




 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... 🔍

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

Functions should not be defined with a variable number of arguments

Code Smell

"@Deprecated" code should not be used

Analyze your code

Code Smell

Minor ?

cwe obsolete cert

Once deprecated, classes, and interfaces, and their members should be avoided, rather than used, inherited or extended. Deprecation is a warning that the class or interface has been superseded, and will eventually be removed. The deprecation period allows you to make a smooth transition away from the aging, soon-to-be-retired technology.

Noncompliant Code Example

```
/**
 * @deprecated As of release 1.3, replaced by {@link #Fee}
 */
@Deprecated
public class Fum { ... }

public class Foo {
    /**
     * @deprecated As of release 1.7, replaced by {@link #doT
     */
    @Deprecated
    public void doTheThing() { ... }

    public void doTheThingBetter() { ... }
}

public class Bar extends Foo {
    public void doTheThing() { ... } // Noncompliant; don't ov
}

public class Bar extends Fum { // Noncompliant; Fum is depr

    public void myMethod() {
        Foo foo = new Foo(); // okay; the class isn't deprecate
        foo.doTheThing(); // Noncompliant; doTheThing method is
    }
}
```

See

- MITRE, CWE-477 - Use of Obsolete Functions
- CERT, MET02-J. - Do not use deprecated or obsolete classes or methods

Available In:

sonarlint





sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-1874

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of

1/2

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>