

Coding Dojo: Refactoring the Tennis Kata

October 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”
- I help development teams



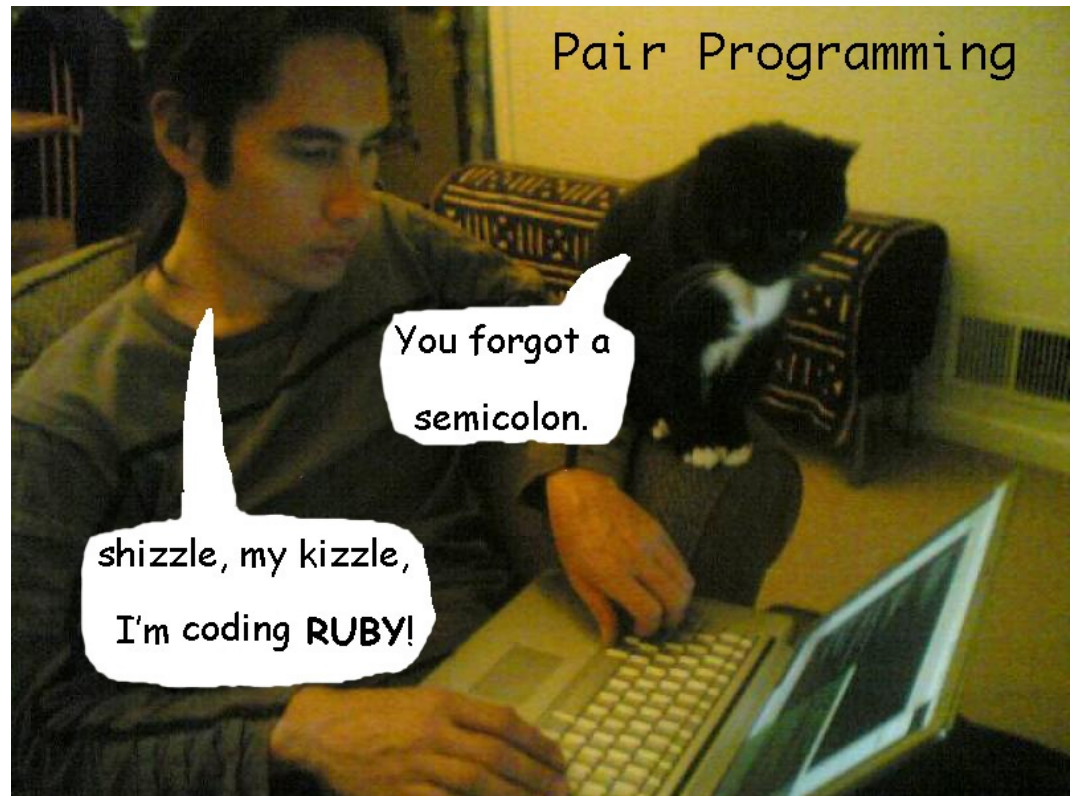
I help development teams with

- Professionalism
- Quality and Productivity
- Continuous Improvement



Mentoring

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



Developing Quality Software Developers

Coding Dojo?
Expectations?

Coding Dojo Mindset

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



Dojo Structure

- Introduction 15'
- Coding 60'
- Retrospective 15'
- Break 15'
- Coding 60'
- Retrospective 15'
- etc.



Refactoring

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

(Martin Fowler)

Identify Code Smells

- Code Smells are indications of a deeper problem in the system.
- e.g.
 - bad names,
 - duplication,
 - call chains,
 - missing object orientation,
 - etc.




Refactor Mercilessly

- 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>

TDD Skills

- Driving Development with Tests
- Designing Test Cases
- Designing Clean Code
- Refactoring Safely 

Assignment: Tennis



Scoring a Tennis Game

- The Scoring Algorithm is done.
- Covered by a complete suite of test.
- You need to **clean-up** the code.
- Refactor Mercilessly!
- Work in small steps. („Baby Steps“)
- Rely on the tests, run them often.
- Experiment with approaches.

