




 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... 🔍

The members of an interface or class declaration should appear in a pre-defined order

Code Smell

Abstract class names should comply with a naming convention

Code Smell

Strings literals should be placed on the left side when checking for equality

Code Smell

Files should contain an empty newline at the end

Code Smell

Source code should be indented consistently

Code Smell

A close curly brace should be located at the beginning of a line

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines

Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line

Code Smell

An open curly brace should be located at the beginning of a line

Code Smell

An open curly brace should be located at the end of a line

Code Smell

Tabulation characters should not be used

Code Smell

Deprecated elements should have both the annotation and the Javadoc tag

Analyze your code

Code Smell

Major ?

obsolete bad-practice

Deprecation should be marked with both the @Deprecated annotation and @deprecated Javadoc tag. The annotation enables tools such as IDEs to warn about referencing deprecated elements, and the tag can be used to explain when it was deprecated, why, and how references should be refactored.

Noncompliant Code Example

```
class MyClass {

    @Deprecated
    public void foo1() {    // Noncompliant: Add the missing @
    }

    /**
     * @deprecated
     */
    public void foo2() {    // Noncompliant: Add the missing @
    }

}
```

Compliant Solution

```
class MyClass {

    /**
     * @deprecated (when, why, refactoring advice...)
     */
    @Deprecated
    public void foo1() {
    }

}
```

Exceptions

The members and methods of a deprecated class or interface are ignored by this rule. The classes and interfaces themselves are still subject to it.





```
/**
 * @deprecated (when, why, etc...)
 */
@Deprecated
class Qix {

    public void foo() {} // Compliant; class is deprecated

}
```

https://rules.sonarsource.com/java/RSPEC-1123

1/2

Functions should not be defined with a variable number of arguments  Code Smell
Local-Variable Type Inference should be used  Code Smell
Migrate your tests from JUnit4 to the new JUnit5 annotations  Code Smell
Track uses of disallowed classes  Code Smell

```
}

/**
 * @deprecated (when, why, etc...)
 */
@Deprecated
interface Plop {

    void bar();

}
```

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](#)