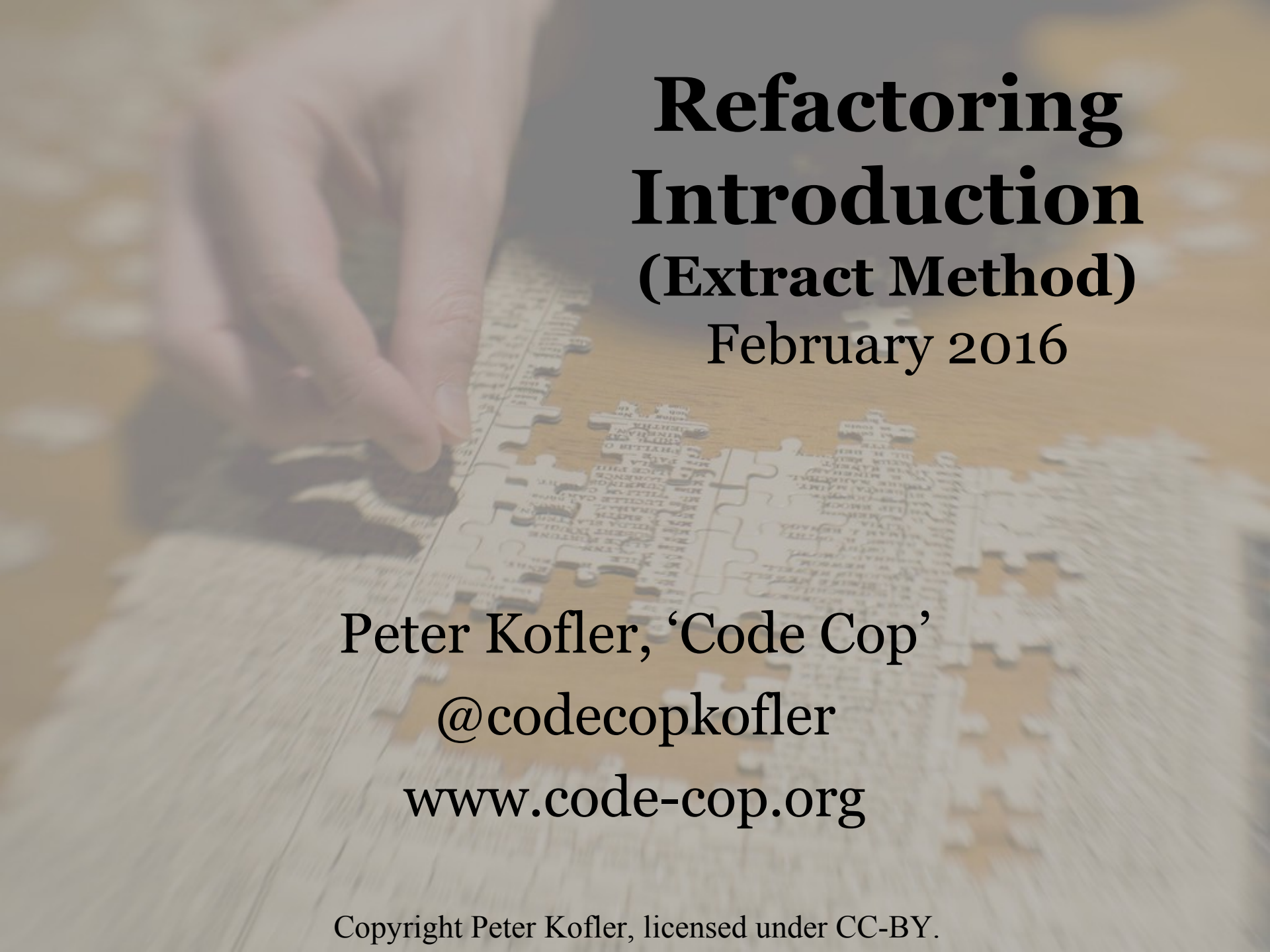


Refactoring Introduction (Extract Method)

