Clean Code .NET

Introduction

“Honesty in small things is not a small thing.” It was a good omen consistent with what I already wanted to say here. Small things matter. This is a book about humble concerns whose value is nonetheless far from small.

> God is in the details, said the architect Ludwig mies van der Rohe.

As Uncle Bob relates in his front matter, good software practice requires such discipline: focus, presence of mind, and thinking.

The 5S philosophy comprises these concepts:

”***Seiri***”, or organization (think “**sort**” in English). Knowing where things are—using approaches such as suitable naming—is crucial.

“***Seiton***”, or tidiness (think “**systematize**” in English). There is an old American saying: A place for everything, and everything in its place. A piece of code should be where you expect to find it—and, if not, you should re-factor to get it there.

“***Seiso***”, or cleaning (think “**shine**” in English): Keep the workplace free of hanging wires, grease, scraps, and waste. What do the authors here say about littering your code with comments and commented-out code lines that capture history or wishes for the future? Get rid of them.

“***Seiketsu***”, or **standardization**: The group agrees about how to keep the workplace clean. Do you think this book says anything about having a consistent coding style and set of practices within the group? Where do those standards come from?

“***Shutsuke***”, or discipline (**self-discipline**). This means having the discipline to follow the practices and to frequently reflect on one’s work and be willing to change.

What it takes to be true professional

There are two parts to learning craftsmanship: **knowledge** and **work**.

You must gain the knowledge of principles, patterns, practices, and heuristics that a craftsman knows, and you must also grind that knowledge into your fingers, eyes, and gut by working hard and practicing.

What is Clean Code

> Pragmatic Dave Thomas and Andy Hunt said this a different way. They used the metaphor of broken windows. A building with broken windows looks like nobody cares about it. So other people stop caring. They allow more windows to become broken. Eventually they actively break them. They despoil the facade with graffiti and allow garbage to collect. One broken window starts the process toward decay.

The Boy Scout Rule

It's not enough to write the code well. The code has to be kept clean over time. We've all seen code rot and degrade as time passes. So we must take an active role in preventing this degradation.

The Boy Scouts of America have a simple rule that we can apply to our profession.

> Leave the campground cleaner than you found.

Meaningful Names

Names are everywhere in software. We name our variables, our functions, our arguments, classes, and packages.
We name our source files and the directories that contain them. We name our jar files and war files and ear files. We name and name and name.

Because we do so much of it, we'd better do it well. What follows are some simple rules for creating good names.

There are some rules that we should apply to our everyday code:

-Use Intention-Revealing Names

-Make Meaningful Distinctions

-Use Pronouncable Names

-Use Searchable Names

-Avoid Mental Mapping

Readers shouldn't have to mentally translate your names into other names they already know. This problem generally arises from a choice to use neither problem domain terms nor solution domain terms.

-Class Names

Classes and objects should have noun or noun phrase names like Customer,
WikiPage,
Account, and AddressParser.
Avoid words like Manager, Processor, Data, or Info in the name of a class.
A class name should not be a verb.

-Method Names

Methods should have verb or verb phrase names like postPayment, deletePage, or save. Accessors, mutators, and predicates should be named for their value and prefixed with get, set, and is according to the javabean standard.

When constructors are overloaded, use static factory methods with names that describe the arguments.

-Pick One Word per Concept

-Don't Pun

-Use Problem Domain Names

Final Words

People are also afraid of renaming things for fear that some other developers will object. We do not share that fear and find that we are actually grateful when names change (for the better). Most of the time we don't really memorize the names of classes and methods.

You will probably end up surprising someone when you rename, just like you might with any other code improvement. Don't let it stop you in your tracks

Follow some of these rules and see whether you don't improve the readability of your code. If you are maintaining someone else's code, use refactoring tools to help resolve these problems. It will pay off in the short term and continue to pay in the long run.
