Bài tập buổi 7: Thiết kế

I. CÂU HỎI TRẮC NGHIỆM:

1. 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

2. 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

3. 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

4. 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

5. 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

6. 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

7. 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

8. 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
9. 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
B. Thêm các thuộc tính và phương thức đầy đủ cho từng lớp
C. Thêm giao diện người dùng
D. Thêm các sơ đồ tuần tự
10.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. <u>CÂU HỎI TRẢ LỜI NGẮN:</u>

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

Đóng gói dữ liệu (encapsulation) giúp che giấu chi tiết cài đặt bên trong lớp, bảo vệ dữ liệu khỏi truy cập trái phép, đảm bảo tính toàn vẹn và cho phép kiểm soát truy cập thông qua các phương thức getter/setter.

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

Nguyên lý hướng trách nhiệm (Single Responsibility Principle) yêu cầu mỗi lớp hoặc phương thức chỉ đảm nhiệm một nhiệm vụ cụ thể, giúp mã nguồn dễ bảo trì, mở rộng và giảm độ phức tạp.

3. Thẻ CRC là gì?

Thẻ CRC (Class-Responsibility-Collaboration) là công cụ thiết kế hướng đối tượng, mô tả tên lớp, trách nhiệm (chức năng) của lớp và các lớp khác mà nó tương tác (collaboration).

4. Quan hệ Aggregation và Composition khác nhau như thế nào?
· **Aggregation**: Mối quan hệ "has-a" yếu, phần tử con có thể tồn tại độc lập với phần tử cha (ví dụ: Car và Wheel).
· **Composition**: Mối quan hệ "has-a" mạnh, phần tử con không thể tồn tại nếu phần tử cha bị hủy (ví dụ: House và Room).
5. FSM (Finite State Machine) là gì?

FSM là mô hình toán học mô tả các trạng thái của một đối tượng, các chuyển đổi giữa trạng thái dựa trên sự kiện/đầu vào, thường dùng để thiết kế hệ thống điều khiển hoặc mô phỏng hành vi.

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

Chuẩn hóa cơ sở dữ liệu nhằm loại bỏ dư thừa dữ liệu, đảm bảo tính nhất quán, giảm lỗi khi thêm/xóa/sửa dữ liệu và tối ưu hóa cấu trúc lưu trữ.

7. Lớp điều khiển có vai trò gì trong hệ thống?

Lớp điều khiển (Controller) xử lý logic nghiệp vụ, điều phối luồng dữ liệu giữa lớp giao diện (View) và lớp thực thể (Model), đảm bảo các thành phần hệ thống hoạt động đồng bộ.

8. Nguyên lý thực hiện lời gọi nhiều lần giúp ích gì trong thiết kế phương thức?

Nguyên lý này khuyến khích thiết kế phương thức tái sử dụng được trong nhiều ngữ cảnh, giảm trùng lặp mã nguồn, tăng hiệu quả lập trình và dễ bảo trì.

9. Sơ đồ lớp là gì?

Sơ đồ lớp (Class Diagram) là biểu đồ UML mô tả các lớp trong hệ thống, bao gồm thuộc tính, phương thức và các quan hệ giữa chúng (association, aggregation, composition, inheritance).

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

Dạng chuẩn 3-NF (Third Normal Form) là trạng thái cơ sở dữ liệu đã đạt 1NF, 2NF và loại bỏ các phụ thuộc bắc cầu (transitive dependencies), đảm bảo không có thuộc tính phụ thuộc gián tiếp vào khóa chính.

III. CÂU HỎI THẢO LUẬN NHÓM:

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

· **Vai trò**: Đóng gói (encapsulation) bảo vệ dữ liệu bằng cách ẩn chi tiết cài đặt, chỉ cho phép truy cập qua các phương thức công khai (getter/setter). Điều này ngăn chặn sửa đổi dữ liệu trái phép, tăng tính bảo mật và toàn vẹn.

· **Lợi ích**: Giảm lỗi khi thay đổi mã, dễ bảo trì, hỗ trợ mở rộng hệ thống mà không ảnh hưởng đến các lớp khác.

