




 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... 🔍

specified




Code Smell

Classes and methods that rely on the default system encoding should not be used




Code Smell

Simple class names should be used




Code Smell

Variables should not be declared before they are relevant




Code Smell

Extensions and implementations should not be redundant




Code Smell

"==" and "!=" should not be used when "equals" is overridden




Code Smell

An abstract class should have both abstract and concrete methods




Code Smell

Sets with elements that are enum values should be replaced with EnumSet




Code Smell

String operations should not rely on the default system locale




Code Smell

Comments should not be located at the end of lines of code




Code Smell

Track uses of "CHECKSTYLE:OFF" suppression comments



Code Smell


Loggers should be "private static final" and should share a naming convention





Code Smell

Only static class initializers should be used

Analyze your code

 Code Smell

 Major ?

 pitfall

Non-static initializers are rarely used, and can be confusing for most developers because they only run when new class instances are created. When possible, non-static initializers should be refactored into standard constructors or field initializers.

Noncompliant Code Example

```
class MyClass {
    private static final Map<String, String> MY_MAP = new Hash

    // Noncompliant - HashMap should be extended only to add
    {
        put("a", "b");
    }

};
}
```

Compliant Solution

```
class MyClass {
    private static final Map<String, String> MY_MAP = new Hash

    static {
        MY_MAP.put("a", "b");
    }
}
```

or using Java 9 Map.of:

```
class MyClass {
    // Compliant
    private static final Map<String, String> MY_MAP = java.util
}
```

or using Guava:

```
class MyClass {
    // Compliant
    private static final Map<String, String> MY_MAP = Immutabl
}
```

Available In:





sonarlint

sonarcloud

sonarqube

https://rules.sonarsource.com/java/RSPEC-1171

1/2

 Code Smell
Track uses of "NOPMD" suppression comments
 Code Smell
Packages should have a javadoc file 'package-info.java'
 Code Smell
The members of an interface or class declaration should appear in a pre-defined order
 Code Smell
Abstract class names should comply

© 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](#)