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

Operators should be overloaded consistently

Analyze your code

Code Smell | Major | pitfall

When implementing operator overloads, it is very important to make sure that all related operators and methods are consistent in their implementation.

The following guidelines should be followed:

- When providing `operator ==` you should also provide `operator !=` and vice-versa.
- When providing `operator ==` you should also provide `Equals(Object)` and `GetHashCode()`.
- When providing `operator +, -, *, / or %` you should also provide `operator ==`, respecting previous guidelines.

This rule raises an issue when any of these guidelines are not followed on publicly-visible type (public, protected or protected internal).

Noncompliant Code Example

```
using System;

namespace MyLibrary
{
  public class Foo // Noncompliant
  {
    private int left;
    private int right;

    public Foo(int l, int r)
    {
      this.left = l;
      this.right = r;
    }

    public static Foo operator +(Foo a, Foo b)
    {
      return new Foo(a.left + b.left, a.right + b.right);
    }

    public static Foo operator -(Foo a, Foo b)
    {
      return new Foo(a.left - b.left, a.right - b.right);
    }
  }
}
```

Compliant Solution

```
using System;

namespace MyLibrary
{
```

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

```
  public class Foo
  {
    private int left;
    private int right;

    public Foo(int l, int r)
    {
      this.left = l;
      this.right = r;
    }

    public static Foo operator +(Foo a, Foo b)
    {
      return new Foo(a.left + b.left, a.right + b.right);
    }

    public static Foo operator -(Foo a, Foo b)
    {
      return new Foo(a.left - b.left, a.right - b.right);
    }

    public static bool operator ==(Foo a, Foo b)
    {
      return (a.left == b.left && a.right == b.right);
    }

    public static bool operator !=(Foo a, Foo b)
    {
      return !(a == b);
    }

    public override bool Equals(Object obj)
    {
      Foo a = obj as Foo;
      if (a == null)
        return false;
      return this == a;
    }

    public override int GetHashCode()
    {
       return (this.left * 10) + this.right;
    }
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube