Refacturing with Code Smells

Michael Meissner 13.03.2023

Agenda

- Problems with „old“ code
- Core OO Priniciples
- Code Smells

Refacturing

How to eat an elephant?

There is only one way to eat an elephant,
a bite at a time.

Desmond Tutu

Code Smells shows us the path …

„Old“ Code
Potential problems:

- **Rigidity**: When the software is difficult to change.
  A small change triggers a cascade of changes.
- **Fragility**: When a single change breaks the software
  in many different places.
- **Immobility**: When code is not reusable.
- **Viscosity of Design:** When design is not simple enough
  it introduces the hazard of taking shortcuts and
  introducing technical debt.
- **Viscosity of Environment:**
  When the environment is not effective enough
  (slow, flaky tests, clunky CI, test environments limitations, etc.)
  it introduces the hazard of not doing those activities properly by
  everyone and introducing technical debt as a consequence

Where we want to go?

Maximize
Cohesion

Minimize
Coupling

What can help us?
OO Core Principles

- KISS : Keep ist Simple
- DRY: Don’t repeat yourself
- YAGNI: you aren’t gonna need it
- LEAST ASTONISHMENT (WTF PRINCIPLE)
- Law of Demeter: Tell don’t ask
- SOLID
  - Single responsibilty Principle
  - Open-Closed Principle
  - Liskov Substitution Principle („has“ vs „is“)
  - Interface Segregation Principle
  - Dependancy Inversion Principle
- and more

What is the hardest Principle?

In my opinion:
KISS - Keep it simple …

Everybody understands
something else of „Simple“.

Code Smells

Dispensables

A dispensable is something pointless and unneeded whose absence would make the code cleaner, more efficient and easier to understand.

- § **Comments**
- § **Duplicate Code**
- § **Data Class**
- § **Dead Code**
- § **Lazy Class**
- § **Speculative Generality**

Code Smells

Bloaters

Bloaters are code, methods and classes that have increased to such gargantuan proportions that they are hard to work with. Usually these smells do not crop up right away, rather they accumulate over time as the program evolves (and especially when nobody makes an effort to eradicate them).

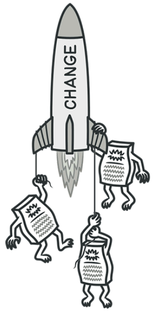
- § Long Method
- § Primitive Obsession
- § Data Clumps
- § Large Class
- § Long Parameter List

Couplers

All the smells in this group contribute to excessive coupling between classes or show what happens if coupling is replaced by excessive delegation.

- § Feature Envy
- § Incomplete Library Class
- § Middle Man
- § Inappropriate Intimacy
- § Message Chains

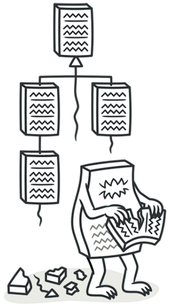
Code Smells



Change Preventers

These smells mean that if you need to change something in one place in your code, you have to make many changes in other places too. Program development becomes much more complicated and expensive as a result.

- Divergent Change
- Parallel Inheritance Hierarchies
- Shotgun Surgery

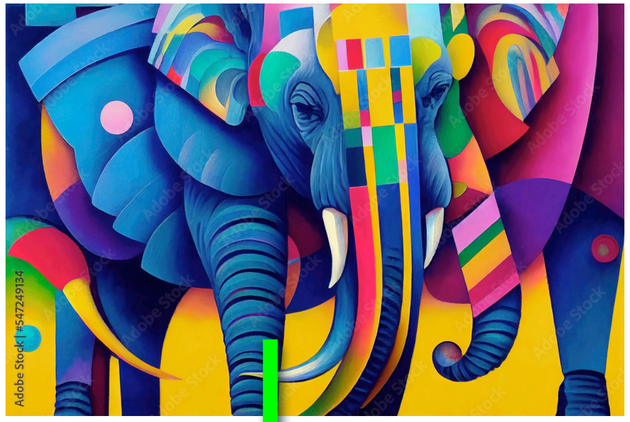


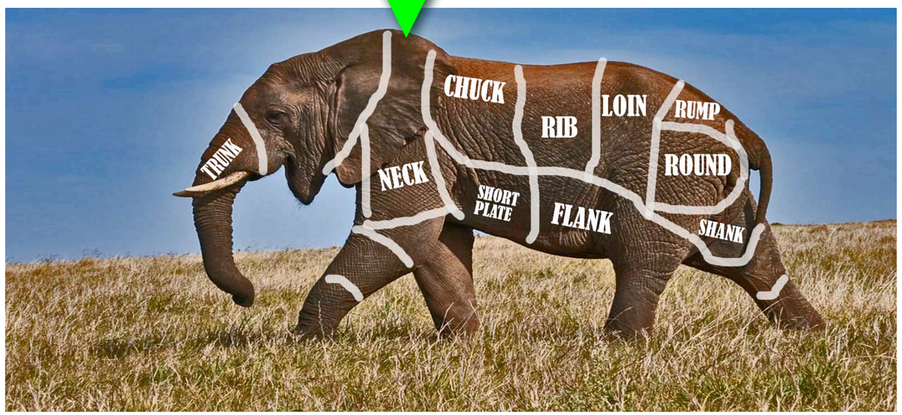
Object-Orientation Abusers

All these smells are incomplete or incorrect application of object-oriented programming principles.

- Alternative Classes with Different Interfaces
- Refused Bequest
- Switch Statements
- Temporary Field

Conclusion


Adobe Stock | #547249134


TRUNK
NECK
CHUCK
RIB
LOIN
RUMP
ROUND
SHORT PLATE
FLANK
SHANK

Questions

References

* Alcor Academy Presentation
* CodeSmells : https://refactoring.guru/refactoring/smells

* Image Elephant: https://www.megafishbein.com/blog/2015/9/15/eating-the-elephant-mastering-your-universe-one-hack-at-a-time

* Image Elephant 2: https://collectiver.com/2018/08/31/project-management-eating-an-elephant/

* Ice Tunnel: © Michael Meissner

Thank You

Michael Meissner

Email: michael.meissner@css.ch

Appendix

Checkliste
- Agenda optional
- Persönlich
- Bilder
- Wenig Text
- Animationen
- Code auf weissen Hintergrund
- Frageseite
- Referenzen
- Danke mit Kontaktangaben