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...

Exception testing via JUnit @Test annotation should be avoided
Code Smell

Escape sequences should not be used in text blocks
Code Smell

Simple string literal should be used for single line strings
Code Smell

Boxed "Boolean" should be avoided in boolean expressions
Code Smell

Type parameters should not shadow other type parameters
Code Smell

"read(byte[],int,int)" should be overridden
Code Smell

An iteration on a Collection should be performed on the type handled by the Collection
Code Smell

"StandardCharsets" constants should be preferred
Code Smell

"@CheckForNull" or "@Nullable" should not be used on primitive types
Code Smell

Composed "@RequestMapping" variants should be preferred
Code Smell

"write(byte[],int,int)" should be overridden
Code Smell

Reflection should not be used to check non-runtime annotations

Analyze your code

Bug Major

The writer of an annotation can set one of three retention policies for it:

- `RetentionPolicy.SOURCE` - these annotations are dropped during compilation, E.G. `@Override`, `@SuppressWarnings`.
- `RetentionPolicy.CLASS` - these annotations are present in a compiled class but not loaded into the JVM at runtime. This is the default.
- `RetentionPolicy.RUNTIME` - these annotations are present in the class file and loaded into the JVM.

Only annotations that have been given a `RUNTIME` retention policy will be available to reflection. Testing for annotations with any other retention policy is simply an error, since the test will always return false.

This rule checks that reflection is not used to detect annotations that do not have `RUNTIME` retention.

Noncompliant Code Example

```
Method m = String.class.getMethod("getBytes", new Class[] {i
int.class, byte[].class, int.class});
if (m.isAnnotationPresent(Override.class)) {  // Noncomplian
```

Available In:

sonarlint | sonarcloud | sonarqube

Functional Interfaces should be as specialised as possible

Code Smell

Null checks should not be used with "instanceof"

Code Smell

"close()" calls should not be redundant

Code Smell

"ThreadLocal.withInitial" should be preferred

Code Smell