· **Ví dụ**: Trong lớp SinhVien, thuộc tính diemTrungBinh là private, chỉ có thể thay đổi qua phương thức setDiemTrungBinh() với kiểm tra hợp lệ.

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

· **Aggregation**: Mối quan hệ "has-a" yếu, phần tử con có thể tồn tại độc lập (ví dụ: lớp University và lớp Student). Nếu University bị xóa, Student vẫn tồn tại.

· **Composition**: Mối quan hệ "has-a" mạnh, phần tử con phụ thuộc vào phần tử cha (ví dụ: lớp Car và lớp Engine). Nếu Car bị xóa, Engine cũng bị xóa.

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

· **Cách xây dựng**:

1. Xác định các lớp chính (ví dụ: SinhVien, Sach, ThuVien).
2. Liệt kê trách nhiệm cụ thể của mỗi lớp (ví dụ: SinhVien mượn/trả sách).
3. Xác định các lớp tương tác (collaboration) để thực hiện trách nhiệm.

· **Hiệu quả**: Sử dụng bảng CRC để thảo luận nhóm trước khi viết mã, đảm bảo trách nhiệm rõ ràng, tránh trùng lặp.

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

· **Lý do**: Sơ đồ FSM (Finite State Machine) mô tả các trạng thái và chuyển đổi của đối tượng, giúp thiết kế hành vi động của lớp (ví dụ: trạng thái đơn hàng: Pending, Shipped, Delivered).
· **Lợi ích**: Dễ dàng debug, kiểm soát logic phức tạp, và đảm bảo hệ thống hoạt động đúng theo yêu cầu.
5.Thảo luận về các bước chuẩn hóa cơ sở dữ liệu và vai trò của từng bước.
· **Các bước chuẩn hóa**:

- **1NF**: Loại bỏ giá trị lặp, mỗi ô chứa một giá trị duy nhất.
- **2NF**: Loại bỏ phụ thuộc một phần, đảm bảo mọi thuộc tính phụ thuộc hoàn toàn vào khóa chính.
- **3NF**: Loại bỏ phụ thuộc bắc cầu, đảm bảo không có thuộc tính phụ thuộc gián tiếp vào khóa chính.

· **Vai trò**:

- 1NF đảm bảo dữ liệu nguyên tử, dễ truy vấn.
- 2NF giảm dư thừa trong bảng có khóa chính phức hợp.
- 3NF tăng tính nhất quán, giảm lỗi khi cập nhật dữ liệu.

6. Làm thế nào để đảm bảo rằng các phương thức của lớp tuân thủ nguyên lý hướng trách nhiệm?
· **Cách thực hiện**:

- Mỗi phương thức chỉ thực hiện một chức năng cụ thể.
- Phân tách logic phức tạp thành các lớp/phương thức riêng (ví dụ: tách tính điểm và lưu trữ dữ liệu).
- Sử dụng thẻ CRC để xác định trách nhiệm rõ ràng.

· **Lợi ích**: Giảm độ phức tạp, dễ bảo trì và kiểm tra.
7. Thảo luận về ưu và nhược điểm của việc sử dụng sơ đồ lớp chi tiết trong thiết kế hệ thống.
· **Ưu điểm**:

- Cung cấp cái nhìn tổng quan về cấu trúc hệ thống.
- Hỗ trợ giao tiếp giữa các thành viên nhóm và khách hàng.
- Dễ dàng phát hiện lỗi thiết kế trước khi viết mã.

· **Nhược điểm**:

- Tốn thời gian để xây dựng và cập nhật.
- Có thể trở nên phức tạp với hệ thống lớn.

8. Tại sao cần chuẩn hóa cơ sở dữ liệu đến dạng 3-NF?
· **Lý do**: 3-NF loại bỏ dư thừa dữ liệu, đảm bảo tính nhất quán, giảm lỗi khi thêm/xóa/sửa dữ liệu, và tối ưu hóa truy vấn.

· **Ví dụ**: Trong bảng StudentID, Course, Instructor, chuẩn hóa đến 3-NF tách InstructorPhone thành bảng riêng để tránh dư thừa.