February 2016

Peter Kofler, 'Code Cop'
@codecopkofler
www.code-cop.org

Peter Kofler

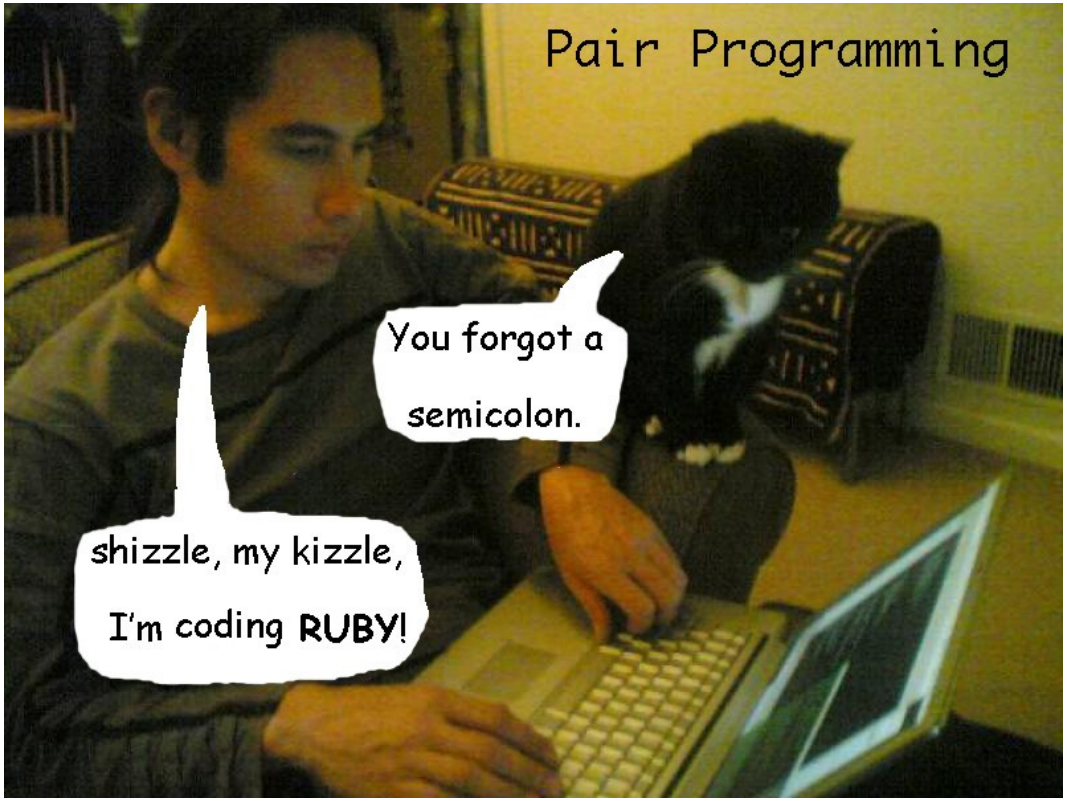
- Ph.D. (Appl. Math.)
- Professional Software Developer for 15 years
- “fanatic about code quality”
- Freelance Code Mentor

I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement

Mentoring

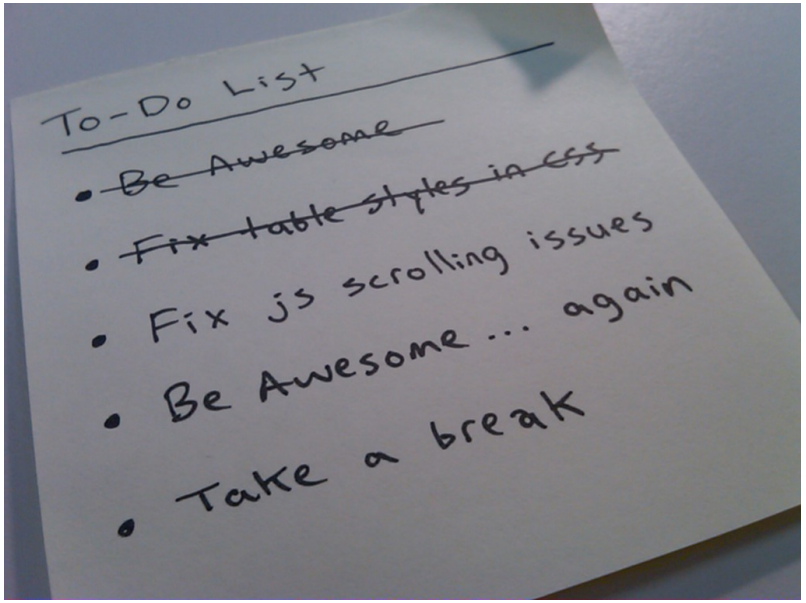
- Pair Programming
- Programming Workshops
- Deliberate Practice, e.g. Coding Dojos



Developing Quality Software Developers

Workshop Structure

- Simple Design
- Code Smells
- Refactoring
- Refactor manually
- Refactoring tools (i.e. PHPStorm)



Coding Dojo Mindset

- Safe place outside work
- We are here to learn
- Need to slow down
- Focus on doing it right
- Collaborative Game

Software Design

Why Software Design?

