Bài tập chương 7

I. Câu hỏi trắc nghiệm:

Câu hỏi 1: Thuộc tính của một lớp nên được thiết kế như thế nào để đảm bảo tính đóng gói dữ liệu? A. Public

B. Private

C. Protected

D. Internal

Câu hỏi 2: Nguyên lý nào đảm bảo rằng dữ liệu chỉ có thể được truy cập thông qua các phương thức công khai của lớp?

A. Hướng trách nhiệm

B. Đóng gói dữ liệu

C. Thực hiện lời gọi nhiều lần

D. Kế thừa

Câu hỏi 3: Thẻ CRC bao gồm những thành phần nào?

A. Class, Relation, Composition

B. Class, Responsibility, Collaboration

C. Class, Reference, Control

D. Class, Role, Connection

Câu hỏi 4: Quan hệ nào giữa các lớp thể hiện việc một lớp là thành phần của lớp khác và không thể tồn tại độc lập?

A. Association

B. Aggregation

C. Composition

D. Inheritance

Câu hỏi 5: Sơ đồ FSM mô tả điều gì?

A. Cấu trúc dữ liệu của hệ thống

B. Các trạng thái và chuyển đổi giữa các trạng thái của đối tượng

C. Các lớp và quan hệ giữa các lớp

D. Cách người dùng tương tác với hệ thống

Câu hỏi 6: Nguyên lý nào khuyến khích việc thiết kế phương thức để tái sử dụng nhiều lần trong các ngữ cảnh khác nhau?

A. Hướng trách nhiệm

B. Thực hiện lời gọi nhiều lần

C. Đóng gói dữ liệu D. Phân lớp

Câu hỏi 7: Chuẩn hóa quan hệ giữa các bảng về dạng 3-NF nhằm mục đích gì?

A. Tăng tốc độ xử lý dữ liệu

B. Giảm thiểu dư thừa dữ liệu và đảm bảo tính nhất quán

C. Đảm bảo các bảng chứa đầy đủ dữ liệu

D. Tăng khả năng mở rộng của hệ thống

Câu hỏi 8: Lớp nào trong hệ thống thường chứa các thuộc tính lưu trữ dữ liệu và không chứa nhiều logic xử lý?

A. Lớp điều khiển

B. Lớp biên

C. Lớp thực thể

D. Lớp giao diện

Câu hỏi 9: Khi xây dựng sơ đồ lớp chi tiết, điều gì cần được bổ sung?

A. Thêm các quan hệ giữa các lớp p

B. Thêm các thuộc tính và phương thức đầy đủ cho từng lớ

C. Thêm giao diện người dùng

D. Thêm các sơ đồ tuần tự

Câu hỏi 10: Trong mô hình cơ sở dữ liệu quan hệ, một thuộc tính của bảng được gọi là:

A. Field

B. Row

C. Record D. Table

II. Câu hỏi ngắn:

1. Tại sao cần đóng gói dữ liệu khi thiết kế lớp?

- Đóng gói dữ liệu (Encapsulation): giúp bảo vệ dữ liệu bên trong lớp khỏi bị truy cập hoặc sửa đổi ngoài ý muốn từ bên ngoài. Nó giúp đảm bảo tính toàn vẹn của dữ liệu, tăng tính bảo mật và giúp mã dễ bảo trì

2. Nguyên lý hướng trách nhiệm trong thiết kế phương thức là gì?

- Nguyên lý hướng trách nhiệm (Single Responsibility Principle - SRP) quy định rằng mỗi phương thức hoặc lớp chỉ nên đảm nhận một nhiệm vụ duy nhất. Điều này giúp mã dễ hiểu, dễ bảo trì và tái sử dụng.

3. Thẻ CRC là gì?

- Thẻ CRC (Class-Responsibility-Collaboration) là một công cụ thiết kế phần mềm giúp xác định lớp, trách nhiệm của lớp, và cách lớp tương tác với các lớp khác. Nó hỗ trợ trong việc phân tích thiết kế hướng đối tượng.

