


-  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... 🔍

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

Constructors should only call non-overridable methods

Analyze your code

Code SmellCritical🔍cert pitfall

Calling an overridable method from a constructor could result in failures or strange behaviors when instantiating a subclass which overrides the method.

For example:

- The subclass class constructor starts by contract by calling the parent class constructor.
- The parent class constructor calls the method, which has been overridden in the child class.
- If the behavior of the child class method depends on fields that are initialized in the child class constructor, unexpected behavior (like a `NullPointerException`) can result, because the fields aren't initialized yet.

Noncompliant Code Example

```
public class Parent {

    public Parent () {
        doSomething(); // Noncompliant
    }

    public void doSomething () { // not final; can be overridden
        ...
    }
}

public class Child extends Parent {

    private String foo;

    public Child(String foo) {
        super(); // leads to call doSomething() in Parent constructor
        this.foo = foo;
    }





    public void doSomething () {
        System.out.println(this.foo.length());
    }
}
```

See

- [CERT, MET05-J](#) - Ensure that constructors do not call overridable methods
- [CERT, OOP50-CPP](#) - Do not invoke virtual functions from constructors or destructors

Available In:

sonarlint🔴 | sonarcloud🟡 | sonarqube🟢

<div>Local-Variable Type Inference should be used</div> <div> Code Smell</div>
<div>Migrate your tests from JUnit4 to the new JUnit5 annotations</div> <div> Code Smell</div>
<div>Track uses of disallowed classes</div> <div> Code Smell</div>
<div>Track uses of "@SuppressWarnings" annotations</div> <div> Code Smell</div>

© 2008-2022 SonarSource S.A., Switzerland. All content is copyright protected. SONAR, SONARSOURCE, SONARLINT, SONARQUBE and SONARCLOUD are trademarks of SonarSource S.A. All other trademarks and copyrights are the property of their respective owners. All rights are expressly reserved.  
[Privacy Policy](#)