


20242 - IT4082 - KTPM - (Bài tổng hợp c6,7)


Điểm: 56/56

1

Họ và tên sinh viên: * 

Phạm Đức Long


2

Mã số sinh viên * 

20225737

✓ **Đúng** 1/1 Điểm

3

Lựa chọn nào sau đây là phù hợp để đặt tên cho Usecase khi thiết kế sơ đồ usecase cho hệ thống cửa hàng sách trực tuyến? 

☐ Enter book title


☐ Enter credit card details

☐ Do not order a book

☒ Cancel order

✓ **Đúng** 1/1 Điểm

4


Sơ đồ Usecase sau đây có mô hình hóa phù hợp cho tình huống: "Để tiến hành một kỳ thi, cần có một học sinh và ít nhất một giáo viên" không? 

☐ Đúng

☒ Sai

✓ **Đúng** 1/1 Điểm

5

Phương án nào phù hợp để mô hình hóa tình huống sau đây bằng sơ đồ Usecase:
"Một sinh viên chỉnh sửa hồ sơ người dùng của mình. Trong quá trình đó, sinh viên cũng có thể thay đổi mật khẩu của mình nếu muốn." 

☐ A

☐ B

☒ C

☐ D

✓ **Đúng** 1/1 Điểm

6

Quan hệ nào sau đây thể hiện **mức độ phụ thuộc thấp nhất** giữa hai lớp trong thiết kế hướng đối tượng? 

☐ Kế thừa (Inheritance)

☐ Hợp thành (Composition)

- ☐ Kết tập (Aggregation)
- ☒ Phụ thuộc qua interface (Dependency via Interface)

✓ **Đúng** 1/1 Điểm


7

Trong thiết kế phần mềm, nguyên tắc **“Separation of concerns”** đề cập đến điều gì? 

- ☐ Phân chia dữ liệu và chức năng thành các khối độc lập
- ☒ Tách biệt các vấn đề phức tạp thành các phần nhỏ để quản lý hơn
- ☐ Phân loại các lỗi thành từng nhóm để xử lý riêng biệt
- ☐ Kết hợp nhiều thành phần lại để tăng tính hiệu quả


✓ **Đúng** 1/1 Điểm

8

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, PaymentProcessor phụ thuộc trực tiếp vào PaymentGateway. Để giảm mức độ phụ thuộc, thiết kế nào sau đây là tốt nhất? 

- ☐ Kế thừa trực tiếp từ PaymentGateway
- ☐ Gộp PaymentProcessor và PaymentGateway thành một lớp
- ☒ Thay PaymentGateway bằng một interface trung gian
- ☐ Sử dụng biến tĩnh để truy cập gateway


✓ **Đúng** 1/1 Điểm

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Phương thức `encryptData(data: string)` trong `PaymentProcessor` không liên quan đến logic xử lý thanh toán, điều này làm giảm chất lượng thiết kế vì: 

- ☐ Gây ra sự trùng lặp mã nguồn
- ☐ Làm tăng mức độ ghép nối giữa các lớp
- ☐ Phá vỡ nguyên lý Interface Segregation
- ☒ Vi phạm nguyên lý Single Responsibility

✓ **Đúng** 1/1 Điểm

10

Phát biểu "Có hai cách thực hiện giao hàng là giao tại cửa hàng hoặc giao hàng tận nhà". Phương án nào trong các sơ đồ sau thể hiện phù hợp cho phát biểu trên? 

- ☒ A
- ☐ Cả 3 biểu đồ đều phù hợp
- ☐ C
- ☐ B

✓ **Đúng** 1/1 Điểm


11

Nguyên tắc nào không phù hợp với việc thiết kế phần mềm chất lượng? 

