Tampere goes agile

Coderetreat

Mikko Pyhäjärvi, 14.12.2013

F-Secure

What you will need today

- A computer
- Programming environment
- A constructive attitude
- The wireless network is TampereHall, password customer2012

Why coderetreats

- Learning through pairing
- Deliberate practice
- Emphasis on experimentation
- Software as a craftsmanship

What is a coderetreat

- 1 day of coding
- Conway's game of life – simple rules, domain is not important
- An intense day of coding – five separate sessions, different "constraints/ rules" in each session
- Many different people to pair with
- Practicing the fundamentals of programming – object-orientation, functional programming, TDD, refactoring, cleaner code, maybe some more advanced concepts like memory management and distributed computation – in any language you and the person you pair with feels comfortable with.
- Focus should be on readability, extendibility and maintainability of the code
- A lot of the time you will not finish your implementation before the deadline.
- Deleting your code after each session
- Have fun doing stuff with fellow software engineers

A session

- One hour
  - 5 minutes introduction
  - 45 minutes of programming
  - 10 minute retrospective / break
  - Pair-programming is necessary, as that is the main method for learning today
  - After each session pairings will be changed
  - After each session code must be deleted. Not put into a branch, not stashed, just deleted with no trace of it left.

Structure of the day

- 09:45-10:45 Session I
- 10:45-11:45 Session II
- 11:45-12:45 Lunch
- 12:45-13:45 Session III
- 13:45-14:15 Coffee break
- 14:15-14:15 Session IV
- 15:15-16:15 Session V

Four rules for simple design

- Passes its tests
- Minimizes duplication
- Maximizes clarity
- Has fewer elements

Conway's game of life

- There's an infinite 2d orthogonal universe
- Initial generation called a seed
- The following rules are applied simultaneously
  - A live cell with less than 2 live neighbors dies
  - A live cell having 2 or 3 live neighbors lives
  - A live cell having more than 3 neighbors dies
  - A dead cell having 3 neighbors becomes alive

Session

- Find a pair
- Choose programming language
- Setup a new project in your environment
- Make a decision with your partner how to start
- In this session one person will be the one at the keyboard for the whole session, but both should actively participate in building the solution
- We'll start in five minutes

Session

- Pair programming
  - Test-driven-development
  - Ping-pong

Lunch

We'll continue at 12.45

Session – sparse data

- Your dataset is vast, but consists mostly of empty space
- Test-driven development
- Evil ping-pong
- No mouse, you can use mouse only to lookup keyboard shortcuts

Session – limited memory

- The amount of your processing memory is very limited. You won't be able to store your whole seed or generation in memory, but you have unlimited disk space.
- Don't focus on minimizing your implementation's memory footprint, but instead on the idea that all your data will not fit into memory regardless of how you encode it.
- Remember code readability

Session – refactoring frenzy

- Refactoring frenzy
- Precise names split (no and/or)
- No conditionals (such as if or switch)
- Maximum of five lines of code per method
- One assert per test
- One behavior per test

Refactoring frenzy

- Lack of tests
- Name not from domain
- Name not expressing intent
- Unnecessary if
- Unnecessary else
- Duplication of constant
- Magic numbers
- Method does more than one thing
- Primitive obsession
- Feature envy
- Method too long
- Too many parameters
- Tests : not unitary
- Tests : setup too complex
- Tests : test tests more than one thing
- Tests : no asserts
- Tests : too many paths

Session - Baby steps

- Initialize source control repository
- Start a timer for 2 minutes
- Write exactly one test
  - Timer rings, the test is red, then revert and start again at 2
  - The test is green before the timer rings, then commit
- Restart timer
- Refactor
  - Timer rings, the refactoring is not complete, revert and refactor again
  - The refactoring is complete before the timer rings, commit and start again

Timers should run continuously

Session – OO / FP distributed game of life

- The dataset is either too large, or the calculation of a cell's value is so expensive that finishing an iteration would take a very long time
- Build a version of the game of life where the computation is parallelizable / distributable
- Only immutable objects or Functional Programming

Closing circle

- What, if anything, did you learn today?
- What, if anything, surprised you?
- What, if anything, will you take to work with you ‘tomorrow’

Protecting the irreplaceable

F-Secure