4. Quan hệ Aggregation và Composition khác nhau như thế nào?

- **Aggregation**: Mô tả mối quan hệ "có nhưng không sở hữu", trong đó một lớp có thể tồn tại độc lập với lớp chứa nó.
- **Composition**: Mô tả mối quan hệ "có và sở hữu", trong đó nếu lớp chứa bị xóa, các đối tượng thành phần cũng bị xóa theo. **5. FSM (Finite State Machine) là gì?**
- FSM (Máy trạng thái hữu hạn) là một mô hình tính toán biểu diễn một hệ thống có số lượng hữu hạn các trạng thái. Nó mô tả các trạng thái của một đối tượng và cách đối tượng chuyển đổi giữa các trạng thái dựa trên các sự kiện.

6. Mục tiêu của việc chuẩn hóa cơ sở dữ liệu là gì?

- Chuẩn hóa cơ sở dữ liệu nhằm loại bỏ sự dư thừa dữ liệu, giảm khả năng xảy ra lỗi cập nhật và đảm bảo tính nhất quán trong dữ liệu. **7. Lớp điều khiển có vai trò gì trong hệ thống?**
- Lớp điều khiển (Controller Class) xử lý luồng dữ liệu giữa giao diện người dùng và logic nghiệp vụ. Nó nhận yêu cầu từ người dùng, điều phối xử lý và trả kết quả về giao diện.

8. **Nguyên lý thực hiện lời gọi nhiều lần giúp ích gì trong thiết kế phương thức?** - Nguyên lý này giúp thiết kế phương thức linh hoạt, có thể được sử dụng nhiều lần trong các bối cảnh khác nhau mà không cần thay đổi mã nguồn. Điều này giúp tối ưu hóa hiệu suất và tăng khả năng tái sử dụng.
9. **Sơ đồ lớp là gì?**

- Sơ đồ lớp (Class Diagram) là một loại sơ đồ UML biểu diễn các lớp trong hệ thống, thuộc tính, phương thức của chúng và các mối quan hệ giữa các lớp.

10.Dạng chuẩn 3-NF của cơ sở dữ liệu là gì?

- Dạng chuẩn 3 (Third Normal Form - 3NF) là cấp độ chuẩn hóa cơ sở dữ liệu trong đó mọi thuộc tính không khóa chỉ phụ thuộc vào khóa chính, giúp loại bỏ sự phụ thuộc bắc cầu và giảm dư thừa dữ liệu.

III. Câu hỏi thảo luận nhóm:

1. Thảo luận về vai trò của việc đóng gói dữ liệu khi thiết kế lớp.

- Đóng gói dữ liệu (Encapsulation) là một nguyên lý quan trọng trong lập trình hướng đối tượng, giúp bảo vệ dữ liệu bên trong lớp và chỉ cho phép truy cập thông qua các phương thức cụ thể.
- Vai trò:

+ Bảo mật dữ liệu: Hạn chế truy cập trực tiếp vào dữ liệu, giảm nguy cơ sửa đổi ngoài ý muốn.
+ Kiểm soát truy cập: Sử dụng các phương thức getter và setter giúp kiểm soát cách dữ liệu được truy cập và thay đổi.
+ Tăng tính bảo trì: Dễ dàng thay đổi cách dữ liệu được lưu trữ hoặc xử lý mà không ảnh hưởng đến phần còn lại của hệ thống.

+ Tăng tính đóng gói và trừu tượng: Giấu chi tiết triển khai, chỉ cung cấp những gì cần thiết cho bên ngoài.

2. So sánh giữa Aggregation và Composition trong thiết kế lớp.

Đặc điểm	Aggregation	Composition
Mối quan hệ	Có nhưng không sở hữu	Có và sở hữu
Tính độc lập	Đối tượng con có thể tồn tại mà không cần đối tượng cha	Đối tượng con phụ thuộc vào đối tượng cha
Tác động khi hủy	Khi đối tượng cha bị hủy, đối tượng convẫn tồn tại	Khi đối tượng cha bị hủy đối tượng con cũng bị hủy
Ví dụ	Lớp Company có Employee (Nhân viên vẫn tồn tại khi công ty giải thể)	Lớp Car có Engine (Nếu xe hủy, động có cũng bị hủy)

3. Thảo luận về cách xây dựng thẻ CRC hiệu quả cho hệ thống.

- Thẻ CRC (Class-Responsibility-Collaboration) giúp xác định trách nhiệm của mỗi lớp và cách chúng tương tác với nhau.
- Cách xây dựng:

+ Xác định các lớp chính: Dựa vào yêu cầu hệ thống, xác định các thực thể quan trọng + Ghi rõ trách nhiệm: Mỗi lớp chỉ nên có một vai trò chính (theo nguyên tắc SRP). + Xác định cộng tác viên (Collaboration): Các lớp có thể cần hợp tác với lớp khác để thực hiện nhiệm vụ.
+ Sử dụng thẻ vật lý hoặc phần mềm chuyên dụng: Giúp dễ dàng tổ chức và thay đổi thiết kế khi cần.

