Clean Code! The Fundamentals

Why Clean Code is essential.

1) Code is for humans
2) Quality
3) Compounds
4) Can't be delayed

Boy Scout Rule - Leave your code better than you found it.

Principle of Least Surprise

Every component should behave in a way that most users expect it to behave.

- Consumers should trust their intuition
- Obvious behaviors should be implemented

Coupling - Changing one element requires changing other elements

Cohesion - measures degree to which elements work together to achieve a goal.

- High Cohesion is valuable/desired
- Focused modules

Idiomatic Code - Having Conventions & Standards

- Community and Alignment
- Reduced Friction
- Consistency

Clean Code ! Naming

The Impact of Naming

- Names are everywhere
- Readability + Clarity depends on it.
- Intention revealing names
- If you cannot pronounce it, you cannot discuss it

Boolean

- Express as question (Is, Has)
- Avoid double negatives (IsNotEmployee)

Methods

- Stick to one convention (i.e. one of Get / fetch / Retrieve
- Use verbs
- Use clear names

Classes

- Use Nouns
- Be specific
- Avoid generic Suffixes

Writing Clean Methods
- Remove nestings
- Return early →
- Use guard clauses
  - Ardalis.Guard is an example
  - Guard.AgainstNull(request) → exception thrown
- Avoid functions w/ and or or in name.
  - Single Responsibility Principle
- Consider splitting methods w/ flag arguments in two.
- Use order to improve readability
- Ideal # of parameters is 0-3
  - Reduce parameter # w/ SRP + new object

Handling Errors
- Do not use exceptions to control flow
- Exceptions can be recoverable or unrecoverable
- Exceptions can be ignorable (should still log)

Dealing with NULLs
- Apply the Null object Pattern to represent "No Objects"
- Always return an empty collection instead of returning null

Clean Tests
- Consider fluent assertion package
- Descriptive titles and variables
- Use Arrange, Act, Assert structure
- Use real classes if not fragile, unstable or slow (avoid mocks)
- Use Builder classes to create objects in tests
- Use realistic test data aka "John Doe" ! "fjjjlll"

Tooling

- Use a Code Complexity Tool
  - Cyclomatic Complexity
  - Maintainability Index
  - Cognitive Complexity
- Use refactoring tools
  - Reshaper
  - Coderush
  - Rosslynator
  - JetBrains
- Use EditorConfigs to enforce standards
- Use a spell checker
- Enforce Code Quality
  - Sonarqube
  - Codacy