Họ và tên: Lâm Thành Trung  
MSSV: N18DCAT096

Nhóm: 10  
Bài tập tuần 3 – Nhập môn công nghệ phần mềm – D22CQAT01-N  
  
**Câu hỏi chương 6:  
Câu hỏi trắc nghiệm:**

**Câu hỏi 1: Lớp nào dưới đây đại diện cho các đối tượng có dữ liệu và trạng thái cần quản lý trong hệ thống?**

**Đáp án: C. Lớp thực thể**

Giải thích: Lớp thực thể (Entity Class) đại diện cho các đối tượng chính trong hệ thống, lưu trữ dữ liệu và trạng thái của chúng.

**Câu hỏi 2: Lớp biên trong hệ thống có vai trò gì?**

**Đáp án: C. Giao tiếp với người dùng hoặc hệ thống bên ngoài**

Giải thích: Lớp biên (Boundary Class) đóng vai trò là giao diện giữa hệ thống và các tác nhân bên ngoài, bao gồm người dùng và các hệ thống khác.

**Câu hỏi 3: Quan hệ nào giữa các lớp thể hiện sự kế thừa?**

**Đáp án: D. Inheritance**

Giải thích: Quan hệ kế thừa (Inheritance) cho phép một lớp con kế thừa các thuộc tính và phương thức từ lớp cha.

**Câu hỏi 4: Sơ đồ lớp mô tả:**

**Đáp án: B. Các lớp và quan hệ giữa các lớp trong hệ thống**

Giải thích: Sơ đồ lớp (Class Diagram) mô tả cấu trúc tĩnh của hệ thống, bao gồm các lớp, thuộc tính, phương thức và mối quan hệ giữa chúng.

**Câu hỏi 5: Quan hệ Include giữa các use case được dùng khi:**

**Đáp án: B. Một use case cần gọi một use case khác để hoàn thành chức năng**

Giải thích: Quan hệ Include thể hiện việc một use case bắt buộc phải sử dụng một use case khác để hoàn thành chức năng của mình.

**Câu hỏi 6: Scenario là gì?**

**Đáp án: B. Một kịch bản mô tả cách hệ thống và người dùng tương tác**

Giải thích: Scenario là một kịch bản chi tiết mô tả một chuỗi các tương tác giữa người dùng và hệ thống để đạt được một mục tiêu cụ thể.

**Câu hỏi 7: Quan hệ nào sau đây biểu diễn việc một lớp chứa một lớp khác nhưng lớp con vẫn có thể tồn tại độc lập?**

**Đáp án: A. Aggregation**

Giải thích: Aggregation thể hiện mối quan hệ "có một", trong đó lớp con có thể tồn tại độc lập với lớp cha.

**Câu hỏi 8: Sơ đồ tuần tự mô tả điều gì?**

**Đáp án: B. Thứ tự các thông điệp được trao đổi giữa các đối tượng**

Giải thích: Sơ đồ tuần tự (Sequence Diagram) mô tả luồng tương tác giữa các đối tượng theo thời gian.

**Câu hỏi 9: Lớp điều khiển trong mô hình MVC tương ứng với thành phần nào?**

**Đáp án: C. Control**

Giải thích: Trong mô hình MVC, lớp điều khiển (Controller) xử lý các tương tác của người dùng và điều phối luồng dữ liệu giữa Model và View.

**Câu hỏi 10: Để biểu diễn quan hệ giữa các lớp, ta sử dụng sơ đồ nào?**

**Đáp án: C. Sơ đồ lớp**

Giải thích: Sơ đồ lớp (Class Diagram) là công cụ chính để mô tả cấu trúc và mối quan hệ giữa các lớp trong hệ thống.

**Câu hỏi ngắn:**

1. Lớp thực thể là gì?

* Lớp thực thể (Entity Class) đại diện cho các đối tượng chính trong hệ thống, lưu trữ dữ liệu và trạng thái của chúng.
* Chúng thường tương ứng với các đối tượng trong thế giới thực hoặc các khái niệm nghiệp vụ quan trọng.
* Ví dụ: Lớp "Khách hàng", "Sản phẩm", "Đơn hàng".

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

* Lớp điều khiển (Control Class) điều phối luồng công việc và tương tác giữa các lớp thực thể và lớp biên.
* Nó nhận yêu cầu từ lớp biên, xử lý logic nghiệp vụ và cập nhật dữ liệu trong lớp thực thể.
* Lớp điều khiển giúp tách biệt logic nghiệp vụ khỏi giao diện người dùng và dữ liệu.

3. Scenario là gì?

* Scenario là một kịch bản chi tiết mô tả một chuỗi các tương tác giữa người dùng và hệ thống để đạt được một mục tiêu cụ thể.
* Nó thường được sử dụng để minh họa các use case và giúp hiểu rõ hơn về cách hệ thống hoạt động trong các tình huống thực tế.
* Ví dụ: Scenario "Khách hàng đăng nhập vào hệ thống và đặt mua sản phẩm".

4. Quan hệ Include giữa các use case là gì?

* Quan hệ Include thể hiện việc một use case bắt buộc phải sử dụng một use case khác để hoàn thành chức năng của mình.
* Use case được include thường là các chức năng chung, được sử dụng bởi nhiều use case khác.
* Ví dụ: Use case "Thanh toán" include use case "Xác thực thông tin thẻ".

5. Mục đích của sơ đồ lớp là gì?

* Sơ đồ lớp (Class Diagram) mô tả cấu trúc tĩnh của hệ thống, bao gồm các lớp, thuộc tính, phương thức và mối quan hệ giữa chúng.
* Nó giúp hiểu rõ hơn về thiết kế của hệ thống và là cơ sở cho việc lập trình.
* Sơ đồ lớp cũng được sử dụng để giao tiếp giữa các thành viên trong nhóm phát triển và với khách hàng.

