sonar

RULES

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 rules632

Vulnerability53

Bug154

Security Hotspot36

Code Smell389

Quick Fix42

Tags

Search by name...

Underscores should be used to make large numbers readable

Code Smell

"Serializable" inner classes of "Serializable" classes should be static

Code Smell

Member variable visibility should be specified

Code Smell

Classes and methods that rely on the default system encoding should not be used

Code Smell

Simple class names should be used

Code Smell

Variables should not be declared before they are relevant

Code Smell

Extensions and implementations should not be redundant

Code Smell

"==" and "!=" should not be used when "equals" is overridden

Code Smell

An abstract class should have both abstract and concrete methods

Code Smell

Sets with elements that are enum values should be replaced with EnumSet

Code Smell

String operations should not rely on the default system locale

Code Smell

Comments should not be located at the end of lines of code

Unused method parameters should be removed

Analyze your code

Code Smell

Major

Quick Fix

cert unused

Unused parameters are misleading. Whatever the values passed to such parameters, the behavior will be the same.

Noncompliant Code Example

```
void doSomething(int a, int b) {    // "b" is unused
    compute(a);
}
```

Compliant Solution

```
void doSomething(int a) {
    compute(a);
}
```

Exceptions

The rule will not raise issues for unused parameters:

- that are annotated with `@javax.enterprise.event.Observes`
- in overrides and implementation methods
- in interface default methods
- in non-private methods that only throw or that have empty bodies
- in annotated methods, unless the annotation is `@SuppressWarnings("unchecked")` or `@SuppressWarnings("rawtypes")`, in which case the annotation will be ignored
- in overridable methods (non-final, or not member of a final class, non-static, non-private), if the parameter is documented with a proper javadoc.

```
@Override
void doSomething(int a, int b) {    // no issue reported on
    compute(a);
}

public void foo(String s) {
    // designed to be extended but noop in standard case
}





protected void bar(String s) {
    //open-closed principle
}

public void qix(String s) {
    throw new UnsupportedOperationException("This method shoul
}

/**
```

https://rules.sonarsource.com/java/RSPEC-1172

1/2

 Code Smell
Track uses of "CHECKSTYLE:OFF" suppression comments  Code Smell
Loggers should be "private static final" and should share a naming convention  Code Smell
Track uses of "NOPMD" suppression comments  Code Smell
Packages should have a javadoc file

```
* @param s This string may be use for further computation i
*/
protected void foobar(int a, String s) { // no issue, method
    compute(a);
}
```

See

- [CERT, MSC12-C](#) - Detect and remove code that has no effect or is never executed

Available In:

