

- 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



C# static code analysis

Unique rules to find Bugs, Vulnerabilities, Security Hotspots, and Code Smells in your C# code

All rules 409

Vulnerability 34

Bug 76

Security Hotspot 28

Code Smell 271

Quick Fix 52

Tags ▾

Search by name...



"protected" members

Code Smell

Underscores should be used to make large numbers readable

Code Smell

"ToString()" calls should not be redundant

Code Smell

"==" should not be used when "Equals" is overridden

Code Smell

An abstract class should have both abstract and concrete methods

Code Smell

Multiple variables should not be declared on the same line

Code Smell

Culture should be specified for "string" operations

Code Smell

"switch" statements should have at least 3 "case" clauses

Code Smell

break statements should not be used except for switch cases

Code Smell

String literals should not be duplicated

Code Smell

Files should contain an empty newline at the end

Code Smell

Unused "using" should be removed

Code Smell

Method overloads should be grouped together

Analyze your code

Code Smell Minor ? convention

For clarity, all overloads of the same method should be grouped together. That lets both users and maintainers quickly understand all the current available options.

Noncompliant Code Example

```
interface IMyInterface
{
    int DoTheThing(); // Noncompliant - overloaded method decl
    string DoTheOtherThing();
    int DoTheThing(string s);
}
```

Compliant Solution

```
interface IMyInterface
{
    int DoTheThing();
    int DoTheThing(string s);
    string DoTheOtherThing();
}
```

Exceptions

As it is common practice to group method declarations by implemented interface, no issue will be raised for implicit and explicit interface implementations if grouped together with other members of that interface.





As it is also a common practice to group method declarations by accessibility level, no issue will be raised for method overloads having different access modifiers.

Example:

```
class MyClass
{
    private void DoTheThing(string s) // Ok - this method is d
    {
        // ...
    }




    private string DoTheOtherThing(string s)
    {
        // ...
    }

    public void DoTheThing()
    {
        // ...
    }
}
```

A close curly brace should be located at the beginning of a line
 Code Smell
Tabulation characters should not be used
 Code Smell
Methods and properties should be named in PascalCase
 Code Smell
Track uses of in-source issue suppressions
 Code Smell

```
}  
}
```

Available In:

© 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](#)