6. Quan hệ Aggregation khác gì so với Composition?

* Aggregation: Thể hiện mối quan hệ "có một", trong đó lớp con có thể tồn tại độc lập với lớp cha. Ví dụ: Lớp "Lớp học" có thể chứa nhiều lớp "Sinh viên", nhưng sinh viên vẫn có thể tồn tại khi lớp học không còn.
* Composition: Thể hiện mối quan hệ "phần của", trong đó lớp con không thể tồn tại độc lập với lớp cha. Ví dụ: Lớp "Ô tô" chứa lớp "Động cơ", và động cơ không thể tồn tại nếu không có ô tô.

7. Sơ đồ tuần tự là gì?

* Sơ đồ tuần tự (Sequence Diagram) mô tả luồng tương tác giữa các đối tượng theo thời gian.
* Nó thể hiện thứ tự các thông điệp được trao đổi giữa các đối tượng để thực hiện một use case hoặc scenario.
* Sơ đồ tuần tự giúp hiểu rõ hơn về cách các đối tượng tương tác với nhau trong một tình huống cụ thể.

8. Quan hệ Extend giữa các use case là gì?

* Quan hệ Extend cho phép một use case mở rộng hành vi của một use case khác dưới các điều kiện cụ thể.
* Use case mở rộng (extension) chỉ được thực hiện khi đáp ứng các điều kiện nhất định.
* Ví dụ: Use case "Thanh toán" extend use case "Xử lý mã giảm giá" khi khách hàng có mã giảm giá.

9. Lớp biên có vai trò gì trong hệ thống?

* Lớp biên (Boundary Class) đóng vai trò là giao diện giữa hệ thống và các tác nhân bên ngoài, bao gồm người dùng và các hệ thống khác.
* Nó xử lý các tương tác của người dùng, chuyển đổi dữ liệu đầu vào và đầu ra, và hiển thị thông tin cho người dùng.
* Ví dụ: Lớp giao diện người dùng (GUI), lớp API.

10. Sơ đồ cộng tác là gì?

* Sơ đồ cộng tác (Collaboration Diagram) mô tả tương tác giữa các đối tượng thông qua việc gửi và nhận thông điệp.
* Nó tập trung vào mối quan hệ giữa các đối tượng và cách chúng phối hợp với nhau để thực hiện một chức năng.
* Sơ đồ cộng tác tương tự như sơ đồ tuần tự, nhưng nó tập trung vào cấu trúc liên kết giữa các đối tượng hơn là trình tự thời gian.

**Câu hỏi thảo luận nhóm:**

1. Thảo luận về vai trò của từng loại lớp (thực thể, biên, điều khiển) trong hệ thống.

* Lớp thực thể (Entity Class):
  + Đại diện cho các đối tượng dữ liệu cốt lõi của hệ thống.
  + Lưu trữ thông tin và trạng thái của đối tượng.
  + Ví dụ: Lớp "Khách hàng", "Sản phẩm", "Đơn hàng".
* Lớp biên (Boundary Class):
  + Xử lý giao tiếp giữa hệ thống và các tác nhân bên ngoài (người dùng, hệ thống khác).
  + Đảm nhiệm việc chuyển đổi dữ liệu đầu vào và đầu ra.
  + Ví dụ: Giao diện người dùng (GUI), API.
* Lớp điều khiển (Control Class):
  + Điều phối luồng công việc và tương tác giữa các lớp thực thể và lớp biên.
  + Thực hiện logic nghiệp vụ của hệ thống.
  + Giúp tách biệt logic nghiệp vụ khỏi giao diện và dữ liệu.

2. So sánh sự khác nhau giữa Aggregation và Composition.

* Aggregation:
  + Mối quan hệ "có một".
  + Lớp con có thể tồn tại độc lập với lớp cha.
  + Ví dụ: "Lớp học" có "Sinh viên", sinh viên có thể tồn tại khi lớp học đóng cửa.
* Composition:
  + Mối quan hệ "phần của".
  + Lớp con không thể tồn tại độc lập với lớp cha.
  + Ví dụ: "Ô tô" có "Động cơ", động cơ không thể tồn tại nếu không có ô tô.

3. Thảo luận về tầm quan trọng của việc xây dựng sơ đồ lớp trong quá trình phân tích hệ thống.

* Giúp hiểu rõ cấu trúc tĩnh của hệ thống.
* Xác định các lớp, thuộc tính, và mối quan hệ giữa chúng.
* Là cơ sở cho việc thiết kế và lập trình.
* Hỗ trợ giao tiếp giữa các thành viên trong nhóm phát triển.

4. Phân biệt sơ đồ tuần tự và sơ đồ cộng tác.

* Sơ đồ tuần tự (Sequence Diagram):
  + Tập trung vào trình tự thời gian của các thông điệp trao đổi giữa các đối tượng.
  + Dễ dàng theo dõi luồng thực thi.
* Sơ đồ cộng tác (Collaboration Diagram):
  + Tập trung vào mối quan hệ giữa các đối tượng và cách chúng phối hợp với nhau.
  + Dễ dàng thấy được cấu trúc liên kết giữa các đối tượng.

5. Thảo luận về vai trò của lớp điều khiển trong mô hình MVC.

* Lớp điều khiển (Controller) nhận yêu cầu từ người dùng (qua View).
* Điều phối luồng dữ liệu giữa Model và View.
* Thực hiện logic nghiệp vụ để xử lý yêu cầu.
* Giúp tách biệt logic nghiệp vụ khỏi giao diện và dữ liệu.

