Coderetreat Facilitator Notes

Introduction
Writers and musicians are always practicing. As software developers we don't very often practice. Coderetreat is our chance to practice outside of work constraints. We need to practice because at work we are often on deadlines so we must cut corners. By forcing people to delete their code it allows them to write code and not worry about finishing. This allows them to experiment and practice in a way they can't at work.

- Start with asking questions
    - who have been to a code retreat before?
    - who have heard of tdd?
    - who have tried tdd?
    - who does tdd daily?
    - who have written a book about tdd? :)
- Do an participants introduction round (very brief. e.g. name, company and hashtag) OR do the introduction round with languages later
- Have sticky name tags ready
- Main introduction could be slides, just talking, using websites, images, …
- Main introduction should address:
    - Introduce Coderetreat (just like it is introduced above)
    - Mention that the goal of the day is to practice our craft
    - Discuss the idea of "Reducing the Cost of Change" in our code
    - Introduce the coderetreat format
        - Make sure participants know they shouldn't try to finish the problem and why that is important
        - Discuss that you will stop participants and they need to just stop
        - Let participants know that they will be **deleting their code** and that this is required

- Introduce Conway's Game of Life
    - You can use the poster https://github.com/marcoemrich/game-of-life-rules
    - You can show a video https://www.youtube.com/watch?v=xP5-iIeKXE8
    - Encourage everyone to have the courage to experiment and to try new approaches to solving the problem or techniques for writing code
- Ask everyone to introduce themselves, what language they prefer and are setup to use, and what language they would like to get experience with at CR
    - One group decided to write the names and languages on a whiteboard so they

    could know who to pair with
  - My style is to do the introduction earlier. And just ask for languages with hand voting at this point.
- Optional things you can discuss:
  - The TDD Cycle Red/Green/Refactor and writing the least amount of code to get it to pass
  - Characteristics and qualities of “Good” Code (What does good code look like?)
  - Four rules of simple design: (http://c2.com/cgi/wiki?XpSimplicityRules)
    - Passes all Tests
    - Expresses Intent (Clear, Expressive, & Consistent)
    - No Duplication
    - Minimal methods, classes, & modules (no superfluous abstractions)
- A good overview of the basic Coderetreat format and structure is available on the coderetreat.org website:
  - http://coderetreat.org/facilitating/structure-of-a-coderetreat
- There is a github repo participants can clone that contains a set of “starting points” for various languages:
  - https://github.com/coreyhaines/coderetreat
  - The starting points are in the starting_points subdirectory
- Feel free to use an online coding environment like http://www.cyber-dojo.org or http://tddbin.com (JavaScript only)
- You can use your own style of facilitation, leave the participants alone, ask a lot of questions or try to teach them. If in doubt do less of facilitation.

Sessions

The first session should be very simple. This is an opportunity for people to make sure they have their environments setup for TDD and to get familiar with Conway’s Game of Life. The first session. After the first session it is usually difficult for people to delete their code.

- Let participants know that at the end of the session you will ask everyone to delete their code and stand up. This adds some peer pressure and helps to get everyone to actually delete their code
- Sessions last 45 minutes followed by a 15 minute retrospective and break
  - Don’t announce the time left and don’t let them worry about the time (if they ask be non-specific)
- During the sessions the facilitator should observe each pair and offer any necessary assistance
- After the first session it might be good to ask about deleting their code: Who found it difficult to delete their code? Why was it difficult?
- It’s ok if they spend a significant portion of session 1 getting the tools setup.
  - Help them if you know the environment
  - Grab someone else in the room to help them
  - The could write their own testing framework

        - Only requires an AssertTrue method
        - Throws exceptions if fails
        - Or prints failed or passed to the console

Session Activities (see http://coderetreat.org/facilitating/activity-catalog)

<u>Testing based Activities</u>

- Ping Pong - One person writes the test, the other implements the test
- Mute Ping Pong - Same as Ping Pong, but the pair is not allowed to talk. No cheating with comments or any form of writing. You can talk about things not related to the problem, but you cannot talk about design. (Facilitator note: this should be teaching about expressive design)
- Evil coder - an activity where the person implementing tries to implement the code in a way the tester doesn't expect (also teaches expressive design)
- You could combine Mute ping pong and Evil coder
- Taking Baby steps - Every pair has their own timer. They set it for 5 minutes. They have 5 minutes to write the code. Then 5 minutes to implement the test. Then 5 minutes to refactor. If they don't finish before the timer goes off they have to delete and start over. (People get frustrated for the first 10 minutes.)

