




 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... 🔍

Code Smell

"final" classes should not have "protected" members

Code Smell

Underscores should be used to make large numbers readable

Code Smell

"Serializable" inner classes of "Serializable" classes should be static

Code Smell

Member variable visibility should be 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

Throwable and Error should not be caught

Analyze your code

Code Smell

Major ?

cwe error-handling bad-practice cert

Throwable is the superclass of all errors and exceptions in Java. Error is the superclass of all errors, which are not meant to be caught by applications.

Catching either Throwable or Error will also catch OutOfMemoryError and InternalError, from which an application should not attempt to recover.

Noncompliant Code Example

```
try { /* ... */ } catch (Throwable t) { /* ... */ }
try { /* ... */ } catch (Error e) { /* ... */ }
```




Compliant Solution

```
try { /* ... */ } catch (RuntimeException e) { /* ... */ }
try { /* ... */ } catch (MyException e) { /* ... */ }
```

See

- [MITRE, CWE-396](#) - Declaration of Catch for Generic Exception
- [C++ Core Guidelines E.14](#) - Use purpose-designed user-defined types as exceptions (not built-in types)




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](#)

https://rules.sonarsource.com/java/RSPEC-1181

1/2

<div><div>String operations should not rely on the default system locale</div><div> Code Smell</div></div>
<div><div>Comments should not be located at the end of lines of code</div><div> Code Smell</div></div>
<div><div>Track uses of "CHECKSTYLE:OFF" suppression comments</div><div> Code Smell</div></div>
<div><div>Loggers should be "private static final" and should share a naming convention</div></div>