What’s clean code?

If you, just as many developers, have the book called *Clean Code*, by Robert Cecil Martin (Uncle Bob) in your desk, you know the meaning of clean code is very wide and specialists have their own definitions and even disagree about them.

Grady Booch, Software Engineering chief-scientist at Software da IBM Research, for example, is cited by the author because he considers that: “Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer's intent but rather is full of crisp abstractions and straightforward lines of control.”

However, Michael Feathers, author of *Working Effectively with Legacy Code*, highlights the word “care” in a clean code’s definition.

Technique to make it easier to write and read code

Even though it has different definitions, you can merge different meanings and say that clean code is applying certain techniques to **make the code’s reading and writing easier**.

This happens because a program or system is never totally complete, it will always have updates and new features will always be added, so it must be as clean as possible.

Why is clean code important?

When you produce a program for a business, the business itself undergoes changes. The business requirements change, the user needs changes, and even the code you wrote might need to change.

And another person will need to read your code, understand it, change it, and even correct it. Having a clean code implies high readability, which helps with maintenance. According to Uncle Bob’s book:

“If you’ve been a programmer for more than two or three years, you probably slowed down while reading confusing code. The slowdown might be significant. [...] The mess can get so big, deep, and high they can’t clean it.”

How should a clean code be?

To avoid it, when you develop systems, you should follow the principles below: The clean code should be:

- **Simple and direct**: following the KISS (Keep It Simple Stupid) principle, a well-written code should have the minimum of complexity possible, so that it’s easily understood and debugged.
- **Dry**: the DRY code stands for Don’t Repeat It – which was a concept created by Andy Hunt, one of the authors of Agile Manifesto -, it’s the

one with no ambiguity, in other words, if you have already added it to the source code, it shouldn't be implemented one more time.

- **Efficient**: The code is programmed to achieve a specific objective, so make sure it will work the way it was intended to.
- **Elegant**: Bjarne Stroustrup, C++ inventor, says he likes his code elegant and that when you read it, you should feel happy. The idea of elegance is making one code different from the others.
- **Attentive to details**: the programmer should always create the code carefully, since having it poorly written from the beginning will have an impact on its maintenance, which causes losses and slowness to understand it.
- **Attentive to comments**: make as few comments as possible in your code. The code should be clear enough so that it doesn't need any comments. If they're really necessary, try to avoid comments as much as possible.

Clean code: programming tests

Considering these rules, it's always important to remember the code you wrote will probably undergo maintenance and refactoring. The code can only be considered clean if it undergoes tests.

When you create a program, hence, you should ensure each code line is validated, so that it keeps working.

You should also adopt TDD (Test-Driven Development) methodology, and perform integrity, installing, settings, usability, security, and integration tests, besides others.

How can you write good code?

So what are the best practices to have a clean code?

- **Define good names**: having a name is essential for a code. It should be direct and represent well what it means, even if it implies having a large name.
- **Class names**: avoid words that are conflicting with language components and use nouns instead of verbs.
- **Method names**: here you should use verbs and express the developer's intention, such as "DeletePage".
- **TDD**: as mentioned above, you should test everything all the time.
- **Hungarian notation isn't effective**: we know today that Hungarian notation (using the variable type right after your name) harms reading and nowadays there are more appropriate programming languages.

- **Good names need no comments**: replace them with clear names so that there’s no more confusion.
- **Develop an error treatment**: if there are mistakes, you should find ways to treat them.
- **Formatting**: pay attention to code formatting and indentation to make it more visible.
- **Package**: categorize coding into packages so that they’re more organized and elegant.

Learn more about SaaS - Software as a Service
Development with a focus on clean code makes programming faster and generates less waste and delay for the business.
When you code like this, you make **process automation** easier in companies, making it more simple for customers to read and use it.

This concept is essential for IT businesses based in Software as a Service (SaaS), which have a recurring profit model, by monthly or yearly subscription, for example.
This is because the system will be on a **cloud** and will give users more flexibility to decide which features the system needs. The IT company gives the complete support the business needs, so that it can modify the system easier.