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

Mutable fields should not be "public static"

Analyze your code

Code Smell | Minor | cwe unpredictable cert

There is no good reason to have a mutable object as the `public` (by default), `static` member of an `interface`. Such variables should be moved into classes and their visibility lowered.

Similarly, mutable `static` members of classes and enumerations which are accessed directly, rather than through getters and setters, should be protected to the degree possible. That can be done by reducing visibility or making the field `final` if appropriate.

Note that making a mutable field, such as an array, `final` will keep the variable from being reassigned, but doing so has no effect on the mutability of the internal state of the array (i.e. it doesn't accomplish the goal).

This rule raises issues for `public static` array, `Collection`, `Date`, and `awt.Point` members.

Noncompliant Code Example

```
public interface MyInterface {
  public static String [] strings; // Noncompliant
}

public class A {
  public static String [] strings1 = {"first","second"};  //
  public static String [] strings2 = {"first","second"};  //
  public static List<String> strings3 = new ArrayList<>();
  // ...
}
```

See

- MITRE, CWE-582 - Array Declared Public, Final, and Static
- MITRE, CWE-607 - Public Static Final Field References Mutable Object
- CERT, OBJ01-J. - Limit accessibility of fields
- CERT, OBJ13-J. - Ensure that references to mutable objects are not exposed

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