

 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

Abstract class names should comply with a naming convention	
Strings literals should be placed on the left side when checking for equality	
Files should contain an empty newline at the end	
Source code should be indented consistently	
A close curly brace should be located at the beginning of a line	
Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines	
Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line	
An open curly brace should be located at the beginning of a line	
An open curly brace should be located at the end of a line	
Tabulation characters should not be used	
Functions should not be defined with a variable number of arguments	

"catch" clauses should do more than rethrow

Analyze your code

 Code Smell

 Minor

 error-handling unused finding clumsy

A catch clause that only rethrows the caught exception has the same effect as omitting the catch altogether and letting it bubble up automatically, but with more code and the additional detriment of leaving maintainers scratching their heads.

Such clauses should either be eliminated or populated with the appropriate logic.

Noncompliant Code Example

```
public String readFile(File f) {
    StringBuilder sb = new StringBuilder();
    try {
        FileReader fileReader = new FileReader(fileName);
        BufferedReader bufferedReader = new BufferedReader(fileR

        while((line = bufferedReader.readLine()) != null) {
            //...
        }
        catch (IOException e) { // Noncompliant
            throw e;
        }
        return sb.toString();
    }
}
```

Compliant Solution


```
public String readFile(File f) {
    StringBuilder sb = new StringBuilder();
    try {
        FileReader fileReader = new FileReader(fileName);
        BufferedReader bufferedReader = new BufferedReader(fileR

        while((line = bufferedReader.readLine()) != null) {
            //...
        }
        catch (IOException e) {
            logger.LogError(e);
            throw e;
        }
        return sb.toString();
    }
}
```


or

```
public String readFile(File f) throws IOException {
    StringBuilder sb = new StringBuilder();
    FileReader fileReader = new FileReader(fileName);
    BufferedReader bufferedReader = new BufferedReader(fileRea
```


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

Track uses of "@SuppressWarnings" annotations

 Code Smell

```
while((line = bufferedReader.readLine()) != null) {  
    //...  
  
    return sb.toString();  
}
```

Available In:  
 |  | 

© 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](#)