




 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... 🔍


Optional value should only be accessed after calling isPresent()
 Bug


Overrides should match their parent class methods in synchronization
 Bug


Value-based classes should not be used for locking
 Bug


Expressions used in "assert" should not produce side effects
 Bug


"volatile" variables should not be used with compound operators
 Bug


"getClass" should not be used for synchronization
 Bug

Min and max used in combination should not always return the same value
 Bug

Assignment of lazy-initialized members should be the last step with double-checked locking
 Bug


"String" calls should not go beyond their bounds
 Bug


Raw byte values should not be used in bitwise operations in combination with shifts
 Bug


Getters and setters should be synchronized in pairs
 Bug


Empty lines should not be tested with regex MULTILINE flag

Analyze your code

 Code Smell

 Critical

 ?

 regex

One way to test for empty lines is to use the regex "`^$`", which can be extremely handy when filtering out empty lines from collections of Strings, for instance. With regard to this, the Javadoc for [Pattern \(Line Terminators\)](#) states the following:

By default, the regular expressions `^` and `$` ignore line terminators and only match at the beginning and the end, respectively, of the entire input sequence. If `MULTILINE` mode is activated then `^` matches at the beginning of input and after any line terminator **except at the end of input**. When in `MULTILINE` mode `$` matches just before a line terminator or the end of the input sequence.

As emphasized, `^` is not going to match at the end of an input, and the end of the input is necessarily included in the empty string, which might lead to completely missing empty lines, while it would be the initial reason for using such regex.

Therefore, when searching for empty lines using a multi-line regular expression, you should also check whether the string is empty.

This rule is raising an issue every time a pattern that can match the empty string is used with `MULTILINE` flag and without calling `isEmpty()` on the string.

Noncompliant Code Example

```
static final Pattern p = Pattern.compile("^$", Pattern.MULTILINE);

// Alternatively
static final Pattern p = Pattern.compile("(?m)^$"); // Noncompliant

boolean containsEmptyLines(String str) {
    return p.matcher(str).find();
}

// ...
System.out.println(containsEmptyLines("a\n\nb")); // correct
System.out.println(containsEmptyLines("")); // incorrectly prints true
```

Compliant Solution

```
static final Pattern p = Pattern.compile("^$", Pattern.MULTILINE);








boolean containsEmptyLines(String str) {
    return p.matcher(str).find() || str.isEmpty();
}

// ...
System.out.println(containsEmptyLines("a\n\nb")); // correct
System.out.println(containsEmptyLines("")); // also correctly prints false
```

Available In:

https://rules.sonarsource.com/java/RSPEC-5846

1/2

<div><div>Non-thread-safe fields should not be static</div><div> Bug</div></div>	<div><div>sonarlint sonarcloud sonarqube</div><div><div>© 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</div></div></div>
<div><div>"null" should not be used with "Optional"</div><div> Bug</div></div>	
<div><div>Unary prefix operators should not be repeated</div><div> Bug</div></div>	
<div><div>"=+" should not be used instead of "+="</div><div> Bug</div></div>	