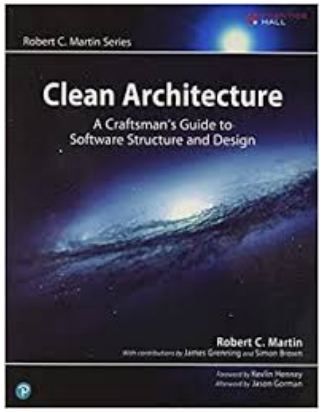
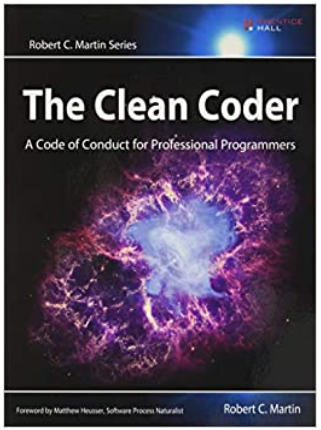
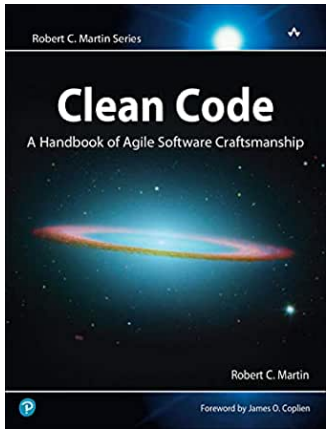






Robert Cecil Martin

American software engineer

cleancoder.com

Robert Cecil Martin, colloquially called "Uncle Bob", is an American software engineer, instructor, and best-selling author. He is most recognized for developing many software design principles and for being a founder of the influential Agile Manifesto. Martin has authored many books and magazine articles. Wikipedia

Born: 1952 (age 68 years), Palo Alto, CA

Other name: "Uncle Bob" Martin

Known for: Agile Manifesto, SOLID principles

Children: Micah Martin

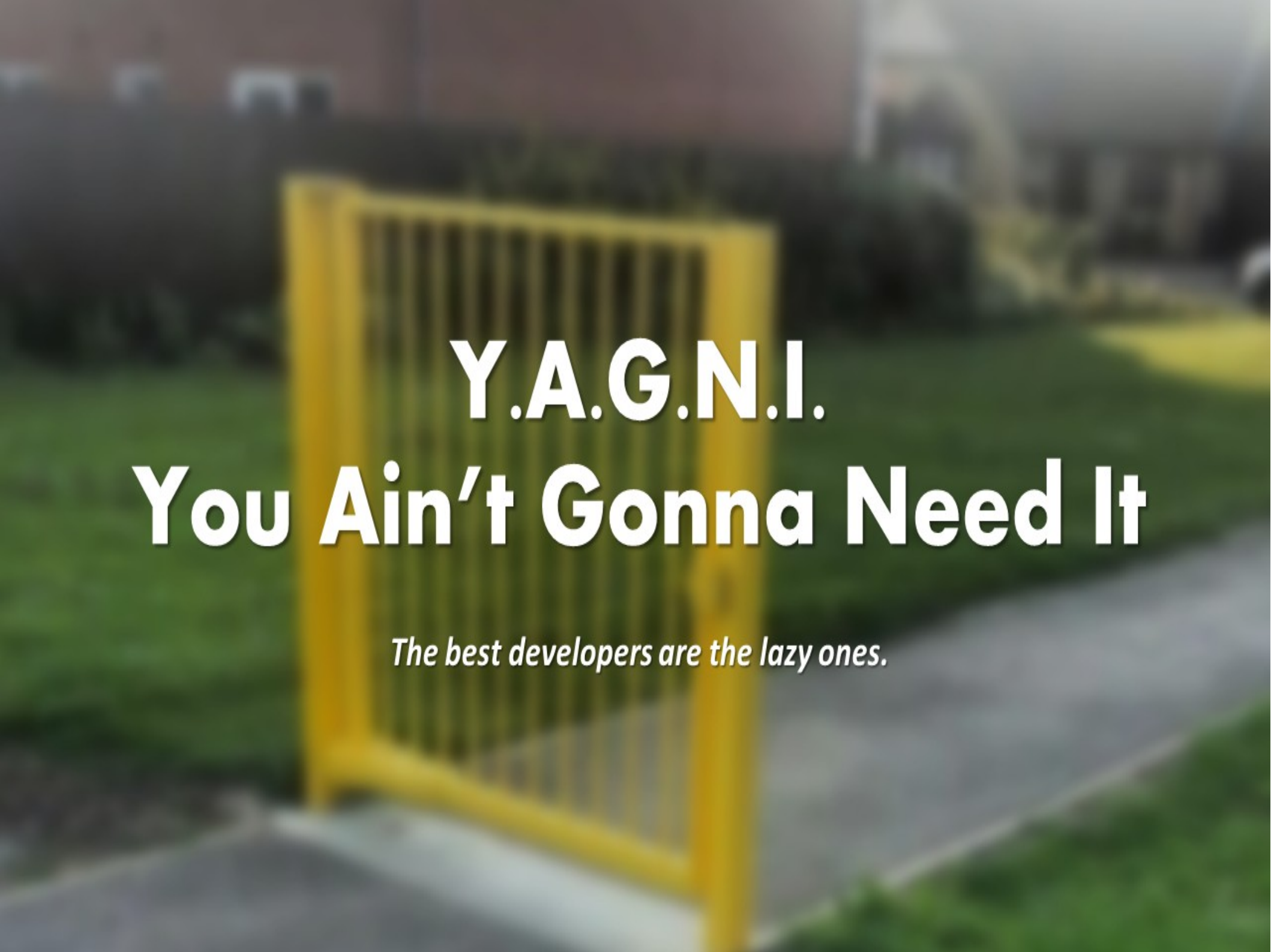
What is Clean Code?

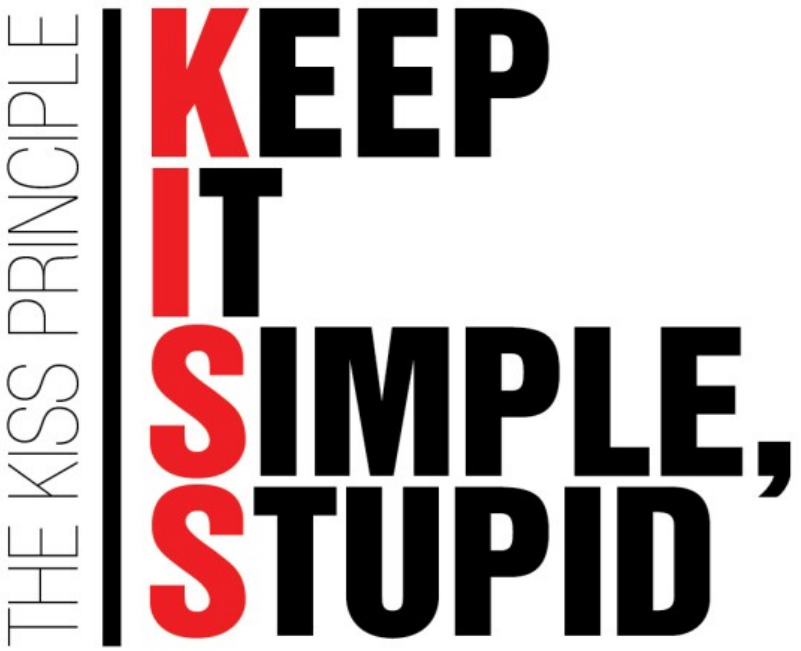
- It is easy to understand the execution flow of the entire application
- It is easy to understand how the different objects collaborate with each other
- It is easy to understand the role and responsibility of each class
- It is easy to understand what each method does
- It is easy to understand what is the purpose of each expression and variable
- Classes and methods are small and only have single responsibility
- Classes have clear and concise public APIs
- Classes and methods are predictable and work as expected
- The code is easily testable and has unit tests (or it is easy to write the tests)
- Tests are easy to understand and easy to change
- Clean Code is SOLID
- Clean Code reads like well-written prose

Why Clean Code Is Important?

- Your teammates will thank you
- Messy code tends to get messier
- Faster decision making
- Faster bug fixing
- Reusability
- Clean code leads to better practices
- It feels great!