Software Design Enables Change

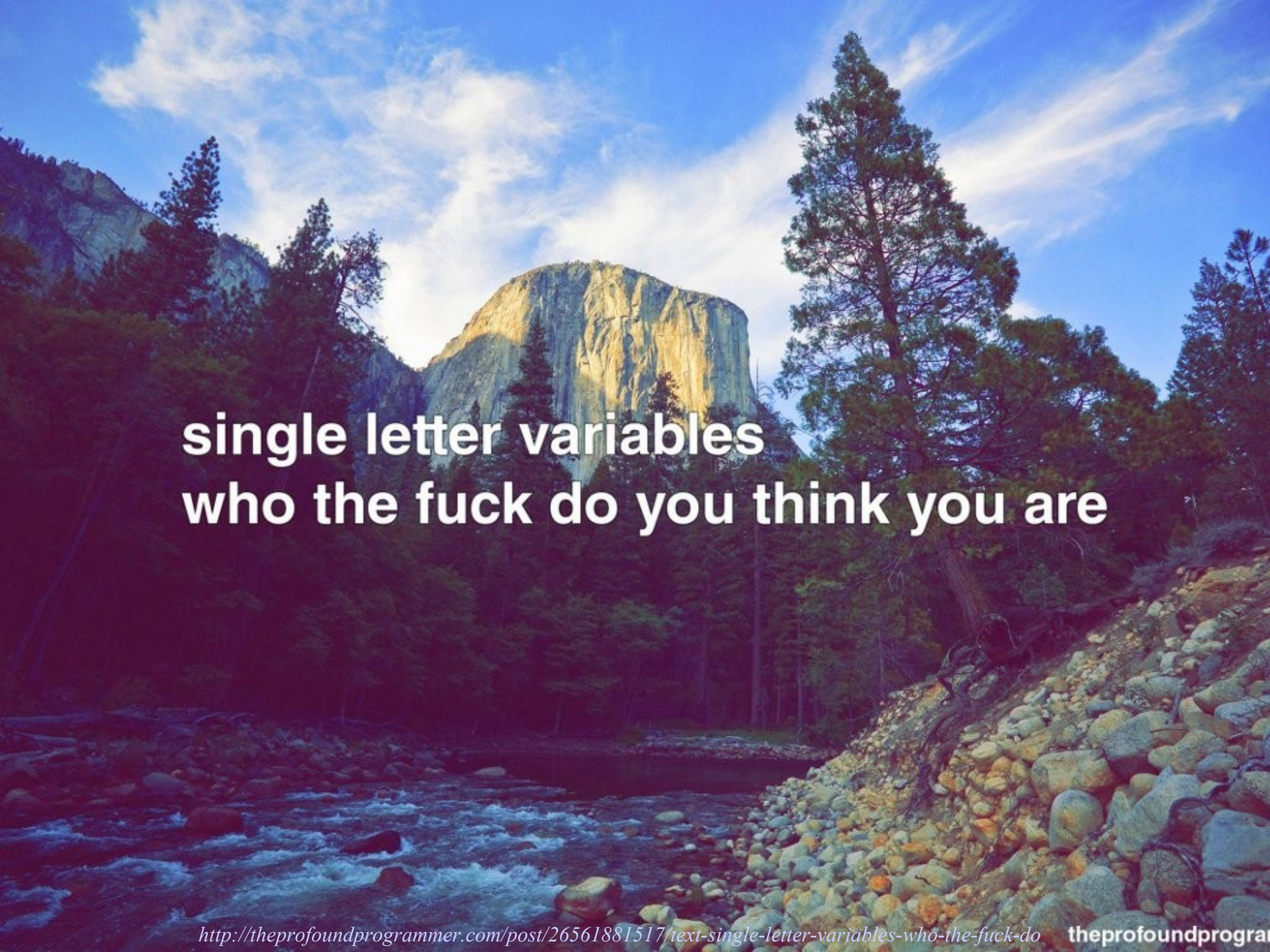
Four Rules of Simple Design

(1) Passes the tests

(2) No duplication

(3) Reveals intention

(4) Fewer elements


single letter variables
who the fuck do you think you are
http://theprofoundprogrammer.com/post/26561881517/text-single-letter-variables-who-the-fuck-do
theprofoundprogra

... that's a Code Smells

Code Smell

- “a surface indication that usually indicates a deeper problem in the system.“
- quick to spot
    - e.g. bad names
    - e.g. long method
    - e.g. duplication
- does not always indicate a problem

List of Code Smells (Fowler)

