SWE 4743:
Object-Oriented Design

Jeff Adkisson

Open-Closed Principle (OCP) and the Decorator Pattern

Open–Closed Principle (OCP)

- Software entities should be open for extension, closed for modification
  - Goal: manage change safely, not avoid edits forever
  - Built on Single Responsibility Principle (SRP)

OCP Depends on SRP

- A class with multiple reasons to change cannot be closed
  - SRP defines stable responsibilities
  - OCP governs how those responsibilities evolve

OCP as Risk Containment

- Protect stable, trusted code
  - Localize new behavior
  - Reduce regression risk

SRP Violation Example

- InvoiceService handles pricing, persistence, and email
  - Multiple actors create competing change pressure
  - Must decompose before OCP can apply

Progressive Evolution Toward OCP

- Start simple when change is unlikely
  - Refactor when change pressure appears
  - Do not over-design on day one

Signals for Strategy Pattern

- if/else logic based on policy
  - New rules repeatedly modify same method
  - Algorithms vary independently

Disciplined Abstraction

- Abstraction should respond to variation
  - Avoid predicting the future
  - Over-abstraction increases complexity

Rules of Thumb

- No interface without variation
  - Abstract after evidence, not before
  - Watch conditionals that grow with business rules

Why Decorator?

- Add behavior without modifying core logic
  - Avoid combinatorial subclass explosion
  - Supports OCP and SRP simultaneously

Decorator Pattern Structure

- Same interface as component
  - Wraps another component
  - Adds behavior before or after delegation

Decorator Use Cases

- Logging
  - Metrics
  - Retries
  - Cross-cutting concerns

Benefits of Decorator

- Extend behavior safely
  - Compose behavior at runtime
  - Keep core classes simple

Key Takeaways

- OCP is earned through refactoring
  - SRP enables OCP
  - Strategy and Decorator isolate change
  - Good design minimizes future pain