




 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... 🔍

Public methods should throw at most one checked exception

Code Smell

"switch case" clauses should not have too many lines of code

Code Smell

Methods should not have too many return statements

Code Smell

Magic numbers should not be used

Code Smell

Files should not have too many lines of code

Code Smell

Lines should not be too long

Code Smell

JEE applications should not "getClassLoader"

Bug

Math should not be performed on floats

Bug

"equals" methods should be symmetric and work for subclasses

Bug

Literal suffixes should be upper case

Code Smell

Unicode-aware versions of character classes should be preferred

Code Smell

Use Java 12 "switch" expression

Code Smell

"serialVersionUID" should not be

Exceptions should be either logged or rethrown but not both

Analyze your code

Code SmellMajor?error-handling clumsy

In applications where the accepted practice is to log an `Exception` and then rethrow it, you end up with miles-long logs that contain multiple instances of the same exception. In multi-threaded applications debugging this type of log can be particularly hellish because messages from other threads will be interwoven with the repetitions of the logged-and-thrown `Exception`. Instead, exceptions should be either logged or rethrown, not both.

Noncompliant Code Example

```
catch (SQLException e) {
    ...
    LOGGER.log(Level.ERROR, contextInfo, e);
    throw new MySQLException(contextInfo, e);
}
```

Compliant Solution

```
catch (SQLException e) {
    ...
    throw new MySQLException(contextInfo, e);
}
```

or

```
catch (SQLException e) {
    ...
    LOGGER.log(Level.ERROR, contextInfo, e);
    // handle exception...
}
```





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](#)

https://rules.sonarsource.com/java/RSPEC-2139

1/2

<div>declared blindly</div> <div> Code Smell</div>
<div>"Stream.collect()" calls should not be redundant</div> <div> Code Smell</div>
<div>Local constants should follow naming conventions for constants</div> <div> Code Smell</div>
<div>Unit tests should throw exceptions</div> <div> Code Smell</div>
<div>Test methods should comply with a naming convention</div>