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

· **Vai trò**: Lớp điều khiển (Controller) trong MVC điều phối dữ liệu giữa View (giao diện) và Model (dữ liệu/logic), đảm bảo tách biệt trách nhiệm.

· **Tầm quan trọng**: Giúp hệ thống dễ mở rộng, bảo trì, và kiểm tra. Ví dụ: Controller xử lý yêu cầu đăng nhập từ View và gọi Model để xác thực.

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

· Phân chia hệ thống thành các module nhỏ (ví dụ: User Management, Payment).

· Xây dựng sơ đồ lớp cho từng module trước, sau đó tích hợp.

· Sử dụng công cụ UML như ArgoUML, StarUML để quản lý độ phức tạp.

· Thường xuyên xem xét và cập nhật với nhóm và khách hàng.

IV. CÂU HỎI TÌNH HUỐNG:

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?

- Phân tích thuộc tính trùng lặp
- Sử dụng lớp kế thừa
- Áp dụng nguyên lý DRY (Don’t Rêpat Yourself)
- Xem sét sử dụng Composition nếu kế thừa không phù hợp
- Cập nhật Class Diagram
- Kiểm tra và xác nhận
- Tài liệu hóa thay đổi

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?

· **Bước 1**: Xác định các phương thức thiếu dựa trên yêu cầu hoặc trách nhiệm của lớp (sử dụng thẻ CRC để kiểm tra).

· **Bước 2**: Cập nhật sơ đồ lớp bằng cách thêm phương thức vào các lớp liên quan, đảm bảo tuân thủ nguyên lý hướng trách nhiệm.

· **Bước 3**: Xem xét tác động đến các quan hệ giữa các lớp và cập nhật nếu cần (ví dụ: thêm phương thức có thể yêu cầu thêm quan hệ).

· **Bước 4**: Thảo luận với nhóm và khách hàng để xác nhận thay đổi, sau đó cập nhật tài liệu thiết kế.

· **Ví dụ**: Nếu lớp DonHang thiếu phương thức huyDonHang(), thêm phương thức này và kiểm tra tác động đến lớp KhachHang.

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?

· **Giải thích rõ ràng**:

- Aggregation: Mối quan hệ lỏng lẻo, phần tử con tồn tại độc lập (ví dụ: Library và Book).
- Composition: Mối quan hệ chặt chẽ, phần tử con bị hủy khi phần tử cha hủy (ví dụ: House và Room).

· **Cung cấp ví dụ thực tế**: Minh họa bằng hệ thống nhóm đang phát triển (ví dụ: trong hệ thống quản lý cửa hàng, Product và Order là Aggregation vì Product tồn tại độc lập).
· **Tổ chức workshop**: Yêu cầu nhóm phân tích các lớp trong dự án và xác định mối quan hệ dựa trên vòng đời của đối tượng.
· **Sử dụng công cụ UML**: Vẽ sơ đồ để trực quan hóa và thảo luận.
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?
· **Bước 1**: Xem xét lại các trách nhiệm trùng lặp và xác định lớp nào phù hợp nhất để đảm nhận.
· **Bước 2**: Tái cấu trúc thẻ CRC, phân chia trách nhiệm rõ ràng dựa trên nguyên lý hướng trách nhiệm (SRP).
· **Bước 3**: Nếu cần, tạo lớp mới để xử lý trách nhiệm chung (ví dụ: lớp DatabaseHelper cho lưu trữ dữ liệu).
· **Bước 4**: Thảo luận nhóm để thống nhất và cập nhật thẻ CRC, sau đó kiểm tra tính hợp lý với yêu cầu hệ thống.
· **Ví dụ**: Nếu cả SinhVien và MonHoc đều có trách nhiệm tính điểm, chuyển trách nhiệm này sang lớp DiemSinhVien.
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?
· **Bước 1**: Phân tích chức năng mới, xác định lớp/phương thức/thuộc tính cần thêm.
· **Bước 2**: Cập nhật sơ đồ lớp, bổ sung lớp mới hoặc chỉnh sửa lớp hiện có, đảm bảo không phá vỡ cấu trúc hiện tại.
· **Bước 3**: Kiểm tra tác động đến các quan hệ và phương thức khác, cập nhật tài liệu thiết kế.
· **Bước 4**: Thảo luận với khách hàng để xác nhận và với nhóm để đảm bảo tính khả thi.
· **Ví dụ**: Nếu thêm chức năng gửi thông báo, có thể thêm lớp Notification và phương thức sendNotification().
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 trạng thái**: Liệt kê các trạng thái chính của DonHang, ví dụ: Pending, Confirmed, Shipped, Delivered, Cancelled.
· **Xác định sự kiện**: Xác định các sự kiện gây chuyển trạng thái, ví dụ: "Khách hàng xác nhận" (Pending → Confirmed), "Hủy đơn" (Pending → Cancelled).
· **Vẽ sơ đồ FSM**: Sử dụng công cụ như StarUML để vẽ trạng thái và mũi tên chuyển đổi, kèm điều kiện/sự kiện.
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?
· **Bước 1**: Phân tích dữ liệu dư thừa (ví dụ: thông tin Instructor lặp lại trong bảng Student).
· **Bước 2**: Áp dụng chuẩn hóa:

