

 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

conventions for constants
Code Smell
Unit tests should throw exceptions
Code Smell
Test methods should comply with a naming convention
Code Smell
Value-based objects should not be serialized
Code Smell
Default annotation parameter values should not be passed as arguments
Code Smell
Method parameters should be declared with base types
Code Smell
Fields should not be initialized to default values
Code Smell
Multiple loops over the same set should be combined
Code Smell
Classes without "public" constructors should be "final"
Code Smell
Unnecessary semicolons should be omitted
Code Smell
Literal boolean values and nulls should not be used in assertions
Code Smell
Test assertions should include messages
Code Smell

Tags ▾

Search by name... 🔍

A field should not duplicate the name of its containing class

Analyze your code

Code Smell Major ? brain-overload

It's confusing to have a class member with the same name (case differences aside) as its enclosing class. This is particularly so when you consider the common practice of naming a class instance for the class itself.

Best practice dictates that any field or member with the same name as the enclosing class be renamed to be more descriptive of the particular aspect of the class it represents or holds.

Noncompliant Code Example

```
public class Foo {
    private String foo;

    public String getFoo() { }
}

Foo foo = new Foo();
foo.getFoo() // what does this return?
```

Compliant Solution

```
public class Foo {
    private String name;

    public String getName() { }
}





//...

Foo foo = new Foo();
foo.getName()
```

Exceptions

When the type of the field is the containing class and that field is static, no issue is raised to allow singletons named like the type.

```
public class Foo {
    ...
    private static Foo foo;
    public Foo getInstance() {
        if(foo==null) {
            foo = new Foo();
        }
        return foo;
    }
    ...
}
```

<div>Mutable members should not be stored or returned directly</div> <div> Code Smell</div>
<div>Redundant modifiers should not be used</div> <div> Code Smell</div>
<div>"private" and "final" methods that don't access instance data should be "static"</div> <div> Code Smell</div>
<div>Files should not be empty</div> <div> Code Smell</div>

Available In:

sonarlint  | **sonarcloud**  | **sonarqube** 

© 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](#)