




 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... 🔍

⚙️ Code Smell

"@Override" should be used on overriding and implementing methods

⚙️ Code Smell

⚙️ Code Smell

Enumeration should not be implemented

⚙️ Code Smell

⚙️ Code Smell

Synchronized classes Vector, Hashtable, Stack and StringBuffer should not be used

⚙️ Code Smell

⚙️ Code Smell

Unused "private" methods should be removed

⚙️ Code Smell

⚙️ Code Smell

Try-catch blocks should not be nested

⚙️ Code Smell

⚙️ Code Smell

Track uses of "FIXME" tags

⚙️ Code Smell

⚙️ Code Smell

Deprecated elements should have both the annotation and the Javadoc tag

⚙️ Code Smell

⚙️ Code Smell

Assignments should not be made from within sub-expressions

⚙️ Code Smell

⚙️ Code Smell

Generic exceptions should never be thrown

⚙️ Code Smell

⚙️ Code Smell

Labels should not be used

⚙️ Code Smell

⚙️ Code Smell

Utility classes should not have public constructors

⚙️ Code Smell

### "String" calls should not go beyond their bounds

Analyze your code

🐞 Bug

🔴 Major

?

Just as you can't cut something into three halves, you can't grab a substring that starts or ends outside the original String's bounds, you can't use substring to get a reversed portion of a String, and you can't get the charAt a value that's before the String starts or after it ends.

This rule detects when negative literal or String::length is passed as an argument to the String::substring, String::charAt and related methods.

#### Noncompliant Code Example

```
String speech = "Now is the time for all good people to come

String substr1 = speech.substring(-1, speech.length()); //
String substr2 = speech.substring(speech.length(), 0); // No
char ch = speech.charAt(speech.length()); // Noncompliant
```

#### Compliant Solution

```
String speech = "Now is the time for all good people to come

String substr1 = speech; // Closest correct values to origin
String substr2 = new StringBuilder(speech).reverse().toString();
char ch = speech.charAt(speech.length()-1);
```

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-3039

1/2

<div>Local variables should not shadow class fields</div> <div> Code Smell</div>
<div>Redundant pairs of parentheses should be removed</div> <div> Code Smell</div>
<div>Inheritance tree of classes should not be too deep</div> <div> Code Smell</div>
<div>Nested blocks of code should not be left empty</div> <div> Code Smell</div>