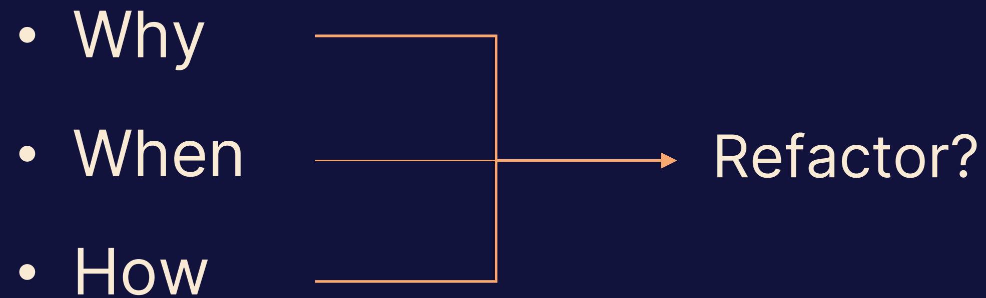
REFACTORING

Definition:
Restructure (code) so as to improve operation, without altering functionality

Simon Austnes

Summary of learnings



bouvet

«If it ain't broke, don't fix it»

- Is it worth your time?
- proverb of the lazy
  - But... I am lazy?

bouvet

So, why should we bother?

bouvet

Stinky code

- Change preventers
- Bloaters
- Object-orientation abusers
- Couplers
- Dispensables

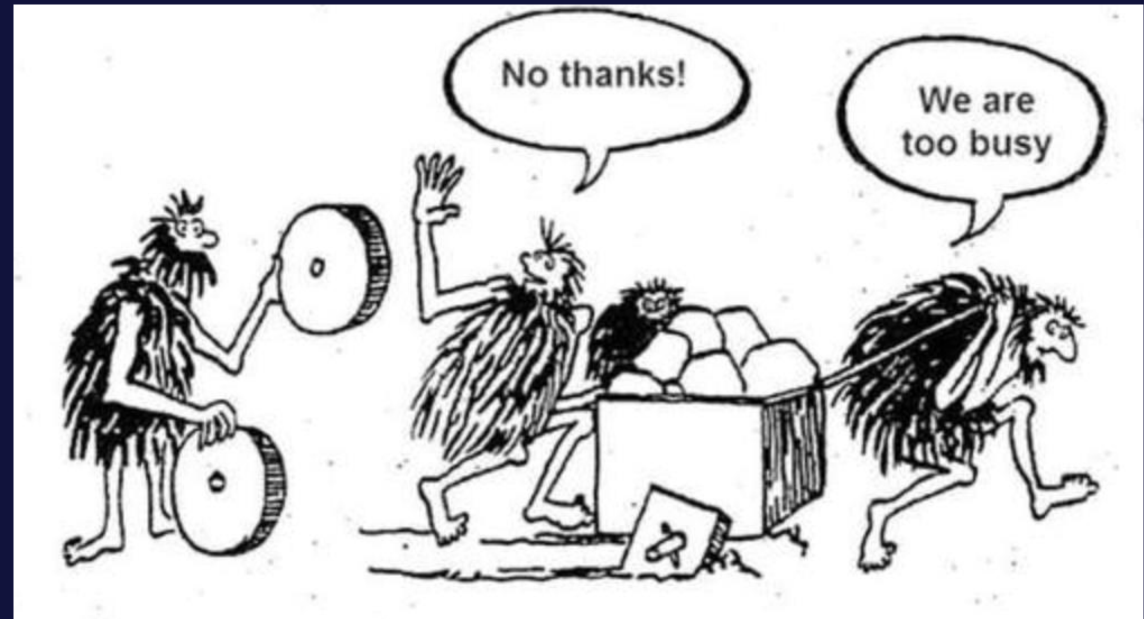
Object calisthenics →	Code smells
Only one level of indentation per method	Long Method
Don't use the ELSE keyword	Long Method / Duplicated Code
Wrap all primitives and strings	Primitive Obsession
First class collections	Divergent Change / Large Class
One dot per line	Message Chains
Keep all entities small	Large Class / Long Method / Long Parameter List
No classes with more than two instance variables	Large Class
No getters / setters / properties	Feature Envy
All classes must have state, no static methods, no utility classes	Lazy Class / Middle man / Feature envy

Source: page 16 of Lesson 2 – Code smells

So, why should we bother?

Causing

- Rigidity
- Fragility
- Immobility
- Viscosity
  - of design
  - of environment



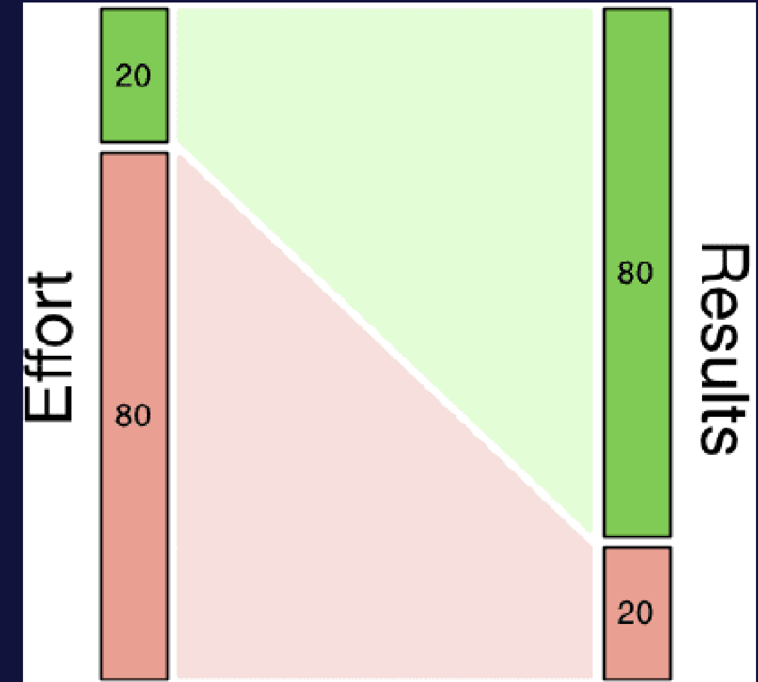
Source: http://www.hamiltonclaimssolutions.co.uk/blog/if-it-isnt-broken-dont-fix-it

bouvet

So, why should we bother?

- Pareto principle

- Afraid of making changes?
  - Write tests!



Source: page 4 of Lesson 1 – Introduction to refactoring

bouvet

When is it needed?

- Rule of 3
- Breaking rules of object calisthenics

- Follow the wise words of Marco and Alessandro
  - Refactor aggresively and constantly

bouvet

Guidelines!

«Refactor not because you know the right abstraction, but because you want to find it.»
- Martin Fowler

- Single responsibility principle
- Open-closed principle
- Liskov substitution principle
- Interface segregation principle
- Dependency inversion principle

- Minimize coupling
- Maximize Cohesion

bouvet

Guidelines!

«Refactor not because you know the right abstraction, but because you want to find it.»
- Martin Fowler

- Refactor readability before design
  - 80/20
- Parallell change
  - Expand, migrate and contract
- IDE agility
  - Use the tools at your disposal

bouvet

Rule #1 of refactoring

- Perserve behavior
  - If you find a bug, consider it a feature

bouvet

Thanks for your attention!

simon.austnes@bouvet.no