Clean Code and Code Refactoring

Description

There are two golden rules for being a good programmer.

1- Writing clean code
2- Changing the structure of your code without changing its external behavior

Clean Code; is a set of principles that refers to writing code that is easy to understand and change by humans.
Writing comprehensible code means that the code can be easily understood, either by its developer or by another developer. All code is written logically and clearly. The connections between the different parts of the code are clear and unambiguous. The task of each function, method, and variable is easily understood by any developer.
Refactoring; are changes made to the internal structure of the software in order to make it simpler, more understandable, easier to change and do not affect the external behavior of the software

This course will teach you all the tips and tricks you need to do both, allowing you to code like a pro in no time.

Outline

Introduction to Clean Code

- What is Clean Code?
- Why is Clean Code important?
- What are Clean Code Principles?

Code Formatting and Naming

- Code formatting and indenting
- Naming conventions for variables, methods, and classes
- Use meaningful names to improve the readability of code

Functions/Methods

- Function arguments and return values
- Naming and editing functions

Reviews and Documentation

- When to use reviews
- Write clear and unambiguous comments
- Certifying codes for sustainability

Introduction to Refactoring

- What is refactoring?
- Why is refactoring important?
- What are the benefits of refactoring?

Refactoring

- Detection and elimination of code smells
- Techniques for improving code quality
- Applied refactoring
- Improving Performance
- Performance Benchmarking
- Method Revisions
- Class and Object Refactoring
- Class Hierarchy Reorganizations
- Pattern-Based Refactorings

Code Smell

- Identify Code Smell
- Common code smells in C# coding
- Refactoring to improve code quality