- ☐ Thiết kế phải giảm thiểu khoảng cách trí tuệ giữa bài toán thực tế và giải pháp phần mềm
- ☒ Thiết kế cần ưu tiên tạo ra một hệ thống hoàn toàn phụ thuộc vào môi trường triển khai


- ☐ Thiết kế phải hỗ trợ dễ dàng tích hợp với hệ thống bên ngoài
- ☐ Thiết kế phải phản ánh tính đồng nhất và khả năng thích nghi

12

Xét tình huống sau: "Một người đi ăn trưa. Trong quá trình đó, người này có thể cần phải rút tiền mặt từ máy ATM". Một nhà phân tích đã mô hình hoá tình huống bằng một biểu đồ Usecase, hãy cho biết biểu đồ này có phù hợp cho mô tả tình huống trên không, giải thích? 

Biểu đồ không phù hợp vì dùng quan hệ include khiến hành động rút tiền luôn xảy ra khi đi ăn trưa, trong khi tình huống mô tả chỉ là "có thể" rút tiền, nên nên dùng extend mới đúng.


13

Một lỗi thường gặp là quá nhiều usecase được đưa vào mô hình yêu cầu với những usecase quá nhỏ (những usecase quá đơn giản, chỉ tương đương với 1 hành động hoặc giá trị kết quả usecase không cụ thể). Em hãy nêu ví dụ về trường hợp không nên đưa vào mô hình usecase. 

"Nhập tên đăng nhập" hoặc "Nhấn nút gửi" trong quá trình đăng nhập.

✓ **Đúng** 1/1 Điểm

14

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Nếu OrderProcessor phải kiểm tra điều kiện logic phức tạp trước khi gọi ShippingService.scheduleDelivery(), kỹ thuật thiết kế nào có thể áp dụng để **giảm độ phức tạp hàm processOrder()**? 

- ☐ Dùng một lớp singleton quản lý luồng xử lý
- ☐ Gộp ShippingService và OrderProcessor

- ☐ Chuyển hết logic đó vào ShippingService
- ☒ Refactor thành lớp xử lý logic nghiệp vụ trung gian

✓ **Đúng** 1/1 Điểm

15

Yếu tố nào không thuộc nguyên tắc thiết kế chất lượng? 

- ☐ Giao diện giảm sự phức tạp của kết nối
- ☐ Dữ liệu và kiến trúc được thể hiện rõ ràng
- ☐ Thiết kế hỗ trợ việc kiểm thử dễ dàng hơn
- ☒ Các thành phần chức năng có độ phụ thuộc cao

✓ **Đúng** 1/1 Điểm

16

Nguyên tắc nào không thuộc nguyên tắc thiết kế? 

- ☐ Thiết kế phải kiểm chứng được bằng thực nghiệm
- ☒ Thiết kế không cần xét đến dữ liệu bất thường
- ☐ Thiết kế nên biểu lộ tính đồng nhất và tích hợp
- ☐ Thiết kế nên được cấu trúc để thích ứng với thay đổi

✓ **Đúng** 1/1 Điểm

17

Phong cách kiến trúc nào được mô tả bởi các tầng xử lý được sắp xếp theo cấp bậc? 

- ☒ Kiến trúc phân lớp
- ☐ Kiến trúc lấy dữ liệu làm trung tâm
- ☐ Kiến trúc gọi và trả về
- ☐ Kiến trúc luồng dữ liệu

✓ **Đúng** 1/1 Điểm

18

Hoàn chỉnh phát biểu sau: là công cụ tốt để mô tả luồng hoạt động trong một Use Case và thường được dùng trong mô hình nghiệp vụ. ☐

- ☐ Sơ đồ triển khai (Deployment Diagrams)
- ☐ Sơ đồ lớp (Class Diagrams)
- ☐ Sơ đồ trạng thái (State Diagrams)
- ☒ Sơ đồ hoạt động (Activity Diagrams)

✓ **Đúng** 1/1 Điểm

19


Một công ty phát triển hệ thống quản lý thư viện với các module độc lập: quản lý sách, quản lý độc giả, quản lý mượn trả và báo cáo thống kê. Các module này đều cần truy cập vào cùng một nguồn dữ liệu. Kiểu kiến trúc nào phù hợp nhất cho hệ thống này? ☐

- ☐ Kiến trúc phân lớp
- ☐ Kiến trúc gọi và trả về
- ☐ Kiến trúc luồng dữ liệu

☒ Kiến trúc lấy dữ liệu làm trung tâm

✓ **Đúng** 1/1 Điểm

20

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Tại sao nên giữ các phương thức như `processPayment`, `validatePayment`, `getPaymentStatus` trong một interface thay vì lớp cụ thể? 