6. Tại sao cần viết các scenario khi phân tích hệ thống?

* Scenario mô tả chi tiết các tương tác giữa người dùng và hệ thống.
* Giúp hiểu rõ hơn về các use case và yêu cầu của hệ thống.
* Hỗ trợ việc kiểm thử và xác thực hệ thống.
* Giúp phát hiện các tình huống và luồng xử lý đặc biệt.

7. Làm thế nào để đảm bảo rằng các use case được trích đầy đủ và chính xác?

* Phỏng vấn kỹ lưỡng các bên liên quan.
* Quan sát quy trình làm việc thực tế.
* Sử dụng các kỹ thuật như brainstorming, use case workshop.
* Xác định rõ các actor và mục tiêu của họ.
* Kiểm tra và xác nhận use case với khách hàng.

8. Thảo luận về mối quan hệ giữa use case và scenario.

* Use case mô tả mục tiêu tổng quát của người dùng.
* Scenario mô tả chi tiết một tình huống cụ thể của use case.
* Scenario giúp minh họa và làm rõ các bước thực hiện của use case.
* Một use case có thể có nhiều scenario khác nhau.

9. Phân tích ưu và nhược điểm của việc sử dụng sơ đồ tuần tự trong thiết kế hệ thống.

* Ưu điểm:
  + Dễ dàng theo dõi luồng thực thi.
  + Hiển thị rõ ràng thứ tự các thông điệp.
  + Hỗ trợ việc kiểm tra và gỡ lỗi.
* Nhược điểm:
  + Có thể trở nên phức tạp khi hệ thống có nhiều đối tượng.
  + Khó thấy được cấu trúc tổng thể của hệ thống.

10. Thảo luận về cách cải thiện chất lượng kịch bản sử dụng (scenario) trong quá trình phân tích.

* Sử dụng ngôn ngữ rõ ràng, dễ hiểu.
* Mô tả chi tiết các bước thực hiện và điều kiện.
* Bao gồm các luồng sự kiện thay thế và xử lý lỗi.
* Kiểm tra và xác nhận scenario với khách hàng.
* Sử dụng các công cụ hỗ trợ để quản lý và theo dõi scenario.

**Câu hỏi tình huống:**1. Trong quá trình phân tích hệ thống quản lý thư viện, nhóm phát triển phát hiện một số yêu cầu mới từ khách hàng sau khi đã viết xong các scenario. Nhóm phát triển nên xử lý như thế nào?

* Đánh giá tác động: Đánh giá tác động của các yêu cầu mới đến các scenario đã viết, cũng như đến thời gian, chi phí và nguồn lực của dự án.
* Thảo luận với khách hàng: Thảo luận với khách hàng về lý do và mức độ quan trọng của các yêu cầu mới.
* Cập nhật scenario: Cập nhật các scenario đã viết để phản ánh các yêu cầu mới.
* Thêm scenario mới: Thêm các scenario mới nếu cần thiết để mô tả đầy đủ các yêu cầu mới.
* Thông báo cho các bên liên quan: Thông báo cho các bên liên quan về các thay đổi trong yêu cầu và scenario.

2. Một nhóm phát triển gặp khó khăn khi xác định các lớp điều khiển trong hệ thống. Hãy đề xuất giải pháp.

* Phân tích use case: Phân tích kỹ lưỡng các use case để xác định các logic nghiệp vụ cần thiết.
* Xác định các tương tác: Xác định các tương tác giữa các lớp thực thể và lớp biên.
* Tìm kiếm các hành vi điều phối: Tìm kiếm các hành vi điều phối và logic nghiệp vụ không thuộc về lớp thực thể hoặc lớp biên.
* Sử dụng mẫu thiết kế: Sử dụng các mẫu thiết kế như Controller hoặc Facade để xác định lớp điều khiển.
* Thảo luận với các thành viên: Thảo luận với các thành viên khác trong nhóm để thống nhất về các lớp điều khiển.

3. Sau khi hoàn thành sơ đồ lớp, khách hàng yêu cầu thêm một số chức năng mới. Nhóm phát triển cần làm gì để cập nhật sơ đồ lớp?

* Đánh giá tác động: Đánh giá tác động của các chức năng mới đến sơ đồ lớp hiện tại.
* Thêm lớp mới: Thêm các lớp mới nếu cần thiết để biểu diễn các chức năng mới.
* Cập nhật quan hệ: Cập nhật các quan hệ giữa các lớp để phản ánh các chức năng mới.
* Cập nhật thuộc tính và phương thức: Cập nhật các thuộc tính và phương thức của các lớp hiện tại để hỗ trợ các chức năng mới.
* Thảo luận với khách hàng: Thảo luận với khách hàng về các thay đổi trong sơ đồ lớp.

4. Khi viết các scenario cho use case "Đăng ký khóa học", nhóm phát triển gặp tình huống có nhiều trường hợp ngoại lệ. Làm thế nào để xử lý tình huống này?

* Xác định các trường hợp ngoại lệ: Xác định rõ ràng các trường hợp ngoại lệ có thể xảy ra.
* Mô tả các luồng sự kiện thay thế: Mô tả chi tiết các luồng sự kiện thay thế cho mỗi trường hợp ngoại lệ.
* Sử dụng các điều kiện: Sử dụng các điều kiện để xác định khi nào một trường hợp ngoại lệ xảy ra.
* Sử dụng các thông báo lỗi: Sử dụng các thông báo lỗi để thông báo cho người dùng về các trường hợp ngoại lệ.
* Kiểm tra kỹ lưỡng: Kiểm tra kỹ lưỡng các scenario để đảm bảo chúng xử lý đúng các trường hợp ngoại lệ.

