Design Pattern & Clean Code

Pattern, Clean Code and Best Practices

Thomas Ley | @CleanCodeCoach

Principles

- Don't Repeat Yourself (DRY)
- Keep it simple, [and] stupid (KISS)
- Beware of Optimizations
- Favour Composition over Inheritance (FCoI)
- Integration Operation Segregation Principle (IOSP).

Practices

- Boy Scout Rule
- Root Cause Analysis
- Version Control
- Simple Refactoring Patterns
- Daily Reflection.

Don't Repeat Youself (DRY)

- Valid principle since begin of software developement
- Refactoring over "copy & paste"
- Avoid fixing a bug multiple times
- Reduce lines of code.

Keep it Simple, [and] Stupid (KISS)

"Everything should be done as simple as possible, but not simpler"

`(Albert Einstein)`

- Simple, clear and easy to understand
- Avoid over-engineering
- Related to "YAGNI"
- "Think Big, Start Small"
- Question: Is my solution expandable?.

Beware of Optimizations!

Rules of Optimization:

Rule 1: Don’t do it.

Rule 2 (for experts only): Don’t do it yet. `(M.A. Jackson)`

- Understandability over optimization
- Related to "YAGNI"
- "Premature optimization is the root of all evil." `(Donald Knuth)` .

Favour Composition over Inheritance (FCoI)

- Reuse functionalities through
  - Inheritance (whitebox reuse)
  - Composition (blackbox reuse)
- Inheritance is unnecessary complex
- Inheritance is poorer to test
- Inheritance is makes it difficult to switch functionality at runtime
- Related to "LSP".

Integration Operation Segregation Principle (IOSP)

- Separate logic and method calls
  - A method contains exclusively logic (transformations, control structures or API invocations) is called an **Operation**
  - A Method does contain exclusively calls other methods within its code basis is called **Integration**
- Helps to keep methods short
- Short methods are easier to test and understand
- **Integration** methods tells the business process.

Boyscout Rule

"Always leave the campground [code] better than you found it."

`(Boyscout Handbook)`

- Remove technical debts
- Renaming, documentation, tests, or
- Clean(er) Code
- Small or smaller, depends on you.

Root Cause Analysis

- Don't fix symptoms (only short term)
- Eliminate the root cause (no further symptoms)
- "Five Why's" `(Toyota Production System)`.

Version Control (e.g. Git)

- Try something and discard
- Commit often (you can squash later)
- Delete old code without losing it.

Simple Refactoring Patterns

- Refactoring => cleaner code
- e.g. extract method
- e.g. rename variables/properties/methods/classes
- Related to Boyscout Rule.

Daily Reflection

- For red grade this would be questions as
  - Are all code fragments under version control?
  - Did I apply the DRY-Principle consequently?
  - Did I in general leave code in a better state as I found it?
- Kind of "daily review".

Learn from yesterday,

live for today,

hope for tomorrow.

The important thing is not to stop questioning [improving oneself].

(Albert Einstein)