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

Generic exceptions should never be thrown

Analyze your code

Code Smell Major cwe error-handling cert

Using such generic exceptions as `Error`, `RuntimeException`, `Throwable`, and `Exception` prevents calling methods from handling true, system-generated exceptions differently than application-generated errors.

Noncompliant Code Example

```
public void foo(String bar) throws Throwable {  // Noncompli
  throw new RuntimeException("My Message");     // Noncompli
}
```

Compliant Solution

```
public void foo(String bar) {
  throw new MyOwnRuntimeException("My Message");
}
```

Exceptions

Generic exceptions in the signatures of overriding methods are ignored, because overriding method has to follow signature of the throw declaration in the superclass. The issue will be raised on superclass declaration of the method (or won't be raised at all if superclass is not part of the analysis).

```
@Override
public void myMethod() throws Exception {...}
```

Generic exceptions are also ignored in the signatures of methods that make calls to methods that throw generic exceptions.

```
public void myOtherMethod throws Exception {
  doTheThing();  // this method throws Exception
}
```

See

- MITRE, CWE-397 - Declaration of Throws for Generic Exception
- CERT, ERR07-J. - Do not throw RuntimeException, Exception, or Throwable

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