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

Declarations should use Java collection interfaces such as "List" rather than specific implementation classes such as "LinkedList"

Analyze your code

Code Smell | Minor | Quick Fix | bad-practice

The purpose of the Java Collections API is to provide a well defined hierarchy of interfaces in order to hide implementation details.

Implementing classes must be used to instantiate new collections, but the result of an instantiation should ideally be stored in a variable whose type is a Java Collection interface.

This rule raises an issue when an implementation class:

- is returned from a `public` method.
- is accepted as an argument to a `public` method.
- is exposed as a `public` member.

Noncompliant Code Example

```
public class Employees {
  private HashSet<Employee> employees = new HashSet<Employee

  public HashSet<Employee> getEmployees() {
    return employees;
  }
}
```

Compliant Solution

```
public class Employees {
  private Set<Employee> employees = new HashSet<Employee>();

  public Set<Employee> getEmployees() {
    return employees;
  }
}
```

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