




 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... 🔍

The ternary operator should not be used

Code Smell

Standard functional interfaces should not be redefined

Code Smell

"NullPointerException" should not be caught

Code Smell

"NullPointerException" should not be explicitly thrown

Code Smell

Classes should not have too many methods

Code Smell

Methods should not have too many lines

Code Smell

Track uses of "NOSONAR" comments

Code Smell

Classes and enums with private members should have a constructor

Code Smell

Track comments matching a regular expression

Code Smell

Statements should be on separate lines

Code Smell

Classes should not be coupled to too many other classes (Single Responsibility Principle)

Code Smell

"java.lang.Error" should not be extended

Classes with only "static" methods should not be instantiated

Analyze your code

Code SmellMajor?clumsy

static methods can be accessed without an instance of the enclosing class, so there's no reason to instantiate a class that has only static methods.

Noncompliant Code Example

```
public class TextUtils {
    public static String stripHtml(String source) {
        return source.replaceAll("<[^>]+>", "");
    }
}

public class TextManipulator {

    // ...

    public void cleanText(String source) {
        TextUtils textUtils = new TextUtils(); // Noncompliant

        String stripped = textUtils.stripHtml(source);

        //...
    }
}
```

Compliant Solution

```
public class TextUtils {
    public static String stripHtml(String source) {
        return source.replaceAll("<[^>]+>", "");
    }
}

public class TextManipulator {

    // ...

    public void cleanText(String source) {
        String stripped = TextUtils.stripHtml(source);

        //...
    }
}
```


See Also

- {rule:java:S1118} - Utility classes should not have public constructors


Available In:

https://rules.sonarsource.com/java/RSPEC-2440


1/2

 Code Smell


Anonymous classes should not have too many lines

 Code Smell




Public types, methods and fields (API) should be documented with Javadoc

 Code Smell

Exception handlers should preserve the original exceptions

 Code Smell

Checked exceptions should not be thrown

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](#)