4. Tại sao cần xây dựng sơ đồ FSM cho các lớp trong hệ thống?

- FSM (Finite State Machine) giúp mô tả cách một đối tượng chuyển đổi giữa các trạng thái khác nhau dựa trên các sự kiện - Lợi ích:

+ Giúp mô hình hóa hành vi hệ thống: Dễ dàng xác định trạng thái và các quy tắc chuyển đổi.
+ Cải thiện tính logic: Tránh các tình huống lỗi khi xử lý trạng thái.
+ Tăng hiệu quả kiểm thử: Dễ phát hiện lỗi khi xác định các trạng thái hợp lệ và không hợp lệ.
+ Ví dụ: Một Order có thể có các trạng thái: Pending, Confirmed, Shipped, Delivered, Cancelled.

5. Thảo luận về các bước chuẩn hóa cơ sở dữ liệu và vai trò của từng bước.

Bước chuẩn hóa	Vai trò
1NF	Đảm bảo mỗi cột chỉ chứa một giá trị duy nhất, loại bỏ các nhóm dữ liệu lặp
2NF	Loại bỏ sự phụ thuộc một phần vào khóa chính, đảm bảo dữ liệu có ý nghĩa rõ ràng
3NF	Loại bỏ sự phụ thuộc bắc cầu giữa các thuộc tính không phải khóa chính
BCNF	Cải thiện 3NF bằng cách xử lý các trường hợp ngoại lệ có thể gây bất thường trong dữ liệu

6. Làm thế nào để đảm bảo rằng các phương thức của lớp tuân thủ nguyên lý hướng trách nhiệm?

- Áp dụng nguyên lý SRP: Mỗi phương thức chỉ nên thực hiện một nhiệm vụ duy nhất.
- Tách biệt các nhiệm vụ khác nhau: Nếu một phương thức xử lý quá nhiều logic, nên tách thành các phương thức nhỏ hơn.
- Sử dụng các lớp hỗ trợ: Nếu một lớp có quá nhiều phương thức, có thể tách thành nhiều lớp con.
- Áp dụng Interface Segregation Principle: Không ép các lớp triển khai các phương thức không cần thiết.

7. Thảo luận về ưu và nhược điểm của việc sử dụng sơ đồ lớp chi tiết trong thiết kế hệ thống.

- Ưu điểm:

+ Giúp hiểu rõ cấu trúc hệ thống

+ Hỗ trợ phát hiện lỗi trong thiết kế trước khi lập trình
+ Dễ dàng giao tiếp với nhóm phát triển và khách hàng
- Nhược điểm:
+ Mất nhiều thời gian để thiết kế
+ Cấu trúc phức tạp có thể làm sơ đồ khó đọc
+ Không phản ánh đầy đủ hành vi hoạt động của hệ thống (cần kết hợp sơ đồ tuần tự, sơ đồ trạng thái)

8. Tại sao cần chuẩn hóa cơ sở dữ liệu đến dạng 3-NF?

- Giảm dư thừa dữ liệu: Tránh trùng lặp thông tin không cần thiết.
- Đảm bảo tính nhất quán: Loại bỏ các bất thường khi chèn, cập nhật, xóa dữ liệu.
- Dễ dàng mở rộng hệ thống: Cấu trúc dữ liệu rõ ràng, dễ bảo trì

9. Thảo luận về tầm quan trọng của lớp điều khiển trong mô hình MVC. -

Lớp điều khiển đóng vai trò trung gian giữa Model và View trong MVC -

Chức năng chính:

+ Nhận yêu cầu từ người dùng
+ Xử lý logic nghiệp vụ bằng các tương tác với Model +

Gửi kết quả về View để hiển thị cho nguòi dùng.

