

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...

Code Smell

Reflection should not be used to increase accessibility of classes, methods, or fields

Static fields should not be updated in constructors

"Thread.sleep" should not be used in tests

"entrySet()" should be iterated when both the key and value are needed

"DateUtils.truncate" from Apache Commons Lang library should not be used

Multiline blocks should be enclosed in curly braces

"readObject" should not be "synchronized"

"Preconditions" and logging arguments should not require evaluation

Boolean expressions should not be gratuitous

"Lock" objects should not be "synchronized"

Classes with only "static" methods should not be instantiated

JUnit5 test classes and methods should not be silently ignored

Analyze your code

BugMajorjunit tests

JUnit5 is more tolerant regarding the visibilities of Test classes and methods than JUnit4, which required everything to be public. JUnit5 supports default package, public and protected visibility, even if it is recommended to use the default package visibility, which improves the readability of code.

But JUnit5 ignores without any warning:

- private classes and private methods
- static methods
- methods returning a value without being a TestFactory

Noncompliant Code Example

```
import org.junit.jupiter.api.Test;

class MyClassTest {
    @Test
    private void test1() { // Noncompliant - ignored by JUnit5
        // ...
    }
    @Test
    static void test2() { // Noncompliant - ignored by JUnit5
        // ...
    }
    @Test
    boolean test3() { // Noncompliant - ignored by JUnit5
        // ...
    }
    @Nested
    private class MyNestedClass { // Noncompliant - ignored by
        @Test
        void test() {
            // ...
        }
    }
}
```





Compliant Solution

```
import org.junit.jupiter.api.Test;

class MyClassTest {
    @Test
    void test1() {
        // ...
    }
    @Test
    void test2() {
        // ...
    }
}
```

https://rules.sonarsource.com/java/RSPEC-5810

1/2

 Code Smell
"Threads" should not be used where "Runnables" are expected
 Code Smell
Inner class calls to super class methods should be unambiguous
 Code Smell
Unused type parameters should be removed
 Code Smell
Parameters should be passed in the correct order

```
@Test
void test3() {
    // ...
}
@Nested
class MyNestedClass {
    @Test
    void test() {
        // ...
    }
}
```

Available In:
sonarlint  | sonarcloud  | sonarqube 