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

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

Feasibility of Open-Closed Principle

**18:20- 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 conformm to the open-closed principle?

Example: Point of Sale System

15:05-20:00 point of sale system

1. What is the secondary value of software? When is it achieved?
2. What is the primary value of software? When is it achieved?
3. What is the primary responsibility of programmers?

20:00-27:50 Friction

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

1. What discipline would make software easier to maintain and enhance?
2. What is the problem with a module that has too many responsibilities?
3. What is the *fan out* of a class? What is the *fan in*?
4. Why is it important to reduce the fan-out of a class?
5. What is one way to achieve reduce the fan-out of a class?
6. What is **collocation of responsibilities**? How does it affect the various actors?
7. What code smell is the likely result of this collocation of responsibilities?

The Single Responsibility Principle

27:50-30:21 single responsibility principle (SRP)

1. What does the SRP say? What are some examples of this?

- **30:21-32:10 Example 1** Discussion
- **32:10-33:27 Example 2** Discussion
- **33:27-35:52 Example 3** Discussion
- **35:52-41:15 Example 4** Discussion

2. (Group discussion) Does our grading application have any violations of the SRP?
3. What is the overall solution to resolving SRP violations?

Break?

Resolving SRP violations

41:15-48:05 solutions

1. In problems related to SRP, there are a number of competing interests:
    - Separating responsibilities into different classes
    - Separating actors from the concrete implementations of their responsibilities
    - Avoiding transitive dependencies between actors
    - Making functions easy to find

    How do the following techniques balance these interests out: **dependency inversion**, **extracting classes**, the **facade** pattern, **interface segregation**

Mastermind: A case study

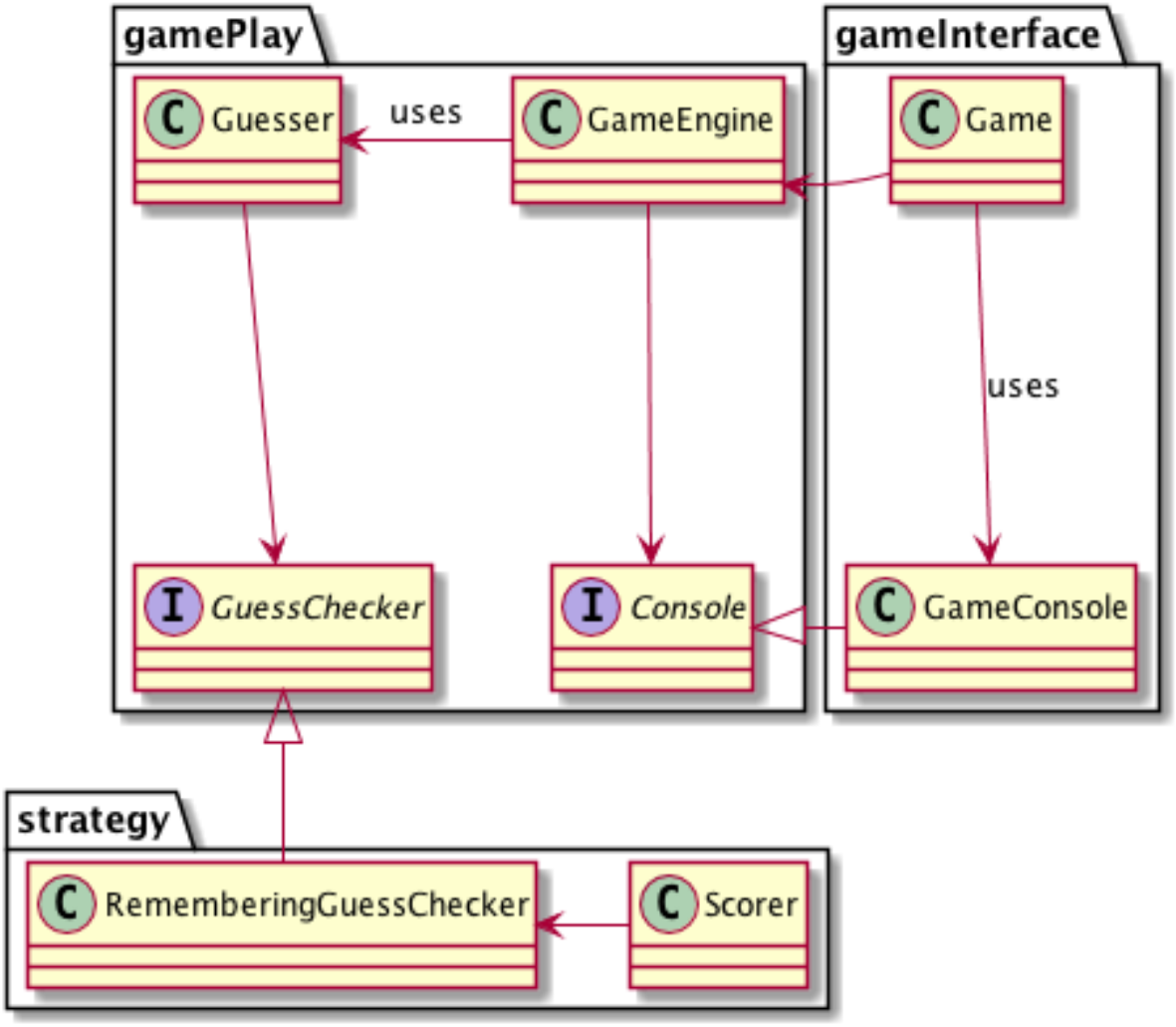
48:05-51:50 The game of mastermind

1. What are the actors and responsibilities in this game?

51:50-53:45 Discussing the three actors

1. List the actors presented in the video and describe the corresponding responsibilities.

53:45-57:00 Details on Mastermind game implementation



57:00-1:00:15 Faking a rational design process

1. How did Uncle Bob come about his design for the Mastermind Game?

2. Why do unit tests tend to align with actors? What is the advantage of this?
3. When should we draw design diagrams?

- **1:00:15-1:02:50 summary**