5. Trong quá trình xây dựng sơ đồ tuần tự, một số đối tượng không có vai trò rõ ràng. Nhóm phát triển nên làm gì?

* Xem xét lại use case: Xem xét lại use case và các scenario để xác định vai trò của các đối tượng.
* Phân tích các thông điệp: Phân tích các thông điệp được trao đổi giữa các đối tượng để xác định trách nhiệm của chúng.
* Loại bỏ đối tượng không cần thiết: Loại bỏ các đối tượng không cần thiết để đơn giản hóa sơ đồ.
* Thảo luận với các thành viên: Thảo luận với các thành viên khác trong nhóm để thống nhất về vai trò của các đối tượng.
* Cập nhật sơ đồ: Cập nhật sơ đồ tuần tự để phản ánh các thay đổi.

6. Sau khi xây dựng sơ đồ lớp, nhóm phát triển phát hiện ra một số quan hệ giữa các lớp bị sai. Hãy đề xuất cách sửa chữa.

* Phân tích quan hệ sai: Phân tích kỹ lưỡng các quan hệ sai để xác định lỗi.
* Xác định quan hệ đúng: Xác định quan hệ đúng giữa các lớp.
* Sử dụng ký hiệu đúng: Sử dụng các ký hiệu đúng để biểu diễn quan hệ.
* Cập nhật sơ đồ: Cập nhật sơ đồ lớp để sửa chữa các quan hệ sai.
* Kiểm tra kỹ lưỡng: Kiểm tra kỹ lưỡng sơ đồ lớp để đảm bảo tất cả các quan hệ đều đúng.

7. Một nhóm phát triển gặp khó khăn khi mô tả các quan hệ giữa các use case. Hãy đề xuất giải pháp.

* Sử dụng quan hệ Include: Sử dụng quan hệ Include để mô tả các use case được sử dụng chung.
* Sử dụng quan hệ Extend: Sử dụng quan hệ Extend để mô tả các use case mở rộng.
* Sử dụng quan hệ Generalization: Sử dụng quan hệ Generalization để mô tả các use case kế thừa.
* Vẽ sơ đồ use case: Vẽ sơ đồ use case để trực quan hóa các quan hệ.
* Thảo luận với các thành viên: Thảo luận với các thành viên khác trong nhóm để thống nhất về các quan hệ.

8. Trong dự án phát triển phần mềm quản lý bán hàng, nhóm phát triển cần xác định các lớp biên cho hệ thống. Hãy đưa ra đề xuất phù hợp.

* Lớp giao diện người dùng (GUI): Cho phép người dùng tương tác với hệ thống.
* Lớp API: Cho phép các hệ thống khác tương tác với hệ thống.
* Lớp báo cáo: Tạo ra các báo cáo cho người dùng.
* Lớp in ấn: In hóa đơn và các tài liệu khác.
* Lớp thanh toán: Xử lý các giao dịch thanh toán.

9. Khách hàng yêu cầu thêm chức năng mới sau khi các scenario đã được hoàn thiện. Nhóm phát triển nên làm gì?

* Đánh giá tác động: Đánh giá tác động của các chức năng mới đến các scenario đã viết.
* Thảo luận với khách hàng: Thảo luận với khách hàng về lý do và mức độ quan trọng của các chức năng mới.
* Cập nhật scenario: Cập nhật các scenario đã viết để phản ánh các chức năng mới.
* Thêm scenario mới: Thêm các scenario mới nếu cần thiết để mô tả đầy đủ các chức năng mới.
* Thông báo cho các bên liên quan: Thông báo cho các bên liên quan về các thay đổi trong yêu cầu và scenario.

10. Trong quá trình xây dựng sơ đồ cộng tác, một số đối tượng không tương tác đúng theo yêu cầu. Hãy đề xuất cách giải quyết.

* Xem xét lại use case: Xem xét lại use case và các scenario để xác định tương tác đúng.
* Phân tích các thông điệp: Phân tích các thông điệp được trao đổi giữa các đối tượng để xác định lỗi.
* Sửa đổi sơ đồ: Sửa đổi sơ đồ cộng tác để phản ánh tương tác đúng.
* Kiểm tra kỹ lưỡng: Kiểm tra kỹ lưỡng sơ đồ cộng tác để đảm bảo tất cả các tương tác đều đúng.
* Thảo luận với các thành viên: Thảo luận với các thành viên khác trong nhóm để thống nhất về các tương tác.

**REVIEW LESSION:**

**Câu hỏi trắc nghiệm:  
Câu hỏi 1: Pha nào trong tiến trình phát triển phần mềm tập trung vào việc thu thập yêu cầu từ khách hàng?**

**Đáp án: C. Pha lấy yêu cầu**

Giải thích: Pha lấy yêu cầu là giai đoạn đầu tiên và quan trọng nhất, nơi các nhà phân tích hệ thống thu thập và phân tích các yêu cầu của khách hàng để hiểu rõ nhu cầu và mong muốn của họ.

**Câu hỏi 2: Mô hình phát triển phần mềm nào có các vòng lặp liên tục và cải tiến sản phẩm sau mỗi lần lặp?**

**Đáp án: C. Mô hình Agile**

Giải thích: Mô hình Agile là một phương pháp phát triển lặp đi lặp lại và linh hoạt, tập trung vào việc cung cấp giá trị liên tục cho khách hàng thông qua các vòng lặp ngắn (sprint).

**Câu hỏi 3: Trong kỹ thuật lấy yêu cầu phần mềm, phương pháp nào giúp khai thác thông tin từ khách hàng bằng cách quan sát cách họ làm việc?**

**Đáp án: C. Quan sát**

Giải thích: Quan sát là một kỹ thuật hiệu quả để hiểu rõ quy trình làm việc thực tế của khách hàng và xác định các yêu cầu tiềm ẩn.