<u>Other Activities</u>

- Remove language concepts
- No conditional statements (no ifs, no switch, no loops for conditional)
    - The alternatives are polymorphism and hashtables
    - Conditional statements are a form of primitive obsession (we tend to use the lowest level of abstraction instead of choosing a higher level). For example, we may choose an int for an employee number instead of an employee number class.
- Every method must be void (very challenging, encourages "tell don't ask" style)
    - In game of life, you might have a GetNextGenaration method. Instead you might write CreateNextGeneration
- No loops
- No naked primitives, they must be enclosed in a class
- Only 4 lines per method
- Only use immutable objects
- Not allowed to touch the mouse (you can use the mouse to discover the keystrokes)
- No editor (not the best idea) teaches people to know their languages well enough they don't need
- After the lunch have a mute session
- A good activity for the last two activities of the day are:
    - After the second to last session have them stand up but don't delete their code, then sit down and write what you do and don't like about your code
    - And then for the last session they aren't going to swap pairs, instead they are

going to swap stations and work on another pair's code. You need to facilitate this in a very active way. Ask each pair what they've decided to improve on. Encourage them to go back to the four rules. This activity does not work if every pair is in a different programming language.

Handling Difficult Participants

- Watch out for people who are very dominant and ask them to pull back a little bit. (Ping pong helps with that)
- If someone does not delete the code or do as told, ask a second time and then ignore this person. We are not the police ;-)
- It is more important to create a healthy learning environment than to follow all the rules to the letter.

Other Notes

- It is very common to implement Game of life as a two dimensional array. You might want to jar them out of that. A good activity for this is "verbs" instead of nouns. This is usually good for session two. Every class name and variable name needs to be a verb.
    - CreatesCellGeneration
    - AppliesRuleNumberOne
    - People tend to want to focus on how to represent the system. But have them start with the rules and how to implement them.

- Take notes throughout the day:
    - Something that was interesting
    - Something that worked well
    - Something that didn't work so well (mistakes)

- When you don't know the answer to a question, throw it back to the group
    - "That's a great question. Does anyone want to try to answer it?"
- Be very strict about deleting code. Have people stand-up at the end of an iteration and stretch after they deleted their code
- Make sure you are the most prepared person in the room
- We want to try to turn people's heads upside down to encourage creativity.

- If people ask if they are ready for a Code Retreat, the answer is always yes.
- Maybe as facilitator take care to watch out for signs of demotivated people.

Tips for Facilitating the Iteration Retrospectives

- Remember: you are facilitating a discussion, not lecturing a group
- Ask questions rather than telling. Let the group discover things on their own.
- Try to get everyone involved. Ask people who haven't contributed much what they think (or some other specific question).

Tips for Facilitating the Closing Circle

- Gather everyone in a circle after the final iteration retrospective
- Prior to the closing circle, tell them the following three questions and then give them a break to think about these three questions. Then in the closing circle ask each participant to answer each of the questions:
    - What if anything, surprised you today?
    - What, if anything, did you learn today?
    - What, if anything, will you do differently on Monday or in the future?

- Enforce the "keep it short" rule as best you can
- Don't take notes if that will make people uncomfortable
- It's okay to ask for facilitation feedback afterwards, but don't ask for it during the circle
- Go first (to show people how it's done)

Common questions participants ask and how to respond

- Are we really going to work on the same problem all day long?
    - Yes. Conway's Game of life works very well for that sort of thing.
    - It doesn't get old. There is always something new to learn

- How much time is left before the iteration is over?
    - A couple of minutes (never be specific, never tell them exactly how much time is left)

Common first-time facilitator questions

- What do I do if I have a participant who is disruptive in some way (rude, inexperienced, etc)?
    - Don't let it bother you too much. The format of the event prevents a single person from sabotaging the entire day (except for really small groups).
    - If someone complains, remind them that they don't have the pair with that person again. So it was only one iteration that was "bad".

- What do I do if we have an odd number of people?
    - You can allow three people to program together rather than two. Having a group of three works fine.
    - Don’t let anyone work alone. Everyone should be working with someone else.
    - Make sure one person doesn’t always end up being the “odd man out”

by