(char1); // Noncompliant; a
```

Compliant Solution





```
str.indexOf(char1, beginIndex) - beginIndex; // index for ch
```

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-4635

1/2

<div>Null pointers should not be dereferenced</div> <div> Bug</div>
<div>Loop conditions should be true at least once</div> <div> Bug</div>
<div>A "for" loop update clause should move the counter in the right direction</div> <div> Bug</div>
<div>Non-public methods should not be "@Transactional"</div> <div> Bug</div>