**Câu hỏi 4: Trong chuẩn hóa cơ sở dữ liệu, dạng chuẩn nào loại bỏ các phụ thuộc bắc cầu?**

**Đáp án: C. 3NF**

Giải thích: Dạng chuẩn 3NF loại bỏ các phụ thuộc bắc cầu, đảm bảo rằng mỗi thuộc tính không khóa chỉ phụ thuộc vào khóa chính.

**Câu hỏi 5: Trong UML, sơ đồ nào giúp mô tả sự tương tác giữa các đối tượng theo thời gian?**

**Đáp án: B. Sơ đồ tuần tự**

Giải thích: Sơ đồ tuần tự (Sequence Diagram) mô tả luồng tương tác giữa các đối tượng theo thứ tự thời gian, giúp hiểu rõ cách chúng tương tác với nhau.

**Câu hỏi 6: Trong UML, sơ đồ nào giúp mô tả sự tương tác giữa các đối tượng theo thời gian?**

* **Đáp án: B. Sơ đồ tuần tự**
* Giải thích: Câu hỏi này bị lặp lại với câu hỏi 5, và đáp án vẫn là sơ đồ tuần tự.

**Câu hỏi 7: CMMI mức 5 – Optimizing tập trung vào điều gì?**

* **Đáp án: C. Cải tiến quy trình**
* Giải thích: CMMI mức 5 tập trung vào việc liên tục cải tiến quy trình dựa trên dữ liệu và phản hồi để tối ưu hóa hiệu suất.

**Câu hỏi 8: Đâu là ưu điểm chính của mô hình phát triển phần mềm Agile?**

* **Đáp án: A. Linh hoạt và thay đổi nhanh chóng**
* Giải thích: Agile được thiết kế để thích ứng với các thay đổi yêu cầu và ưu tiên, cho phép nhóm phát triển phản ứng nhanh chóng với phản hồi của khách hàng.

**Câu hỏi 9: Nguyên tắc DRY (Don’t Repeat Yourself) trong lập trình có ý nghĩa gì?**

**Đáp án: A. Hạn chế viết code lặp lại**

Giải thích: Nguyên tắc DRY khuyến khích việc tái sử dụng code để giảm thiểu sự trùng lặp và cải thiện khả năng bảo trì.

**Câu hỏi 10: Yếu tố nào quan trọng nhất trong việc thiết kế một lớp trong lập trình hướng đối tượng?**

**Đáp án: B. Đảm bảo tính đóng gói và tái sử dụng**

Giải thích: Tính đóng gói và tái sử dụng là hai nguyên tắc cốt lõi của lập trình hướng đối tượng, giúp tạo ra các lớp linh hoạt và dễ bảo trì.

**Câu hỏi 11: Phát biểu nào đúng về kiểm thử phần mềm?**

**Đáp án: C. Kiểm thử có thể thực hiện song song với phát triển phần mềm**

Giải thích: Kiểm thử nên được thực hiện liên tục trong suốt quá trình phát triển để phát hiện và sửa lỗi sớm, giúp tiết kiệm thời gian và chi phí.

**Câu hỏi trả lời ngắn:**1. Định nghĩa phần mềm và công nghệ phần mềm.

* Phần mềm: Là tập hợp các chương trình máy tính, dữ liệu và tài liệu liên quan, được thiết kế để thực hiện các chức năng cụ thể trên hệ thống máy tính.
* Công nghệ phần mềm: Là một ngành kỹ thuật áp dụng các nguyên tắc, phương pháp và công cụ để thiết kế, phát triển, kiểm thử và bảo trì phần mềm một cách hiệu quả và có hệ thống.

2. Mô tả ngắn gọn về các mô hình vòng đời phát triển phần mềm phổ biến.

* Mô hình thác nước (Waterfall): Tuần tự, mỗi giai đoạn chỉ bắt đầu khi giai đoạn trước hoàn thành.
* Mô hình lặp và tăng trưởng (Iterative and Incremental): Phát triển phần mềm qua nhiều vòng lặp, mỗi vòng lặp xây dựng một phần chức năng và tích hợp vào sản phẩm cuối cùng.
* Mô hình xoắn ốc (Spiral): Kết hợp các yếu tố của mô hình thác nước và lặp, tập trung vào việc quản lý rủi ro.
* Mô hình Agile: Linh hoạt, lặp đi lặp lại, tập trung vào việc cung cấp giá trị liên tục cho khách hàng thông qua các vòng lặp ngắn (sprint).

3. Liệt kê ba loại yêu cầu phần mềm chính và giải thích từng loại.

* Yêu cầu chức năng (Functional Requirements): Mô tả các chức năng mà phần mềm phải thực hiện, ví dụ: "Người dùng có thể đăng nhập vào hệ thống".
* Yêu cầu phi chức năng (Non-functional Requirements): Mô tả các thuộc tính chất lượng của phần mềm, ví dụ: hiệu suất, bảo mật, khả năng mở rộng.
* Yêu cầu miền ứng dụng (Domain Requirements): Mô tả các yêu cầu đặc thù của lĩnh vực ứng dụng, ví dụ: các quy định pháp lý, các tiêu chuẩn ngành.

4. Mô tả vai trò của sơ đồ lớp UML trong thiết kế phần mềm.

* Sơ đồ lớp (Class Diagram) mô tả cấu trúc tĩnh của hệ thống, bao gồm các lớp, thuộc tính, phương thức và mối quan hệ giữa chúng.
* Nó giúp hiểu rõ hơn về thiết kế của hệ thống và là cơ sở cho việc lập trình.
* Sơ đồ lớp cũng được sử dụng để giao tiếp giữa các thành viên trong nhóm phát triển và với khách hàng.

