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...

Multiple loops over the same set should be combined
Code Smell

Classes without "public" constructors should be "final"
Code Smell

Unnecessary semicolons should be omitted
Code Smell

Literal boolean values and nulls should not be used in assertions
Code Smell

Test assertions should include messages
Code Smell

Mutable members should not be stored or returned directly
Code Smell

Redundant modifiers should not be used
Code Smell

"private" and "final" methods that don't access instance data should be "static"
Code Smell

Files should not be empty
Code Smell

Collection methods with O(n) performance should be used carefully
Code Smell

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

"collect" should be used with

Anonymous inner classes containing only one method should become lambdas

Analyze your code

Code Smell Major java8

Before Java 8, the only way to partially support closures in Java was by using anonymous inner classes. But the syntax of anonymous classes may seem unwieldy and unclear.

With Java 8, most uses of anonymous inner classes should be replaced by lambdas to highly increase the readability of the source code.

Note that this rule is automatically disabled when the project's `sonar.java.source` is lower than `8`.

Noncompliant Code Example

```
myCollection.stream().map(new Mapper<String,String>() {
  public String map(String input) {
    return new StringBuilder(input).reverse().toString();
  }
});

Predicate<String> isEmpty = new Predicate<String> {
    boolean test(String myString) {
        return myString.isEmpty();
    }
}
```

Compliant Solution

```
myCollection.stream().map(input -> new StringBuilder(input).

Predicate<String> isEmpty = myString -> myString.isEmpty();
```

Available In:
sonarlint sonarcloud sonarqube

"Streams" instead of "list::add"

Code Smell

Switches should be used for sequences of simple "String" tests

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