

Clean Code Analyzer

Mohammad Babaei 96521074

Mobin Dariush Hamedani 96521191

Dr Saeed Parsa

Overview

Today there are many programmers in the world. But what makes a good programmer?

As Linus Torvalds once said, “Talk is cheap. Show me the code”. But how can we say if a piece of code is of high quality and a programmer is professional or not?

Based on the methods of static analysis, checking the code and analyzing it based on the Clean Code principles presented by the famous author Robert C. Martin would be one way to go about doing so.

Goals

1. User should enter the code in an editor
2. The project should be implemented as a website so anyone can use it from any platform.
3. The project should analyze the code and check if it adheres to 20 different clean code principles.
4. Should show the clean code principle quote by Robert C Martin or other famous authors for each issue.
5. For each issue found in the code, it must show the exact line of the issue in the code to the user.


Specifications

All goals described above are implemented in this project.

We used Coco and Roslyn parser and lexical analyzer APIs to help us analyze the code.

Principles

1. A function should do one thing, and only one thing.
2. Blocks within if statements, `_else_` statements, `_while_` statements, and so on should be one line long.
3. The ideal number of arguments for a function is zero (nomadic), Next comes one (monadic), followed closely by two (dyadic). Three arguments (triadic) should be avoided where possible. More than



three (polyadic) requires very special justification — and then shouldn't be used anyway.

4. Nested loops are frequently (but not always) bad practice.
5. It is well known that I prefer code that has few comments. I code by the principle that good code does not require many comments.

6. The use of static methods, in any context, is a perfect indicator of a bad programmer who has no idea what OOP is.

7. We should write our tests, trying to assert one concept per unit test.

8. Replace Magic Numbers with Named Constants

9. As with functions, smaller is the primary rule when it comes to designing Classes.

10. it's usually a very bad idea to have an empty catch block

11. Nested if statements are Bad ideas.

12. The code should be read vertically not horizontally. Therefore Long horizontal lines of code should be avoided.(detect long code lines)

13. Methods should be small, smaller than small.

14. Try to create functions with less than 4 arguments.

15. Static fields introduce global state and so should be avoided

16. Use meaningful names for functions and classes

17. Static fields introduce global state and so should be avoided

18. The name of a variable, function, or class, should answer all the big questions. It should tell you why it exists, what it does, and how it is used.

19. Functions with many parameters are hard to read and hard to use.

20. Method parameters must not use REF or OUT parameters; all results should be via a return

