Clean Code

How to develop software and not turn mad

CIKLUM

Krystof Polansky

Why Clean Code

- Saves time and effort
- Enables painless change of behavior
- Reduces complexity
- Development is easier
- Teaches many principles of good design
- Uses different code smell analysis to discover problems in design
- Teaches to use Test Driven Development because it forces you to avoid bad design decision right in the beginning

Example 1

- Do you know what this code does?
- Can you read the code easily?

Example 1

- Clean code version
- Can you understand the code better now?
- So what is the difference between the clean code version and non-clean code version?
- What principle of clean code is applied here?

Example 1 - Naming

- Very basic tool, but still very powerful
- Everything has a name : packages, classes, methods, variables, if ()
- Avoid magic constants:

```
report.getItems().entrySet().stream()
.filter(entry -> entry.getValue() > 5)
```

- Good name shows the meaning

Naming

- Prefer longer and expressive names over short and meaningless:

```
FileProcessor.process()
FileProcessor.removeInvalidCharsAndSave()

TemplateHandler.handle()
TemplateHandler.fillTemplateWithValues()
String newContent = ….
String normalizedContent = ….
```

Example 2

- Can you understand this code?
- What are you concentrating on?

Example 2 – clean code version

- Can you understand this code better?
- What are you concentrating on?
- So what is the difference between the clean code version and non-clean?
- What principle of clean code is used here?

Example 2 – One level of abstraction per method

- Abstraction: WHAT ? And HOW?
- I want to know WHAT the code does. I don’t care HOW it is done.
- Every method should list a few things WHAT happen.
- Every WHAT thing is a call of another method and the method name explains WHAT happens in that method
- If you want to know HOW that happens, go to that method and see

WHAT things happen in that method.

Example 2 – One level of abstraction per method

Structure of methods should be a Tree. Top level methods = high level concepts, no details. Leaf methods = details.

Open Closed Principle

- A module follows the open closed principle when:

1. It is Open for extension:

Customer needs new behavior. It is possible to add new behavior / change existing behavior of the module

2. Is Closed for modification:

“Extending the module does not need change of the source code of the module. The .jar library does not need to be updated, recompiled, redeployed etc.”

Open Closed Principle

- Paradox: How can we change the behavior of the module without changing the source code?
- Abstraction is the key

Example 3 – Open Closed Principle

- Ocp.pokus1 – pokus 4

Code Smell – Signs of bad design

- 1. Rigidity - the system is hard to change because every change forces other changes of other parts of the system.
- 2. Fragility – Changes cause the system to break in places that have no conceptual relationship to the part that was changed.
- 3. Immobility – It is hard to disentangle the system into components that can be reused in other systems.
- 4. Viscosity – Doing things right is harder than doing things wrong.

Code Smell – Signs of bad design

- 5. Needless complexity – The design contains infrastructure that adds no direct benefit.
- 6. Needless repetition – The design contains repeating structures that could be unified under a single abstraction.
- 7. Opacity – It is hard to read and understand. It does not express the intent well.

How to solve code smell & Resources

- Read these two books and use the principles of clean code :)

- Robert C. Martin: Agile Software Development. Principles, Patterns and Practices.
- Robert C. Martin: Clean Code. A Handbook of Agile Software Craftsmanship.

Code Examples

- All code examples are available in project
- https://github.com/krystofpo/clean-code2