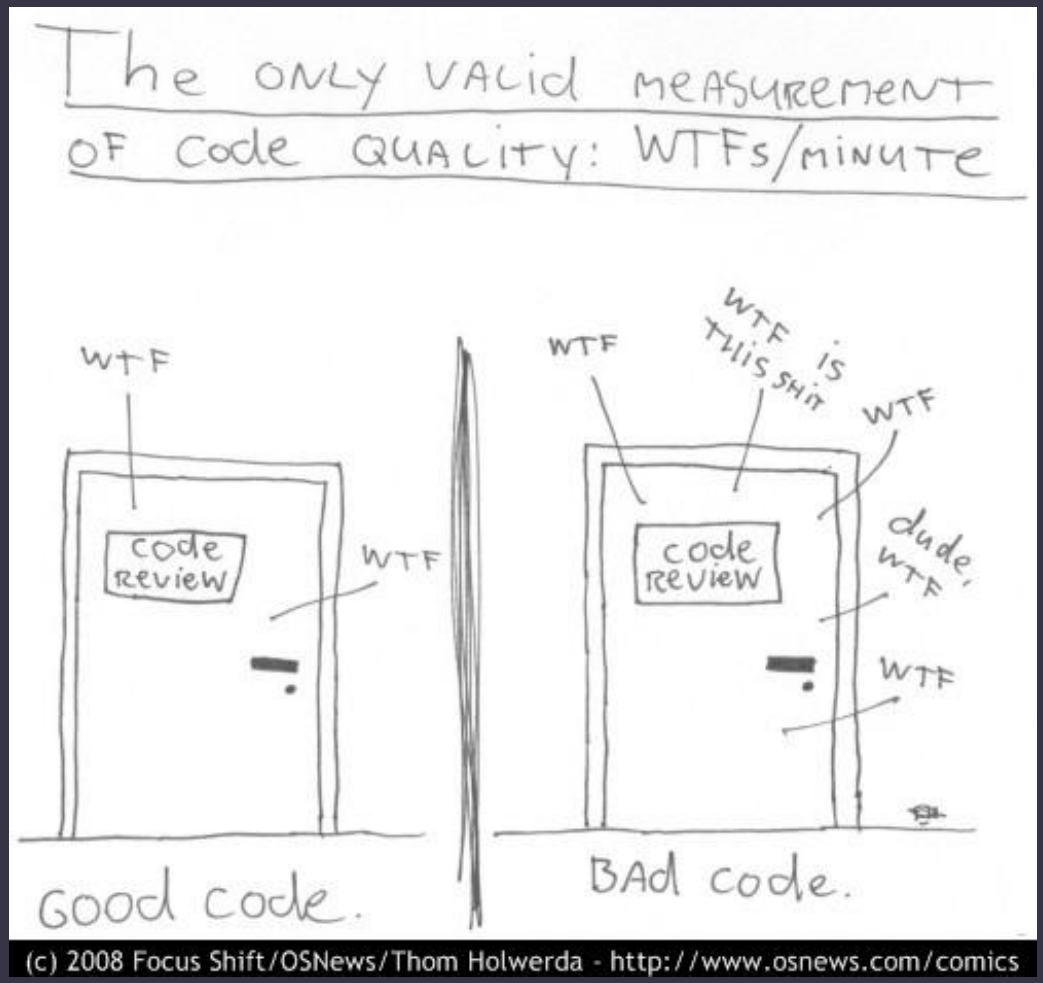
# Clean Code

E-Portfolio - Celina Adam

28.05.2019


The ONLY VALid MEASUREMENT OF Code QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics

# Content

- Why are we writing bad code?
- Why should we write clean code?
- What is clean code?
- How can we achieve clean code?
- Demo

"My code is working well, the website I built is looking great, and my client is happy. So why would I still care about writing clean code?"

