C# static code analysis

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

"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

"GC.SuppressFinalize" should not be invoked for types without destructors

Analyze your code

Code Smell   Minor   Quick Fix   unused confusing

`GC.SuppressFinalize` asks the Common Language Runtime not to call the finalizer of an object. This is useful when implementing the dispose pattern where object finalization is already handled in `IDisposable.Dispose`. However, it has no effect if there is no finalizer defined in the object's type, so using it in such cases is just confusing.

This rule raises an issue when `GC.SuppressFinalize` is called for objects of `sealed` types without a finalizer.

Note: {rule:csharpsquid:S3971} is a stricter version of this rule. Typically it makes sense to activate only one of these 2 rules.

Noncompliant Code Example

```
sealed class MyClass
{
  public void Method()
  {
    ...
    GC.SuppressFinalize(this); //Noncompliant
  }
}
```

Compliant Solution

```
sealed class MyClass
{
  public void Method()
  {
    ...
  }
}
```

Available In:

sonarlint | sonarcloud | sonarqube

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