- Lợi ích:

+ Giúp tách biệt logic nghiệp vụ và giao diện
+ Dễ dàng bảo trì và mở rộng
+ Hỗ trợ xử lý yêu cầu một cách có tổ chức

10.Làm thế nào để xây dựng sơ đồ lớp chi tiết cho một hệ thống lớn và phức tạp? -

Xác định các thực thể chính: Dựa trên yêu cầu hệ thống.

- Xác định thuộc tính và phương thức: Mô tả dữ liệu và hành vi của mỗi lớp.
- Xác định quan hệ giữa các lớp: Dùng các quan hệ như Association, Aggregation, Composition, Inheritance.
- Chia hệ thống thành các mô-đun nhỏ: Nhóm các lớp liên quan để dễ quản lý.
- Sử dụng ký hiệu UML chuẩn: Giúp biểu diễn rõ ràng các lớp và mối quan hệ.
- Kiểm tra và tối ưu: Đảm bảo thiết kế không dư thừa, không vi phạm nguyên tắc thiết kế hướng đối tượng.

IV. <u>Câu hỏi tình huống:</u>

1. Trong quá trình thiết kế hệ thống quản lý sinh viên, bạn nhận ra rằng nhiều lớp có các thuộc tính giống nhau. Bạn sẽ xử lý vấn đề này như thế nào?

- Giải pháp:

+ Sử dụng kế thừa (Inheritance): Tạo một lớp cha chứa các thuộc tính chung và cho các lớp con kế thừa.
+ Sử dụng Interface hoặc Abstract Class: Nếu chỉ cần đảm bảo các lớp có cùng hành vi mà không cần chia sẻ dữ liệu.
+ Dùng Composition thay vì Kế thừa: Nếu các thuộc tính chung là các thực thể riêng biệt, có thể dùng Aggregation hoặc Composition.

- Ví dụ: Nếu SinhVien và GiaoVien đều có thuộc tính HoTen, DiaChi, ta có thể tạo lớp Nguoi và kế thừa: SinhVien, GiaoVien

2. Sau khi hoàn thành sơ đồ lớp, nhóm phát triển phát hiện thiếu một số phương thức cần thiết. Làm thế nào để bổ sung vào sơ đồ lớp?

- Cập nhật sơ đồ lớp: Xác định vị trí hợp lý để thêm phương thức mà không phá vỡ cấu trúc.
- Kiểm tra nguyên tắc SOLID: Đảm bảo không làm vi phạm nguyên lý thiết kế.
- Liên hệ với nhóm phát triển: Đảm bảo rằng phương thức mới không ảnh hưởng đến các thành phần khác.
- Sử dụng công cụ UML để cập nhật: Các công cụ như StarUML, Enterprise Architect giúp dễ dàng chỉnh sửa sơ đồ.

3. Một nhóm phát triển gặp khó khăn trong việc phân biệt giữa Aggregation và Composition khi thiết kế sơ đồ lớp. Làm thế nào để giúp nhóm giải quyết vấn đề này?

- **Aggregation**: Quan hệ "có nhưng không sở hữu", đối tượng con có thể tồn tại độc lập.VD: Trường học có Giáo viên (Giáo viên vẫn tồn tại nếu trường bị đóng cửa).
- **Composition**: Quan hệ "có và sở hữu", đối tượng con phụ thuộc vào đối tượng cha.VD: xe hơi có Động cơ (Nếu xe bị hủy, động cơ cũng bị hủy theo)

4. Trong quá trình xây dựng thẻ CRC, nhóm phát triển phát hiện nhiều lớp có trách nhiệm trùng lặp. Làm thế nào để xử lý tình huống này?

- Tách bớt trách nhiệm: Nếu một lớp có quá nhiều trách nhiệm, có thể tách thành nhiều lớp nhỏ hơn.
- Hợp nhất các lớp có trách nhiệm giống nhau: Nếu hai lớp thực hiện nhiệm vụ giống nhau, có thể kết hợp chúng thành một.
- Xác định lại vai trò của từng lớp: Đảm bảo mỗi lớp chỉ có một vai trò chính (theo nguyên tắc Single Responsibility).

5. Khách hàng yêu cầu thêm một chức năng mới sau khi sơ đồ lớp đã được hoàn thiện. Nhóm phát triển cần làm gì để cập nhật sơ đồ lớp?