5. Tại sao kiểm thử phần mềm quan trọng trong phát triển phần mềm?

* Phát hiện và sửa lỗi sớm, giúp tiết kiệm thời gian và chi phí.
* Đảm bảo chất lượng của phần mềm, tăng độ tin cậy và sự hài lòng của khách hàng.
* Cải thiện hiệu suất và bảo mật của phần mềm.
* Giúp phần mềm đáp ứng các yêu cầu của khách hàng.

6. Định nghĩa nguyên lý SOLID trong lập trình hướng đối tượng.

* S (Single Responsibility Principle): Một lớp chỉ nên có một lý do để thay đổi.
* O (Open/Closed Principle): Các thực thể phần mềm (lớp, module, hàm) nên mở rộng để bổ sung chức năng, nhưng đóng để sửa đổi.
* L (Liskov Substitution Principle): Các lớp con có thể thay thế cho lớp cha mà không làm thay đổi tính đúng đắn của chương trình.
* I (Interface Segregation Principle): Nhiều giao diện đặc thù tốt hơn là một giao diện tổng quát.
* D (Dependency Inversion Principle): Các module cấp cao không nên phụ thuộc vào các module cấp thấp. Cả hai nên phụ thuộc vào các trừu tượng.

7. Mô tả sự khác biệt giữa kiểm thử hộp đen và kiểm thử hộp trắng.

* Kiểm thử hộp đen (Black-box Testing): Kiểm thử dựa trên các yêu cầu và chức năng của phần mềm, không quan tâm đến cấu trúc bên trong của mã nguồn.
* Kiểm thử hộp trắng (White-box Testing): Kiểm thử dựa trên cấu trúc bên trong của mã nguồn, tập trung vào việc kiểm tra các đường dẫn thực thi và các điều kiện logic.

8. Mô tả quy trình thiết kế cơ sở dữ liệu từ sơ đồ lớp UML.

* Xác định các lớp thực thể trong sơ đồ lớp.
* Chuyển đổi các lớp thực thể thành các bảng trong cơ sở dữ liệu.
* Chuyển đổi các thuộc tính của lớp thành các cột trong bảng.
* Chuyển đổi các mối quan hệ giữa các lớp thành các ràng buộc trong cơ sở dữ liệu.
* Chuẩn hóa cơ sở dữ liệu để giảm thiểu sự dư thừa dữ liệu.

9. Nêu ba ưu điểm của việc sử dụng mô hình phát triển phần mềm Agile.

* Linh hoạt: Dễ dàng thích ứng với các thay đổi yêu cầu.
* Tương tác với khách hàng: Khách hàng tham gia vào quá trình phát triển, đảm bảo sản phẩm đáp ứng nhu cầu.
* Phản hồi nhanh chóng: Các vòng lặp ngắn giúp phát hiện và sửa lỗi sớm.

10. Liệt kê các giai đoạn chính trong quá trình chuẩn hóa cơ sở dữ liệu.

* 1NF (First Normal Form): Loại bỏ các nhóm lặp lại.
* 2NF (Second Normal Form): Loại bỏ các phụ thuộc không đầy đủ.
* 3NF (Third Normal Form): Loại bỏ các phụ thuộc bắc cầu.
* BCNF (Boyce-Codd Normal Form): Loại bỏ các phụ thuộc không khóa.
* 4NF (Fourth Normal Form): Loại bỏ các phụ thuộc đa giá trị.
* 5NF (Fifth Normal Form): Loại bỏ các phụ thuộc kết hợp.

**Câu hỏi thảo luận nhóm:**1. So sánh mô hình phát triển phần mềm Agile và mô hình Waterfall.

* Mô hình Waterfall:
  + Tuần tự, mỗi giai đoạn chỉ bắt đầu khi giai đoạn trước hoàn thành.
  + Yêu cầu cố định từ đầu.
  + Khó thay đổi khi dự án đang tiến hành.
  + Phù hợp với các dự án có yêu cầu rõ ràng, ổn định.
* Mô hình Agile:
  + Lặp đi lặp lại, linh hoạt.
  + Cho phép thay đổi yêu cầu trong quá trình phát triển.
  + Tập trung vào việc tương tác với khách hàng và phản hồi nhanh chóng.
  + Phù hợp với các dự án có yêu cầu thay đổi thường xuyên, không chắc chắn.

2. Lợi ích của việc sử dụng UML trong thiết kế phần mềm là gì?

* Giúp trực quan hóa thiết kế hệ thống.
* Tạo ra một ngôn ngữ chung để giao tiếp giữa các thành viên trong nhóm.
* Giúp phát hiện sớm các lỗi thiết kế.
* Tạo ra tài liệu thiết kế chi tiết, dễ hiểu.

3. Làm thế nào để đảm bảo phần mềm có thể bảo trì tốt trong tương lai?

* Viết mã nguồn rõ ràng, dễ hiểu, tuân thủ các quy chuẩn lập trình.
* Sử dụng các kỹ thuật thiết kế hướng đối tượng tốt (SOLID).
* Viết tài liệu đầy đủ, chi tiết.
* Thực hiện kiểm thử kỹ lưỡng.
* Sử dụng các công cụ quản lý mã nguồn.

4. Tại sao các công ty phần mềm thường sử dụng mô hình phát triển lặp (Iterative Development)?

* Giúp giảm thiểu rủi ro.
* Cho phép phản hồi nhanh chóng từ khách hàng.
* Tăng tính linh hoạt của dự án.
* Giúp phát hiện sớm các lỗi.

