The Importance of Writing Clean Code

What do we mean by Clean Code?

How many of us have been assigned to an application that we haven't written and wound up scratching our heads wondering who wrote this? How much time did it take for you to figure out what that code was trying to do? Well maybe what you were dealing with is commonly described as spaghetti code! So how do we avoid this? By writing clean code of course! Clean code is simply code that is readable and can easily be followed by a human. Clean code should be simple, readable, understandable and easily maintainable. Clean code should flow and almost read like a story unto itself. It should not require a lot of comments either as each function, method and variable should be self-explanatory.

Why is writing Clean Code Important?

Have you ever had to add a feature to existing code that you've found to be hard to follow? Maybe the code was written as one huge minified blob with no spacing, no commenting, obscure variable and method naming? You struggle to understand, to follow, what the previous developer was thinking as you try and add the new feature only to realize that what you've added inadvertently broke something else and you're not sure where or what broke. That happens quite often and always results in a bad release or extra time and money spent trying to fix the new issue. That's why writing clean code is so important! It's to simply write code in a way so that the next developer, or even yourself, can easily follow and understand so that they can safely add to it or change it knowing that it won't break or negatively affect the existing code. Clean code therefore can be viewed as basically a time and cost saver and that's what really makes it important.

So How do we go about writing Clean Code?

Writing clean code is about taking certain guiding principles and some common sense into account when coding. It's about code craftmanship, telling a story – excelling at our craft. So how do we get there? The following outlines some guiding principles that will help you in your journey towards writing clean code.

Readability over conciseness

Obviously code needs to work and be understood by the machine executing it. Developers though also need to be able to understand the code, especially if you're working on a project with multiple people. That's why the readability of code is always more important than its' conciseness. Simple, common sense things like judicious spacing within a function or method, variable names that somewhat describe what they are being used for and adding comments where maybe a certain section of code is doing something that might not be obvious. There's no point in writing concise code if other developers cannot understand it.

KISS

Keep it simple, stupid is one of the oldest principles of clean code. KISS is all about writing code as simply as possible. You should avoid making your code unnecessarily complex. When it comes to coding, there is almost never a single way to solve a problem. What you should really be asking yourself is: "What is the easiest, most straightforward, way to solve this problem?".

DRY

Don't repeat yourself is a more specific version of KISS. According to the DRY principle, functions or methods in clean code should, wherever possible, only do one thing within the overall application.

YAGNI

You aren't gonna need it is based on the principle that a developer should only introduce additional functionality to code when it is necessary. YAGNI is closely tied to agile software development methods. According to the YAGNI principle, instead of starting from an overarching concept during development, you should code the software architecture in small steps to be able to react to problems dynamically and individually. Clean code is always produced when the underlying problem has been solved in the most efficient way possible.

Lasagna Code

Over abstraction happens because the developer is too eager to capture every possible future development that may occur. This can lead to code that is so vague, or so layered, that it can send future developers down a rabbit hole trying to figure it out. Don't code for two years into the future, code for the now. This will allow your code to organically grow in a structurally sound manner. It also keeps you from being tied into implemented ideas that may never materialize.

Clean Code when it comes to Security

If your code lacks consistency, is poorly laid out and undocumented, you're adding to the overall complexity of your application. This can lead to security issues because complexity hides bugs, some of which may result in security vulnerabilities. An attacker only needs to find one way into your application. You are playing the role of defender and as the defender, you must try to find and mitigate them all. This task gets harder as your codebase grows in size and complexity, but you can minimize this effect by following the principles as outlined above and writing clean code.

To Summarize

Code should be written in such a way that it is well-structured, self-documenting, non-ambiguous and easy to follow. No one likes inheriting and maintaining spaghetti code - the opposite of clean code. Clean code will minimize the introduction of bugs simply because the code is understandable and well-structured. Code that is not clean can lead to unnecessary bugs and more importantly, vulnerabilities, which can threaten the

integrity and security of your data. We should all strive to write clean code, to be code craftsmen as it will ultimately reduce bugs, vulnerabilities and in the long run, cost.