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

Classes should not be coupled to too many other classes (Single Responsibility Principle)

Analyze your code

Code Smell | Major | brain-overload

According to the Single Responsibility Principle, introduced by Robert C. Martin in his book "Principles of Object Oriented Design", a class should have only one responsibility:

> If a class has more than one responsibility, then the responsibilities become coupled.
> Changes to one responsibility may impair or inhibit the class' ability to meet the others.
> This kind of coupling leads to fragile designs that break in unexpected ways when changed.

Classes which rely on many other classes tend to aggregate too many responsibilities and should be split into several smaller ones.

Nested classes dependencies are not counted as dependencies of the outer class.

Noncompliant Code Example

With a threshold of 5:

```
public class Foo    // Noncompliant - Foo depends on too man
{
  private T1 a1;    // Foo is coupled to T1
  private T2 a2;    // Foo is coupled to T2
  private T3 a3;    // Foo is coupled to T3

  public T4 Compute(T5 a, T6 b)    // Foo is coupled to T4,
  {
    T7 result = a.Process(b);    // Foo is coupled to T7
    return result;
  }

  public static class Bar    // Compliant - Bar depends on 2
  {
    public T8 a8;
    public T9 a9;
  }
}
```

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