# Why are we writing bad code?

# Why are we writing bad code?

- Bad code isn‘t immediately evident to the customer
- The user only cares for a working application
- Strict schedule and stress
- Pressure from superiors (you don’t get the time to refactor your code because it is working)
- New features have a higher priority

# Why would I still care about writing Clean Code?

***"Programming is the art of telling another human what one wants the computer to do."***

*— Donald Knuth*

**Readability**

**Comprehensibility**

**Maintainability**

# What is Clean Code?

***"If you want your code to be easy to write, make it easy to read"***

*— Robert C. Martin*

***"Clean code always looks like it was written by someone who cares. There is nothing obvious you can do to make it better."***

*— Michael Feathers*

***"Clean Code is simple and direct."***

*— Grady Booch*

# What is Clean Code?

*„Clean code is code that is easy to understand and easy to change."*

- The code has to be easy to read and easy to understand on every level.
- The code has to be easy to extend and refactor.
- It has to be easy to fix bugs.

# How can we achieve Clean Code?

- Coding Principles
- Helpful Mentalities
- Refactoring

# Coding Principles

- DRY (Don‘t repeat yourself)
- Reduce dependencies as much as possible
- KISS (Keep It Simple, Stupid)
- SRP (Single-Responsibility Principle)
- YAGNI (You Aren’t Gonna Need It)

# Helpful Mentalities

- Test as you write
- Validate your data
- Handle errors nicely
- Tidy up after yourself
- Make your code unsurprising

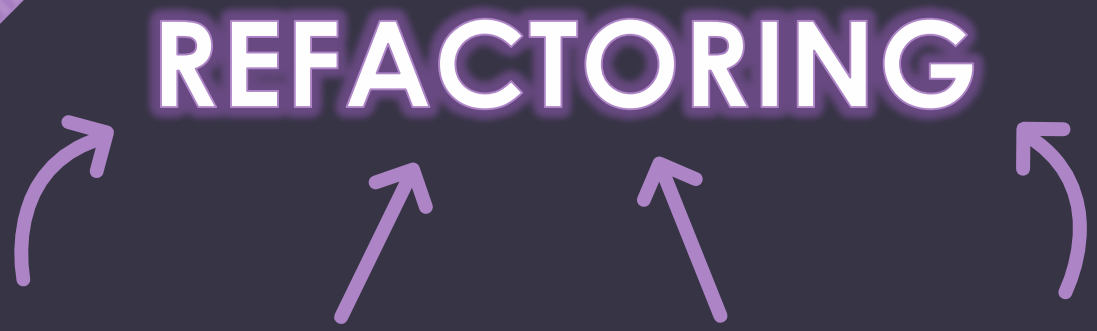
# Refactoring

*„[P]rocess of changing a software system*
*in such a way that it does not*
*alter the external behavior of the code*
*yet improves its internal structure.“*

# Refactoring

1. Writing a solid test suite
2. Extract methods
3. Rename variables/classes/…
4. Move methods to their corresponding classes
5. Removing temporary variables
6. Reducing complexity of classes and methods (e.g. with polymorphism)
7. Shorten parameter list
8. Transform error codes into exceptions

# When do we refactor?




- Copy & Paste?
- Having to check a method or class to find out what it does?
- You need to write or read comments to understand the method?
- Bad metrics?

# Kata

- Name comes from the japanese martial arts
- Exercise in programming
- Term used by Dave Thomas (1999)
- Helps hone your skills through practice and repetition
- Often combines Clean Code with Test Driven Development

# Interested in more?

https://ccd-school.de/coding-dojo/class-katas/galgenmaennchen/

https://www.itexico.com/blog/software-development-kiss-yagni-dry-3-principles-to-simplify-your-life

https://clean-code-developer.de/

https://cvuorinen.net/2014/04/what-is-clean-code-and-why-should-you-care/

http://www.hurricanesoftwares.com/most-important-coding-principles/

https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf

# Thanks for listening!