5. Vai trò của kiến trúc phần mềm trong việc xây dựng một hệ thống phần mềm phức tạp.

* Xác định cấu trúc tổng thể của hệ thống.
* Đảm bảo tính nhất quán và khả năng mở rộng của hệ thống.
* Giúp quản lý sự phức tạp của hệ thống.
* Tạo ra một nền tảng vững chắc cho việc phát triển và bảo trì phần mềm.

6. Làm thế nào để đảm bảo rằng một hệ thống phần mềm đáp ứng được yêu cầu bảo mật?

* Thực hiện đánh giá rủi ro bảo mật.
* Áp dụng các biện pháp bảo mật phù hợp (mã hóa dữ liệu, kiểm soát truy cập, v.v.).
* Thực hiện kiểm thử bảo mật.
* Tuân thủ các tiêu chuẩn bảo mật.
* Cập nhật phần mềm thường xuyên để vá các lỗ hổng bảo mật.

7. So sánh kiểm thử đơn vị (Unit Testing) và kiểm thử tích hợp (Integration Testing).

* Kiểm thử đơn vị (Unit Testing):
  + Kiểm tra từng đơn vị mã nguồn riêng lẻ.
  + Mục đích: Phát hiện lỗi trong mã nguồn.
  + Thực hiện bởi lập trình viên.
* Kiểm thử tích hợp (Integration Testing):
  + Kiểm tra sự tương tác giữa các thành phần phần mềm.
  + Mục đích: Phát hiện lỗi trong giao tiếp giữa các thành phần.
  + Thực hiện bởi nhóm kiểm thử.

8. Những thách thức chính trong việc thu thập yêu cầu phần mềm là gì?

* Khách hàng không biết rõ mình muốn gì.
* Khách hàng thay đổi yêu cầu liên tục.
* Khó khăn trong việc giao tiếp với khách hàng.
* Yêu cầu không rõ ràng, mơ hồ.
* Khó khăn trong việc xác định các yêu cầu phi chức năng.

9. Cách áp dụng quy trình Scrum vào dự án phát triển phần mềm thực tế.

* Chia dự án thành các sprint (vòng lặp ngắn).
* Xác định vai trò (Product Owner, Scrum Master, Development Team).
* Tổ chức các cuộc họp Scrum (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective).
* Sử dụng bảng Scrum để theo dõi tiến độ.
* Tập trung vào việc cung cấp giá trị liên tục cho khách hàng.

10. Tại sao việc tối ưu hóa mã nguồn lại quan trọng trong phát triển phần mềm?

* Tăng hiệu suất của phần mềm.
* Giảm thiểu việc sử dụng tài nguyên hệ thống.
* Giúp phần mềm chạy nhanh hơn.
* Cải thiện trải nghiệm người dùng.
* Giúp dễ dàng bảo trì và nâng cấp phần mềm.

**Câu hỏi tình huống:**

1. Bạn là một kỹ sư phần mềm trong một nhóm phát triển, khách hàng liên tục thay đổi yêu cầu. Bạn sẽ xử lý như thế nào?

* Ghi nhận và đánh giá: Ghi nhận tất cả các thay đổi yêu cầu và đánh giá tác động của chúng đến thời gian, chi phí và nguồn lực của dự án.
* Thảo luận với khách hàng: Thảo luận với khách hàng về lý do thay đổi và các chi phí phát sinh.
* Ưu tiên thay đổi: Ưu tiên các thay đổi quan trọng nhất và ảnh hưởng lớn đến giá trị sản phẩm.
* Cập nhật tài liệu: Cập nhật tài liệu yêu cầu và thiết kế để phản ánh các thay đổi.
* Sử dụng quy trình linh hoạt: Áp dụng các quy trình phát triển linh hoạt như Agile để dễ dàng thích ứng với các thay đổi.

2. Trong quá trình kiểm thử, bạn phát hiện một lỗi nghiêm trọng nhưng trưởng nhóm quyết định không sửa chữa. Bạn sẽ làm gì?

* Ghi nhận và báo cáo: Ghi nhận chi tiết lỗi và báo cáo lại cho trưởng nhóm, nhấn mạnh mức độ nghiêm trọng và ảnh hưởng của nó.
* Thảo luận và tìm hiểu: Thảo luận với trưởng nhóm để hiểu rõ lý do không sửa chữa.
* Đề xuất giải pháp: Đề xuất các giải pháp khả thi để khắc phục lỗi.
* Báo cáo lên cấp cao hơn: Nếu trưởng nhóm vẫn quyết định không sửa chữa và bạn tin rằng lỗi có thể gây hậu quả nghiêm trọng, hãy báo cáo lên cấp quản lý cao hơn.

3. Một dự án phần mềm gặp tình trạng chậm tiến độ do thay đổi yêu cầu liên tục từ khách hàng. Bạn sẽ đề xuất giải pháp gì?

* Thống nhất yêu cầu: Thống nhất lại các yêu cầu với khách hàng và ưu tiên các yêu cầu quan trọng nhất.
* Sử dụng quy trình Agile: Áp dụng các phương pháp Agile để quản lý thay đổi linh hoạt hơn.
* Tăng cường giao tiếp: Tăng cường giao tiếp với khách hàng để cập nhật tiến độ và quản lý kỳ vọng.
* Đánh giá lại tiến độ: Đánh giá lại tiến độ dự án và điều chỉnh kế hoạch nếu cần thiết.
* Tăng cường nguồn lực: Nếu cần thiết, đề xuất tăng cường nguồn lực cho dự án.

4. Nhóm của bạn đang thiết kế cơ sở dữ liệu cho một hệ thống thương mại điện tử. Làm thế nào để đảm bảo thiết kế cơ sở dữ liệu không bị dư thừa?

