REVIEW SÁCH CHƯƠNG 7:
TỪ MODULE ĐẾN ĐỐI TƯỢNG

I. Tóm tắt nội dung:

Chương này tập trung vào sự phát triển từ mô hình **lập trình hướng cấu trúc (modules)** sang**lập trình hướng đối tượng (objects)**. Nó giải thích tại sao lập trình hướng đối tượng (OOP) mang lại nhiều lợi ích hơn trong thiết kế phần mềm, đặc biệt là về tính đóng gói, tính kế thừa, và tính đa hình.

1. Khái niệm về Module:

· Khi một chương trình quá lớn và chỉ có một khối mã duy nhất, nó rất khó bảo trì. Dođó, cần chia nhỏ thành các **module**.
· Một module là một tập hợp các câu lệnh chương trình có thể được gọi bởi các phầnkhác trong hệ thống.
· Các module giúp cải thiện tính **dễ đọc**, **tái sử dụng** và **bảo trì** của mã nguồn. **2. Mô hình module và**

các vấn đề gặp phải:

· Nếu thiết kế module không tốt, việc bảo trì sẽ khó khăn, giống như thiết kế phần cứngtách rời không hợp lý.
· Chia module theo cách sai có thể làm tăng độ phức tạp và giảm tính tái sử dụng. **3. Độ kết nối (Cohesion) của**

module:

Mức độ kết nối trong một module có thể được phân loại từ **tệ nhất đến tốt nhất**:

- **Kết nối tình cờ (Coincidental Cohesion)** – Module làm nhiều việc không liên quan đến nhau.
- **Kết nối logic (Logical Cohesion)** – Module thực hiện nhiều tác vụ nhưng không có quan hệ rõ ràng.
- **Kết nối thời gian (Temporal Cohesion)** – Các tác vụ trong module xảy ra cùng một thời điểm.
- **Kết nối thủ tục (Procedural Cohesion)** – Module thực hiện một chuỗi thao tác theo thứ tự nhất định.
- **Kết nối giao tiếp (Communication Cohesion)** – Các thao tác trong module đều xử lý cùng một dữ liệu.
- **Kết nối chức năng (Functional Cohesion)** – Module thực hiện đúng một nhiệm vụ duy nhất.
- **Kết nối thông tin (Informational Cohesion)** – Module hoạt động trên một cấu trúc dữ liệu duy nhất.

4. Độ liên kết (Coupling) giữa các module:

Mức độ liên kết giữa các module có thể ảnh hưởng đến khả năng bảo trì phần mềm: - **Liên kết nội dung (Content Coupling)** – Một module thay đổi trực tiếp nội dung của module khác (**xấu nhất**).
- **Liên kết chung (Common Coupling)** – Các module cùng truy cập một biến toàn cục (**gây lỗi bảo mật**).
- **Liên kết điều khiển (Control Coupling)** – Một module truyền biến cờ (flag) để module khác quyết định hành vi.
- **Liên kết tem (Stamp Coupling)** – Truyền toàn bộ cấu trúc dữ liệu thay vì chỉ các thành phần cần thiết.
- **Liên kết dữ liệu (Data Coupling)** – Chỉ truyền những dữ liệu cần thiết giữa các module(**tốt nhất**).

5. Tại sao lập trình hướng đối tượng tốt hơn?:

· **Đóng gói (Encapsulation)**: Giúp ẩn thông tin và chỉ cho phép truy cập qua giao diện. · **Kế thừa (Inheritance)**: Tái sử dụng mã nguồn mà không cần viết lại từ đầu. · **Đa hình (Polymorphism)**: Cho phép cùng một hành động có thể thực thi theo nhiều cách khác nhau.

6. Lập trình hướng đối tượng và thiết kế phần mềm:

· Các lớp trong OOP thực chất là các **module có tính đóng gói cao**. · Khi sử dụng đúng cách, OOP giúp phần mềm dễ bảo trì, dễ mở rộng và ít lỗi hơn.

II. <u>Bài học rút ra:</u>

- **Thiết kế module tốt** giúp phần mềm dễ bảo trì và mở rộng hơn.
- **Giảm liên kết (coupling) và tăng kết nối (cohesion)** giữa các module giúp phần mềm linh hoạt hơn.
- **Lập trình hướng đối tượng (OOP) là sự phát triển tự nhiên** của mô hình module, giúp phần mềm tái sử dụng và dễ mở rộng.
- **Sử dụng các nguyên tắc như đóng gói, kế thừa, và đa hình** để tăng cường tính hiệu quả của mã nguồn.

=> **Tóm lại**, chương này nhấn mạnh rằng lập trình hướng đối tượng không phải là một cuộc cách mạng, mà là sự tiến hóa tự nhiên từ mô hình module, giúp phần mềm dễ quản lý và bảo trì hơn!