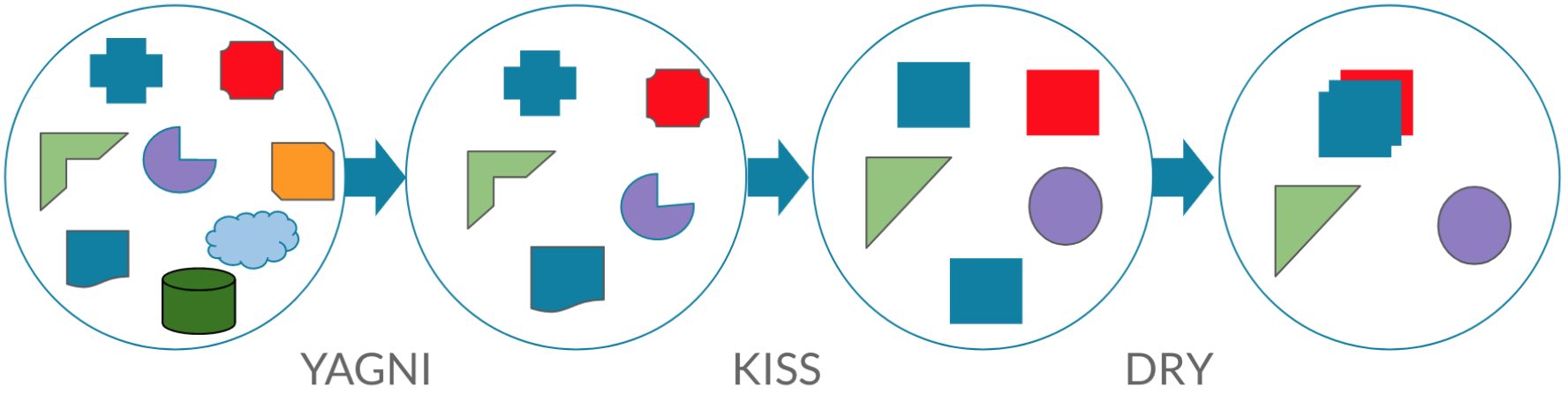
First Write Your Code Than Refactor It


Y.A.G.N.I.
You Ain't Gonna Need It
The best developers are the lazy ones.


THE KISS PRINCIPLE
KEEP
IT
SIMPLE,
STUPID

DRY

Don't Repeat Yourself


YAGNI
KISS
DRY

Broken Window Theory

THE BOY SCOUT RULE

ALWAYS
LEAVE
CODE
CLEANER
THAN YOU
FOUND IT!

Separation Of Concerns

- How much code you have to change?
- How easy it is to make the changes?
- How likely you are to break existing features that are being used by other customers
- How much you can reuse your existing model/architecture?

“Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.”

Martin Fowler

Clean Code vs Performance

Code Smells

- Rigidity
  - The design is hard to change
  - Sign: Huh, it was a lot more complicated than I thought.
- Fragility
  - The design is easy to break
  - Sign: some modules are constantly on the bug list
- Immobility
  - The design is hard to reuse
- Viscosity
  - Changes are easier to implement by doing the wrong thing (Hack)
  - Sign: When a change is needed, you are tempted to hack rather than to preserve the original design

Reasons Writing Bad Code Happens

- Deadline
- The Broken Window Theory
- Over Architecting (YAGNI)
  - You are not gonna need it
- Bad Design
- Poor Skill

Money

The Cost Of The Bad Code

Time, Productivity, Stress

All it is about the Money

How to write Clean Code?

Names

- Choose descriptive and unambiguous names.
- Make meaningful distinction.
- Use pronounceable names (modymdhms)
- Use searchable names.
- Replace magic numbers with named constants (P)
- Avoid encodings. Don't append prefixes or type information.
- Don't use a,b,c,i
- Class should be a noun
- method should be a verb
- Boolean names should answer Yes/No
- Pick one word per concept

You Think You Found A Better Name:

Rename It

Functions

- Small
- Smaller than that
- Do one thing
- Less 80 characters
- Less 7 line
- Don't use switch .. it violates SRP, OCP (Use Abstract Factory)
- Don't use else
- Arguments
- Don't be afraid to make a name long ... it's better than use short enigmatic
- Have no side effect
- Command query separation
- DRY code (Don't Repeat Yourself)
- KISS
- Don't afraid of new line
- Don't afraid of Exception

Class

- Should be small
- SRP (Single Responsibility Principle)
- Cohesion
- Loose Coupling (Use Dependency Inversion)
- Less than 300 line
- Use Your Conventions

Comment

- Don't comment bad code – rewrite it
- TODO comment

Formatting

- Consistency
- Vertical Formatting
- Horizontal Formatting
- PSR-12
  - https://www.php-fig.org/psr/psr-12/
- Team Rules

Single Responsibility

A Class, Module, Method Should Have Only One Reason To Change

References

- https://www.youtube.com/watch?v=TMuno5RZNeE
- Books
  - Clean Code – Robert C.Martin
  - Refactoring – Martin Fowler
  - Pattern Of Enterprise Application – Martin Fowler
  - Test Driven Development – Kent Beck
  - Working Effectively with Legacy Code - Michael C. Feathers
  - The Pragmatic Programmer - Andy Hunt and Dave Thomas
  - Head First Object-Oriented Analysis & Design - Brett McLaughlin
  - Head First Design Pattern - Elisabeth Freeman and Kathy Sierra
  - Design Patterns: Elements of Reusable Object-Oriented Software (GOF)
  - Agile Software Development, Principles, Patterns, and Practices – Robert C.Martin