The Bloaters
- Long Method
- Large Class
- Primitive Obsession
- Long Parameter List
- DataClumps

The Object-Orientation Abusers
- Switch Statements
- Temporary Field
- Refused Bequest
- Alternative Classes with Different Interfaces

The Change Preventers
- Divergent Change
- Shotgun Surgery
- Parallel Inheritance Hierarchies

The Dispensables
- Lazy class
- Data class
- Duplicate Code
- Dead Code
- Speculative Generality

The Couplers
- Feature Envy
- Inappropriate Intimacy
- Message Chains
- Middle Man

Comment explaining section of code

- code grouped into blocks and there is a comment above each few lines of code

```
# delete previously converted BAS
File.delete PRG if File.exist? PRG
Dir['*.BAS'].each { |bas| File.delete bas }

# convert basic PRG to readable format
Dir['*.PRG'].each do |prg|
  # handles only 8.3 names, rename
  File.rename(prg, PRG)

  puts "converting #{prg}" +
       `#{CBM2ASC} #{PRG} #{BAS} b`

  File.rename(BAS,
              prg.sub(/\.PRG$/i,'.BAS'))
end

# delete converted PRG
Dir['*.PRG'].each { |prg| File.delete prg }
```

Duplicate Code

- same code found in 2 or more places
  - e.g. formatting, is user in session?
- same expression found in 2 or more places in the same function
- constant value found in 2 or more places
  - e.g. items per page, default encoding

Method does too much

- has more than 5 lines of code
    - If in doubt – it's too big.

- does different things
    - e.g. DB and UI
    - has strange name or
    - has 'and' in its name

Code Smell Exercise

- Group into pairs.
- Get the printout of the Tennis code.
- Read the code carefully.
- Mark found code smells.
    - where? and which?
    - maybe more smells in one line
- We found **25**, how many can you find?

How do we fix it?

Refactoring

Definition

Refactoring is a technique
for **restructuring**
an existing body of code,
altering its internal structure
without changing
its external behaviour.

(Martin Fowler)

Small Transformations

- behaviour preserving transformation
- each transformation is small, less likely to go wrong
- system is fully working after each change
- verified by working tests
- sequence of transformations produce a significant restructuring

Extract Method

- Create a new method, name by intention
- Copy extracted code from source method to new method
- Scan for local variables.
    - Temporary variables local to method?
    - Local-scope variables modified?
      Return changes back to parent method.
    - Pass local-scope variables as parameters.
- (Compile.)
- Replace extracted code with call to new method.
- (Compile and) test.

Introduce (Explaining) Variable

- Declare a (final) temporary variable.
- Set it to the result of part of the expression.
- Replace the result part of the expression with the value of the temp.
- If the result part of the expression is repeated, you can replace the repeats one at a time.
- (Compile and) test.
- Repeat for other parts of the expression.

Refactoring Exercise

- Group into new pairs (with computer).
- Go to the Tennis project.
- Run `./vendor/bin/phpunit`
- Open `TennisGame2.php` in **text editor**.
- Extract >= 2 methods (by hand).
- Extract >= 2 local expressions (by hand).
- Extract >= 2 constant (by hand).

Refactor Mercilessly

Seriously ;-)

- mer·ci·less is defined as having or showing no mercy, cold-blooded, hard-boiled, heartless, insensitive, hard, pitiless, remorseless, **ruthless**, **slash-and-burn**, soulless, **take-no-prisoners**, unfeeling, unsympathetic

- e.g. „extract till you drop“
  https://sites.google.com/site/unclebobconsultingllc/one-thing-extract-till-you-drop