Constraints

- Challenges during a dojo or coderetreat
- Moving to the extreme is a way of learning
- Examples
 - Missing Tool (No Mouse, ...)
 - Missing Feature (No IFs, ...)

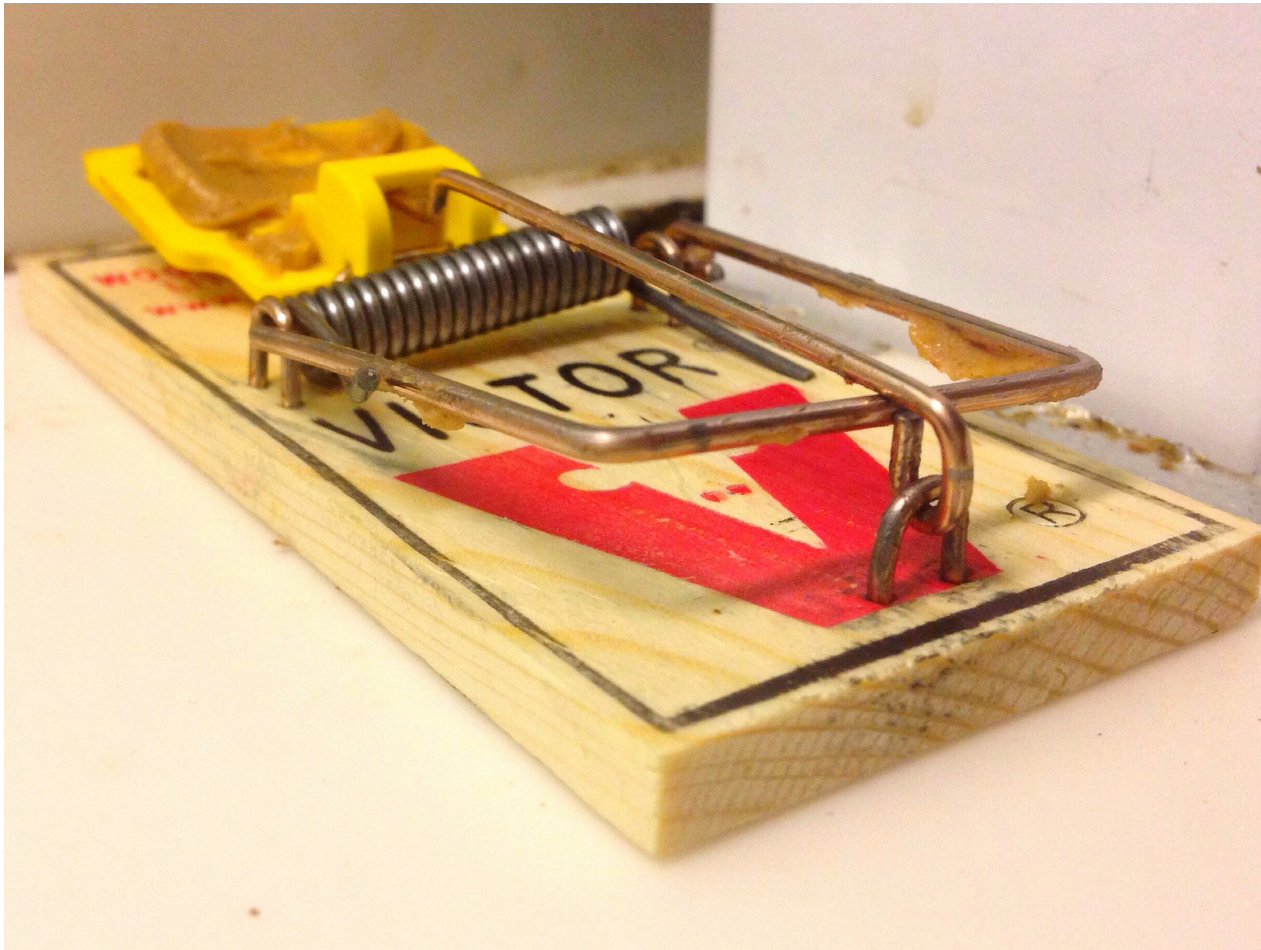
Only Use Refactoring Tools



Constraint: Refactoring Tools

- Using automatic refactoring is less risky.
- Only use refactoring tools of your IDE.
 - e.g. Extract Variable, Inline Method, ...
- If you (have to) change code manually
 - undo that change and
 - think of a way to do the same change
 - with a series of automatic refactorings.

Constraint: No Mouse



Prepare

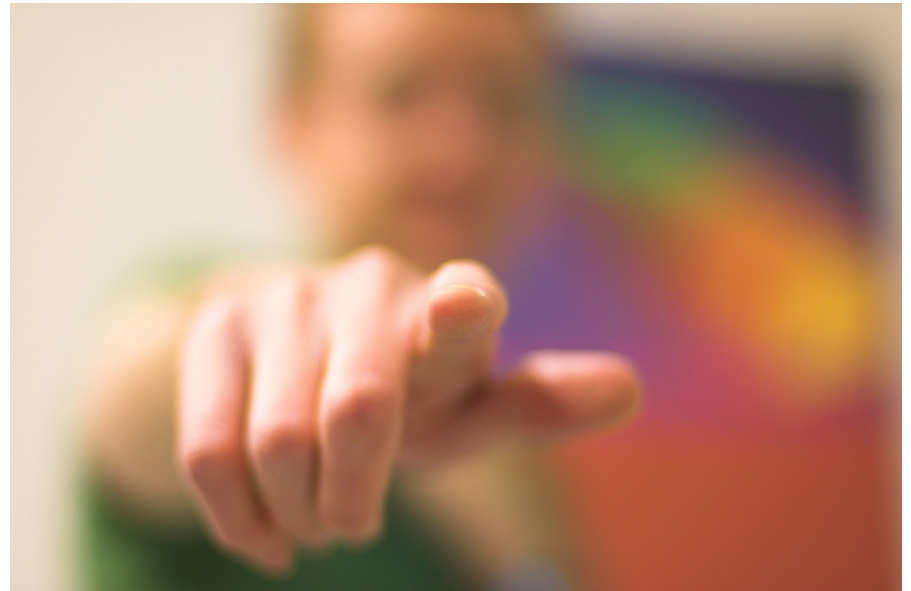
- Find a pair.
- Get the code. (“Download Zip”)
- Run tests, see many green tests.
- Start with TennisGame #2.
- Focus entirely on clean-up.
- Identify code smells.
- Try to use only automatic Refactorings.

Don't Focus on
Getting it Done.
Focus on Doing
It Perfectly.

→ Practice

Closing Circle

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





Peter Kofler

@codecopkofler

www.code-cop.org

Kata by

Emily Bache

@emilybache

<http://coding-is-like-cooking.info/2011/08/refactoring-kata-fun/>

CC Images

- Bruce <http://www.flickr.com/photos/sherpas428/4350620602/>
- pairing <http://www.flickr.com/photos/dav/94735395/>
- Dojo <http://www.flickr.com/photos/49715404@N00/3267627038/>
- agenda <http://www.flickr.com/photos/24293932@N00/2752221871/>
- smells <http://www.flickr.com/photos/hhbw/4215183405/>
- mercy <http://www.flickr.com/photos/williac/99551756/>
- Tennis <http://www.flickr.com/photos/gagillphoto/3706167856/>
- tools <https://www.flickr.com/photos/tom-margie/5019211728/>
- trap <https://www.flickr.com/photos/stevendepolo/13714018553/>
- wants you <http://www.flickr.com/photos/shutter/105497713/>