- Xác định ảnh hưởng của chức năng mới: Kiểm tra xem chức năng ảnh hưởng đến lớp nào. - Thêm lớp mới hoặc mở rộng lớp cũ: Nếu cần, có thể tạo lớp mới hoặc mở rộng lớp hiện tại bằng cách thêm phương thức mới.
- Cập nhật tài liệu UML: Đảm bảo rằng sơ đồ lớp phản ánh đúng hệ thống.
- Thông báo với nhóm phát triển: Để tránh ảnh hưởng đến tiến độ dự án.

6. Khi xây dựng sơ đồ FSM cho lớp DonHang, nhóm phát triển gặp khó khăn trong việc xác định các trạng thái và sự kiện. Bạn sẽ hướng dẫn nhóm như thế nào?

- Xác định các trạng thái chính: Ví dụ, Chờ xác nhận, Đã xác nhận, Đang giao hàng, Hoàn thành, Hủy.
- Xác định các sự kiện kích hoạt thay đổi trạng thái: Ví dụ, Thanh toán, Xác nhận đơn, Giao hàng thành công, Hủy đơn
- Vẽ sơ đồ FSM: Sử dụng công cụ UML để minh họa trạng thái và cách chuyển đổi giữa chúng.

7. Trong quá trình chuẩn hóa cơ sở dữ liệu, nhóm phát triển gặp vấn đề khi các bảng chứa nhiều dữ liệu dư thừa. Làm thế nào để xử lý vấn đề này?

- Giải pháp:
+ Kiểm tra dạng chuẩn:
1NF: đảm bảo mỗi cột chỉ chứa giá trị duy nhất
2NF: loại bỏ sự phụ thuộc một phần vào khóa chính
3NF: loại bỏ sự phụ thuộc bắc cầu giữa các thuộc tính không phải khóa
+ Tách bảng và tạo khóa ngoại: sử dụng FOREIGN KEY
- Ví dụ: Bảng SinhVien(MaSV, HoTen, Lop, TenLop, Khoa)
SinhVien(MaSV, HoTen, MaLop)
Lop(MaLop, TenLop, Khoa)

8. Một nhóm phát triển gặp khó khăn khi thiết kế các phương thức cho lớp vì chưa hiểu rõ nguyên lý hướng trách nhiệm. Làm thế nào để hướng dẫn nhóm áp dụng nguyên lý này?

- Giải thích nguyên lý Single Responsibility (SRP): Mỗi phương thức chỉ nên làm một nhiệm vụ.
- Tách phương thức lớn thành phương thức nhỏ hơn: Nếu một phương thức xử lý nhiều logic, nên chia nhỏ thành các phương thức chuyên biệt.
- Dùng Interface Segregation: Không ép một lớp thực hiện quá nhiều phương thức không liên quan.

9. Trong quá trình xây dựng sơ đồ lớp chi tiết, khách hàng yêu cầu thay đổi một số yêu cầu đã thống nhất trước đó. Nhóm phát triển nên xử lý như thế nào? - Xác định tác động của thay đổi: Kiểm tra xem thay đổi ảnh hưởng đến những lớp nào - Cập nhật sơ đồ lớp: Nếu cần, sửa đổi hoặc thêm mới các lớp/phương thức.
- Cập nhật tài liệu và giao tiếp với nhóm phát triển: Đảm bảo mọi người hiểu rõ thay đổi

- Sử dụng kỹ thuật phát triển linh hoạt (Agile): Giúp dễ dàng thích nghi với yêu cầu mới.

10. Khi kiểm tra lại sơ đồ lớp, nhóm phát triển nhận thấy một số quan hệ giữa các lớp bị sai. Làm thế nào để sửa lại sơ đồ lớp mà không ảnh hưởng đến tiến độ dự án?

- Phân tích lại quan hệ: Kiểm tra xem quan hệ giữa các lớp có phản ánh đúng thực tế không.
- Sử dụng nguyên tắc Open-Closed (OCP): Hạn chế thay đổi code hiện có, thay vào đó mở rộng bằng cách thêm lớp hoặc phương thức mới.
- Sử dụng công cụ UML để mô phỏng thay đổi: Dùng StarUML hoặc Lucidchart để kiểm tra trước khi áp dụng vào code.
- Kiểm tra ảnh hưởng của thay đổi: Nếu có thể, thử nghiệm trước trên một mô hình nhỏ để đảm bảo không làm gián đoạn hệ thống.