PROFESSIONELLE SOFTWAREENTWICKLUNG

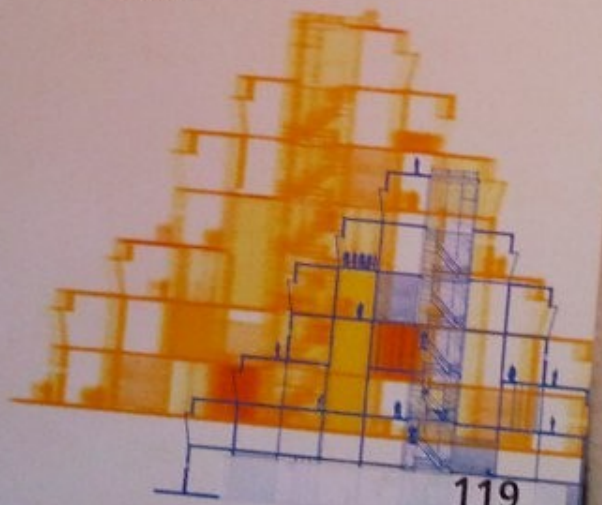
Martin Fowler

Refactoring

Wie Sie das Design vorhandener Software verbessern

Mit Beiträgen von Kent Beck, John Brant, William Opdyke, Don Roberts

Vorwort von Erich Gamma

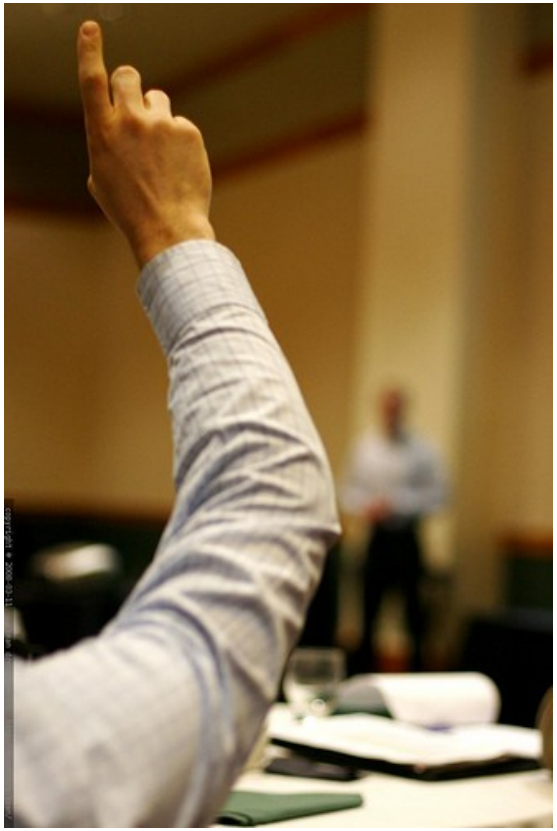


119

ADDISON-WESLEY

Refactoring Questions

- How long is a long method?
- How do we refactor?
- During refactoring, how long does code not compile?
- When to run the tests?



Refactoring with Tools

Automatic Refactorings (PHPStorm)

- Rename
- Change Signature
- Move
- Extract Variable
- Extract Constant
- Extract Field
- Extract Method
- Extract Parameter
- Inline
- Safe Delete

- Copy/Clone
- Extract Interface
- Pull Members up
- Push Members down

JavaScript:

- Change Signature
- Extract Variable
- Extract Parameter

- (Format)

Demo

(Refactoring in PHPStorm)

PHPStorm Exercise A

- Group into new pairs.
- Open Gilded Rose project in PHPStorm.
- On folder `test`, select Run Tests
- Open `gilded_rose.php`.
- Clean up in small steps.
- Try to use only automatic refactorings.
- Run tests often.
- Commit to Git whenever tests pass.

What you could do

- Make it more readable.
- Remove duplication (extract duplicates).
- Split into logically coherent blocks.
- Simplify complex boolean conditions.

- Bonus Round: Replace duplicated if-statements with polymorphism (extract Strategy pattern).

PHPStorm Exercise B

- Group into new pairs.
- Clone Yatzy project in PHPStorm.
- On folder `php`, run tests `YatzyTest.php`.
- Open `yatzy.php`.
- Clean up in small steps.
- Try to use only automatic refactorings.
- Run tests often.
- Commit to Git whenever tests pass.

What you could do

- Make it more readable.
- Remove duplication (extract duplicates).
- Split into logically coherent groups of methods.

- Bonus Round: Separate low level couting from high level game rules (extract class).

Closing Circle

- What did you learn today?
- What surprised you today?
- What will you do differently in the future?

created by Peter Kofler
@codecopkofler
www.code-cop.org

with help from Aki Salmi
@rinkkasatiainen
https://about.me/rinkkasatiainen

all katas by Emily Bache
@emilybache
http://coding-is-like-cooking.info/2011/08/refactoring-kata-fun/

CC Images

- puzzle https://www.flickr.com/photos/lizadaly/2944362379/
- todos http://www.flickr.com/photos/kylesteeddesign/3724074594/
- dojo http://www.flickr.com/photos/49715404@N00/3267627038/
- wants you http://www.flickr.com/photos/shutter/105497713/
- boys https://www.flickr.com/photos/andymorffew/16925347231/
- smells http://www.flickr.com/photos/hhbw/4215183405/
- exercise https://www.flickr.com/photos/sanchom/2963072255/
- mercy http://www.flickr.com/photos/williac/99551756/
- questions http://www.flickr.com/photos/seandreilinger/2326448445/
- tools https://www.flickr.com/photos/tom-margie/5019211728/