2. Structural Patterns – Nhóm cấu trúc

Tập trung vào cách tổ chức các lớp và đối tượng để tạo ra cấu trúc lớn hơn.

Tên Pattern	Mô tả ngắn
Adapter	Biến đổi interface của class này thành interface mà client mong muốn
Bridge	Tách abstraction khỏi implementation
Composite	Cấu trúc cây để biểu diễn các phần - tổng thể
Decorator	Thêm chức năng cho đối tượng mà không thay đổi code gốc
Facade	Cung cấp interface đơn giản để truy cập hệ thống phức tạp
Flyweight	Tối ưu tài nguyên khi có nhiều đối tượng giống nhau
Proxy	Đại diện cho một đối tượng khác để kiểm soát truy cập

Bài tập luyện tập

✅ 1. Adapter Pattern

Mục tiêu: Biến đổi interface của một class thành interface mà client mong muốn.

🔸 Bài tập 1 – Vừa

Đề:
Giả sử bạn có một hệ thống phát nhạc cũ (`OldMediaPlayer`) chỉ hỗ trợ file `.mp3`. Bạn cần tích hợp vào một hệ thống mới (`NewAudioPlayer`) hỗ trợ cả `.mp4`, `.vlc` bằng cách sử dụng Adapter.

Yêu cầu:

- Viết interface `MediaPlayer` (hệ thống mới)
- Tạo lớp Adapter chuyển `AdvancedMediaPlayer` (mp4, vlc) về `MediaPlayer`
- Client chỉ dùng `MediaPlayer`

🔸 Bài tập 2 – Nâng cao

Đề:
Bạn có một hệ thống quản lý thanh toán `Paypal`, `Stripe`, và `Crypto`. Mỗi hệ thống có API khác nhau. Hãy sử dụng Adapter để xây dựng một API thanh toán chung cho client (PaymentClient).

✅ 2. Bridge Pattern

Mục tiêu: Tách abstraction khỏi implementation để hai phần có thể phát triển độc lập.

🔸 Bài tập 1 – Vừa

Đề:
Thiết kế hệ thống điều khiển từ xa cho nhiều loại TV (Samsung, Sony). Dùng Bridge để điều khiển từ xa (RemoteControl) hoạt động với nhiều loại TV mà không bị phụ thuộc chặt.

🔸 Bài tập 2 – Nâng cao

Đề:
Tạo một hệ thống hiển thị tài liệu hỗ trợ nhiều định dạng (`PDF`, `Word`, `Markdown`) và nhiều loại thiết bị (`Mobile`, `Desktop`). Dùng Bridge để tách rendering logic khỏi loại thiết bị và loại tài liệu.

✅ 3. Composite Pattern

Mục tiêu: Dùng cấu trúc cây để biểu diễn mối quan hệ phần – tổng thể.

🔸 Bài tập 1 – Vừa

Đề:
Xây dựng hệ thống quản lý nhân sự: nhân viên có thể là cá nhân (leaf) hoặc quản lý (composite). Viết chương trình in toàn bộ sơ đồ công ty.

🔸 Bài tập 2 – Nâng cao

Đề:
Xây dựng hệ thống menu ứng dụng, gồm: `MenuItem` và `SubMenu`. Hỗ trợ hiển thị menu dạng cây với nhiều cấp độ lồng nhau.

✅ 4. Decorator Pattern

Mục tiêu: Thêm hành vi cho đối tượng mà không sửa đổi lớp gốc.

🔸 Bài tập 1 – Vừa

Đề:
Tạo hệ thống in cà phê. Lớp `Coffee` cơ bản có thể được trang trí thêm đường, sữa, kem... Dùng Decorator để tính giá và mô tả cuối cùng.

🔸 Bài tập 2 – Nâng cao

Đề:
Tạo hệ thống gửi email. Email có thể được mã hóa, nén, hoặc thêm chữ ký điện tử. Dùng Decorator để kết hợp các tính năng này mà không thay đổi lớp `EmailSender` gốc.

✅ 5. Facade Pattern

Mục tiêu: Cung cấp một interface đơn giản để truy cập vào hệ thống phức tạp.

🔸 Bài tập 1 – Vừa

Đề:
Thiết kế hệ thống xem phim tại nhà gồm nhiều thành phần: `DVDPlayer`, `Projector`, `SoundSystem`. Viết lớp `HomeTheaterFacade` để đơn giản hóa thao tác xem phim.

🔸 Bài tập 2 – Nâng cao

Đề:
Viết hệ thống thanh toán thương mại điện tử. Gồm các subsystem: `Inventory`, `PaymentGateway`, `Shipping`, `Notification`. Tạo Facade `OrderService` cho client sử dụng đơn giản.

✅ 6. Flyweight Pattern

Mục tiêu: Tối ưu tài nguyên khi có nhiều đối tượng giống nhau.

🔸 Bài tập 1 – Vừa

Đề:
Tạo hệ thống game nơi có rất nhiều lính (`Soldier`). Mỗi lính có đặc điểm giống nhau (vũ khí, trang phục) trừ vị trí. Dùng Flyweight để chia sẻ dữ liệu dùng chung.

🔸 Bài tập 2 – Nâng cao

Đề:
Thiết kế hệ thống hiển thị ký tự văn bản (`Character`) cho trình soạn thảo. Mỗi ký tự chia sẻ kiểu font, màu sắc, kích thước, chỉ khác vị trí và ký tự cụ thể. Áp dụng Flyweight để giảm bộ nhớ.

✅ 7. Proxy Pattern

Mục tiêu: Đại diện cho một đối tượng khác để kiểm soát truy cập, lazy loading, hoặc logging.

🔸 Bài tập 1 – Vừa

Đề:
Tạo hệ thống xem hình ảnh: `Image` là interface, `RealImage` là lớp load ảnh thật. Tạo `ProxyImage` để trì hoãn việc load ảnh cho đến khi hiển thị thật sự.

🔸 Bài tập 2 – Nâng cao

Đề:
Xây dựng hệ thống truy cập tài liệu an toàn. Mỗi file có thể bị hạn chế theo role (admin, guest). Dùng Proxy để kiểm soát việc đọc file dựa trên quyền người dùng.