* Chuẩn hóa dữ liệu: Áp dụng các quy tắc chuẩn hóa dữ liệu (1NF, 2NF, 3NF, BCNF) để loại bỏ sự dư thừa.
* Phân tích quan hệ: Phân tích kỹ lưỡng các quan hệ giữa các thực thể để đảm bảo không có sự trùng lặp dữ liệu.
* Sử dụng khóa ngoại: Sử dụng khóa ngoại để liên kết các bảng và tránh dư thừa dữ liệu.
* Thiết kế mô hình ERD: Thiết kế mô hình ERD (Entity-Relationship Diagram) để trực quan hóa cấu trúc cơ sở dữ liệu.
* Kiểm tra và tối ưu hóa: Kiểm tra và tối ưu hóa thiết kế cơ sở dữ liệu để đảm bảo hiệu suất và tính toàn vẹn.

5. Bạn cần lựa chọn giữa hai mô hình phát triển phần mềm: Waterfall và Agile. Bạn sẽ chọn mô hình nào cho một dự án startup công nghệ? Tại sao?

* Chọn mô hình Agile:
  + Lý do:
    - Linh hoạt: Dự án startup thường có yêu cầu thay đổi liên tục, Agile cho phép thích ứng nhanh chóng.
    - Phản hồi nhanh: Agile cho phép nhận phản hồi từ khách hàng sớm và thường xuyên, giúp điều chỉnh sản phẩm phù hợp.
    - Tối ưu hóa giá trị: Agile tập trung vào việc cung cấp giá trị cho khách hàng trong từng giai đoạn phát triển.

6. Một dự án phần mềm lớn gặp vấn đề về hiệu suất. Những bước nào cần thực hiện để tối ưu hiệu suất phần mềm?

* Phân tích hiệu suất: Sử dụng các công cụ phân tích hiệu suất để xác định các điểm nghẽn.
* Tối ưu hóa mã nguồn: Tối ưu hóa các thuật toán, cấu trúc dữ liệu và logic xử lý.
* Tối ưu hóa cơ sở dữ liệu: Tối ưu hóa truy vấn, chỉ mục và cấu trúc bảng.
* Tối ưu hóa tài nguyên: Tối ưu hóa việc sử dụng bộ nhớ, CPU và các tài nguyên khác.
* Kiểm thử hiệu suất: Kiểm thử hiệu suất sau khi tối ưu hóa để đảm bảo cải thiện.

7. Một hệ thống đang hoạt động có nhiều lỗi bảo mật. Bạn sẽ làm gì để tăng cường bảo mật mà không ảnh hưởng đến người dùng hiện tại?

* Đánh giá rủi ro: Đánh giá các rủi ro bảo mật và ưu tiên các lỗ hổng nghiêm trọng nhất.
* Cập nhật phần mềm: Cập nhật phần mềm lên phiên bản mới nhất để vá các lỗ hổng đã biết.
* Triển khai tường lửa: Triển khai tường lửa và các biện pháp bảo mật mạng.
* Mã hóa dữ liệu: Mã hóa dữ liệu nhạy cảm.
* Kiểm soát truy cập: Áp dụng cơ chế kiểm soát truy cập chặt chẽ.
* Kiểm thử bảo mật: Kiểm thử bảo mật định kỳ để phát hiện và khắc phục các lỗ hổng mới.

8. Khi thiết kế phần mềm, nhóm của bạn có nhiều ý kiến trái ngược nhau về cách triển khai một tính năng. Làm thế nào để đưa ra quyết định tốt nhất?

* Thảo luận và phân tích: Tổ chức cuộc họp để các thành viên trình bày và phân tích các ý kiến.
* Đánh giá ưu nhược điểm: Đánh giá ưu nhược điểm của từng phương án dựa trên các tiêu chí cụ thể.
* Bỏ phiếu hoặc đồng thuận: Sử dụng phương pháp bỏ phiếu hoặc đồng thuận để đưa ra quyết định cuối cùng.
* Ghi nhận quyết định: Ghi nhận lại quyết định và lý do lựa chọn.
* Thử nghiệm và đánh giá: Nếu có thể, thử nghiệm các phương án khác nhau và đánh giá kết quả.

9. Khách hàng yêu cầu hệ thống phải có giao diện thân thiện với người dùng. Bạn sẽ làm gì để đảm bảo hệ thống đáp ứng tiêu chí này?

* Nghiên cứu người dùng: Nghiên cứu và hiểu rõ nhu cầu và hành vi của người dùng.
* Thiết kế giao diện trực quan: Thiết kế giao diện trực quan, dễ sử dụng và nhất quán.
* Sử dụng các nguyên tắc UX/UI: Áp dụng các nguyên tắc thiết kế UX/UI (User Experience/User Interface) tốt.
* Kiểm thử người dùng: Tổ chức kiểm thử người dùng để thu thập phản hồi và cải thiện giao diện.
* Lặp lại và cải tiến: Lặp lại quá trình thiết kế và kiểm thử để liên tục cải thiện giao diện.

10. Công ty bạn vừa tiếp nhận một dự án phần mềm cũ, nhưng không có tài liệu hướng dẫn. Bạn sẽ làm gì để hiểu và tiếp tục phát triển hệ thống này?

* Phân tích mã nguồn: Phân tích mã nguồn để hiểu cấu trúc và logic của hệ thống.
* Chạy thử và gỡ lỗi: Chạy thử hệ thống và gỡ lỗi để hiểu rõ cách hoạt động.
* Tạo tài liệu: Tạo tài liệu hướng dẫn dựa trên phân tích mã nguồn và chạy thử.
* Phỏng vấn người phát triển cũ: Nếu có thể, phỏng vấn người phát triển cũ để thu thập thông tin.
* Sử dụng công cụ phân tích mã nguồn: Sử dụng các công cụ phân tích mã nguồn để tự động tạo tài liệu và sơ đồ.