sonar RULES Products

- 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...

Abstract class names should comply with a naming convention
Code Smell

Strings literals should be placed on the left side when checking for equality
Code Smell

Files should contain an empty newline at the end
Code Smell

Source code should be indented consistently
Code Smell

A close curly brace should be located at the beginning of a line
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be on two different lines
Code Smell

Close curly brace and the next "else", "catch" and "finally" keywords should be located on the same line
Code Smell

An open curly brace should be located at the beginning of a line
Code Smell

An open curly brace should be located at the end of a line
Code Smell

Tabulation characters should not be used
Code Smell

Functions should not be defined with a variable number of arguments
Code Smell

"Exception" should not be caught when not required by called methods

Analyze your code

Code Smell | Minor | cwe error-handling

Catching `Exception` seems like an efficient way to handle multiple possible exceptions. Unfortunately, it traps all exception types, both checked and runtime exceptions, thereby casting too broad a net. Indeed, was it really the intention of developers to also catch runtime exceptions? To prevent any misunderstanding, if both checked and runtime exceptions are really expected to be caught, they should be explicitly listed in the `catch` clause.

This rule raises an issue if `Exception` is caught when it is not explicitly thrown by a method in the `try` block.

Noncompliant Code Example

```
try {
  // do something that might throw an UnsupportedDataTypeExc
} catch (Exception e) { // Noncompliant
  // log exception ...
}
```

Compliant Solution

```
try {
  // do something
} catch (UnsupportedEncodingException|UnsupportedDataTypeExc
  // log exception ...
}
```

or if runtime exceptions should not be caught:

```
try {
  // do something
} catch (UnsupportedEncodingException|UnsupportedDataTypeExc
  // log exception ...
}
```

See

- MITRE, CWE-396 - Declaration of Catch for Generic Exception

Available In:
sonarlint sonarcloud sonarqube

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