- 1NF: Loại bỏ giá trị lặp, chia thành các bảng nhỏ.

- 2NF: Đảm bảo thuộc tính phụ thuộc hoàn toàn vào khóa chính.
- 3NF: Tách các thuộc tính phụ thuộc bắc cầu (ví dụ: tách InstructorPhone thành bảng Instructor).

· **Bước 3**: Viết lại câu lệnh SQL để tạo bảng mới và chuyển dữ liệu.
· **Bước 4**: Kiểm tra tính nhất quán và hiệu suất truy vấn.
· **Ví dụ**: Từ bảng StudentID, Course, Instructor, tách thành bảng Instructor riêng với khóa ngoại liên kết.
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 SRP**: Mỗi phương thức/lớp chỉ nên có một trách nhiệm duy nhất.
· **Hướng dẫn thực hành**:

- Yêu cầu nhóm liệt kê trách nhiệm của lớp (ví dụ: SinhVien lưu thông tin, tính điểm).
- Tách trách nhiệm phức tạp thành các lớp/phương thức riêng (ví dụ: lớp DiemSinhVien cho tính điểm).
- Sử dụng thẻ CRC để xác định trách nhiệm rõ ràng.

· **Ví dụ**: Nếu lớp SinhVien vừa lưu thông tin vừa xử lý đăng ký môn học, tách thành lớp DangKyHoc.
· **Workshop**: Tổ chức phân tích một lớp cụ thể và tái cấu trúc phương thức.
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?
· **Bước 1**: Ghi nhận và phân tích yêu cầu mới, xác định tác động đến sơ đồ lớp (thêm/xóa lớp, phương thức, quan hệ).
· **Bước 2**: Cập nhật sơ đồ lớp, đảm bảo tuân thủ các nguyên lý thiết kế (SRP, encapsulation).
· **Bước 3**: Thảo luận với khách hàng để xác nhận thay đổi và đánh giá tác động đến tiến độ/ngân sách.
· **Bước 4**: Cập nhật tài liệu và thông báo cho nhóm để đồng bộ.
· **Ví dụ**: Nếu khách hàng yêu cầu thêm quản lý khuyến mãi, thêm lớp KhuyenMai và cập nhật quan hệ với DonHang.
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?
· **Bước 1**: Xác định các quan hệ sai (ví dụ: Aggregation thay vì Composition).
· **Bước 2**: Sửa sơ đồ lớp, tập trung vào các quan hệ sai mà không thay đổi cấu trúc tổng thể.
· **Bước 3**: Sử dụng công cụ UML để cập nhật nhanh và kiểm tra tính hợp lệ.
· **Bước 4**: Phân công nhiệm vụ sửa đổi cho một thành viên trong khi nhóm tiếp tục các công việc khác để tránh chậm tiến độ.
· **Bước 5**: Xem xét và xác nhận với nhóm và khách hàng.
· **Ví dụ**: Nếu quan hệ giữa KhachHang và DonHang sai (Association thay vì Composition), sửa lại và kiểm tra tác động đến phương thức.