




 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

Deprecated code should be removed

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

Code Smell

Track lack of copyright and license headers

Code Smell

Octal values should not be used

Code Smell

Exit methods should not be called

Code Smell

HTTP response headers should not be vulnerable to injection attacks

Vulnerability

Members of Spring components should be injected

Vulnerability

Classes should not be loaded dynamically

Vulnerability

Equality operators should not be used in "for" loop termination conditions

Code Smell

### Vararg method arguments should not be confusing

Analyze your code

Code Smell

Major ?

Passing single null or primitive array argument to the variable arity method may not work as expected. In the case of null, it is not passed as array with single element, but the argument itself is null. In the case of a primitive array, if the formal parameter is `Object...`, it is passed as a single element array. This may not be obvious to someone not familiar with such corner cases, and it is probably better to avoid such ambiguities by explicitly casting the argument to the desired type.

#### Noncompliant Code Example

```
class A {
    public static void main(String[] args) {
        vararg(null); // Noncompliant, prints "null"
        int[] arr = {1,2,3};
        vararg(arr); // Noncompliant, prints "length: 1"
    }

    static void vararg(Object... s) {
        if (s == null) {
            System.out.println("null");
        } else {
            System.out.println("length: " + s.length);
        }
    }
}
```

#### Compliant Solution





```
class A {
    public static void main(String[] args) {
        vararg((Object) null); // prints 1
        Object[] arr = {1,2,3};
        vararg(arr); // prints 3
    }

    static void vararg(Object... s) {
        if (s == null) {
            System.out.println("null"); // not reached
        } else {
            System.out.println("length: " + s.length);
        }
    }
}
```

Available In:  
sonarlint | sonarcloud | sonarqube

https://rules.sonarsource.com/java/RSPEC-5669

1/2

<b>"Bean Validation" (JSR 380) should be properly configured</b>  Code Smell	<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. <a href="#">Privacy Policy</a></div>
<b>Spring beans should be considered by "@ComponentScan"</b>  Code Smell	
<b>Number patterns should be regular</b>  Code Smell	
<b>Lazy initialization of "static" fields should be "synchronized"</b>  Code Smell	