Activity 7-1: The Open-Closed Principle

Interjecting segments of the OCP video[1] (only available in class or by purchase).

Overview

01:00-04:10 overview

Open and Closed

11:40-18:20 open and closed

A module or component conforms to the **Open-Closed Principle** if its both *open to extension* and *closed to modification*. These two things seem contradictory, until you dig deeper into what these phrases mean.

1. What does it mean for a module to be *open to extension*?
2. What does it mean for a module to be *closed to modification*?
3. What does it mean to *separate out extensible behavior using an abstraction*?
4. What is a *software dependency*, and what does it mean to *invert* that dependency*?
5. How does the combination of *abstraction* and *inversion* allow for the behavior of a module to be extended without modifying its source code?
6. What is the main implication of conforming to the open-closed principle?

Feasibility of Open-Closed Principle

**18:20-20:40 feasibility of open-closed principle 1. Is is *possible* to always write your code so that it conforms to the open-closed principle? 2. Is it *practical* to always write your code to conform to the open-closed principle? 3. What is the "crystal ball" problem? 4. Is a single class easier to make conform to the open-closed principle than a large system component composed of many different classes? Why or why not?

Example: Accounting System

20:40-42:45 accounting system example

1. Stop at 24:40
    - Go over code handout and what is being accomplished by each part of the printReport and the Expense class.

[1] ../videos/11-ocp.html

2. Skip forward to 27:30
3. Pause at “business rules”:
    - What does he mean by *business rules*?
    - Where are the business rules in the code?
4. Pause after “new meal type: lunch or snack”:
    - What is the first line of code that we would have to change?
    - How places just in printReport would we need to change?
5. Pause at 34:30
    - We could extract out methods for each bit of functionality, but it won’t help. Why?
6. Stop at 35:23
    - Rewind to show printReport diagram.
    - Go over handout with new system design.
        a. What does the “” on the ExpenseName box mean?
        b. What does the “” on the Expense box mean?
        c. What is the difference between the dotted line and the solid line?
        d. Where are the abstractions in the diagram?
7. Skip forward to 38:40
8. Pause at 40:55
    - Where does he mean by *derivitive*?

The Lie

42:45-46:55 the lie

Two Solutions

46:55-52:12 two solutions

- Pause at 48:20
    - What is the problem with “big design upfront”?

Agile Design in Practice

52:10-56:15 agile design in practice

- Pause at 52:12
  - What does a development team do during “iteration 0”?
  - What is the goal of iteration 0?

Reprise

56:15-57:25 reprise

Summary and Closing Credits

57:25-1:02:17 summary/credits SKIP