SE303

Class 1 Notes:

Agenda

Intros

Course Overview

Exercise

ToDos:

-Syllabus & Pro Dev Quizes

-Exploration 1.1 , 1.2

Refactoring Overview

* Have an open mind
* About Techniques

SE303

Class 2 Notes:

Agenda

* Administrativia and Check – In
* 99 Bottles of reflection
* Shameles Green (Exploration 1.1 highlights)

ToDos:

-Syllabus & Pro Dev Quizes

-Exploration 1.1 , 1.2

Refactoring Overview

* Have an open mind
* About Techniques

Class Notes

* Cyclomatic Complexity, line of code
* <https://gist.github.com/ybakos/a1595640af9860d0f5dcb0b1a96ee03e> (statement function)
* Establishing a discipline is something that we can rely on consistently
* Step 1
  + You’re going to stay horizontal and you’re extracting the method and you should notice what variable that depends on that’ll be out of scope. In addition, you should notice what variables are local to this work.
* Step 2
  + Is there anything that we can do clarify this function?
    - thisAmount gets changed
* Step 3
  + Benefit of having a function in a function (Javascript)
    - As we introduce new abstractions he keeps them as hidden as possible, so by keeping the amount for function inside statement, we are not creating a new function that someone else could use in a different context, that may or may not be appropriate right.
    - So, by declaring it inside amount for no one else can to call it, except for the code amountFor
* Step 4
  + Inline Variable
    - Assign where temp variable normally is with the functions value so you would do playFor(aPerformance).
    - You then can delete the temp variable
* Step 6
  + When renaming a method that has a bunch of calls already made. You can duplicate the method and add the new name, and that way you can test to see if it compiles before implementing the change.
* Step 7
  + [Technique] Split Loop (for volumeCredit and totalAmount variable)
    - Split the loop into own function
    - Slide the temp variable down
    - Replace temp variable with function call (inline varia)… \*Recall inline variable technique is where you replace the temp variable with the result of a function.

Week 1 Reading – 99 Bottles CHP 2

* How to gauge expense (in context of code)
  + How difficult was it to write?
  + How hard is it to understand?
  + How expensive will it be to change?
* Different Version of 99 Bottles Implementation
  + Incomprehensively concise - Overly abstract
    - consistency, duplication of code, naming. Is more costly
  + Speculatively General - Overly verbose
    - harder to understand without being easier to change
  + Concretley Abstract
    - Expensive to change due to method names being at the wrong level of abstraction
    - Name methods on meaning (what they represent in the context of your code) rather than on what they do
  + Shameless Green
    - Brute force
    - Prioritizes Readability of Chageability
  + Good Code
    - works, simple, understandable , expressive and changeable
  + Evaluating Code Based on facts
    - Source Line of Code
      * N lines of code equated to how well productivity was
      * Predict cost of maintenance
      * Predict effort
    - Cyclomatic Complexity
      * A mathematical technique to identify code that is difficult to test or maintain.
      * Algorithm that counts the number of unique execution paths through a body of source code.
      * Method with deeply nested code would score high. A method with no conditionals equates to 0.
      * Tells you the minimum number of tests needed to cover all of the logic in the code.
    - (Assignments, Branches and Conditions) ABC
      * Assignments is a count of variable assignments
      * Branches count not branches of an if statement but branches of control, meaning function calls or message sends
      * Conditions count conditional logic
      * High scores indicate expensive maintainability and hard testing

Week 1 Reading – Refactoring Chp 1

* Split Phase
  + Divide logic up into two parts
  + Ex. one that calculates the data required for the statement, the other that renders it into text or HTML.
  + Treat data as immutable
* Move Function