- ☐ Cho phép khai báo biến toàn cục dễ hơn
- ☐ Giảm rủi ro runtime error
- ☐ Giảm chi phí CPU khi xử lý đa tiến trình
- ☒ Đảm bảo tính nhất quán giữa các implementation

✓ **Đúng** 1/1 Điểm

21


Sự thừa kế giữa hai Use cases còn được biết đến như mối quan hệ nào sau đây? 

- ☐ Refinement
- ☐ Polymorphism
- ☒ Generalization
- ☐ Association

✓ **Đúng** 1/1 Điểm

22


Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Nếu hệ thống chuyển sang hỗ trợ nhiều loại `ShippingService` (ví dụ: Giao hàng nhanh, giao hàng quốc tế), thiết

kế hiện tại cần thay đổi theo hướng nào để tuân thủ **Open-Closed Principle**? 

- ☐ Tạo subclass cho ShippingService và kiểm tra bằng câu lệnh if
- ☐ Dùng các phương thức static trong lớp ShippingService
- ☐ Thêm tất cả logic vào OrderProcessor
- ☒ Sử dụng Factory Pattern để chọn service tương ứng

✓ **Đúng** 1/1 Điểm


23

Kỹ nghệ yêu cầu phần mềm gồm bước nào nhằm xác định các nhân sự có hiểu biết sâu sắc nhất về hệ thống? 

- ☒ Khám phá (Elicitation)
- ☐ Xây dựng (Elaboration)
- ☐ Khởi đầu (Inception)
- ☐ Đàm phán (Negotiation)

✓ **Đúng** 1/1 Điểm

24


Một ứng dụng xử lý ảnh thực hiện các bước tuần tự: đọc ảnh đầu vào, lọc màu RGB, điều chỉnh độ tương phản, và áp dụng hiệu ứng để tạo ảnh đầu ra. Kiểu kiến trúc nào phù hợp nhất cho ứng dụng này? 

- ☐ Kiến trúc lấy dữ liệu làm trung tâm
- ☒ Kiến trúc luồng dữ liệu
- ☐ Kiến trúc phân lớp

☐ Kiến trúc gọi và trả về

✓ **Đúng** 1/1 Điểm

25

Trong biểu đồ Usecase, mối quan hệ _____ được dùng để chỉ hành vi tùy chọn mà chỉ chạy dưới những điều kiện nhất định. 

☒ Mở rộng (extend)


☐ Bao gồm (include)

☐ Kết hợp (association)

☐ Thừa kế (inheritance)

✓ **Đúng** 1/1 Điểm

26

Phong cách kiến trúc nào nhấn mạnh vào khả năng tái sử dụng các thành phần qua giao diện? 

☒ Kiến trúc hướng đối tượng


☐ Kiến trúc gọi và trả về

☐ Kiến trúc phân lớp

☐ Kiến trúc luồng dữ liệu

✓ **Đúng** 1/1 Điểm


27

Một thiết kế tốt cần đảm bảo yếu tố nào sau đây? 

- ☒ Dễ đọc, dễ hiểu và giảm thiểu lỗi khái niệm
- ☐ Phụ thuộc vào ý tưởng ban đầu của người thiết kế
- ☐ Không yêu cầu mô hình hóa để biểu diễn chi tiết
- ☐ Bao gồm tất cả chi tiết cài đặt của phần mềm

✓ **Đúng** 1/1 Điểm

28

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Interface `IPaymentProcessor` chứa phương thức `getPaymentStatus(id: string)`. Việc này phản ánh yêu cầu gì trong thiết kế interface? 

- ☐ Loại bỏ các hành vi dư thừa trong `PaymentProcessor`
- ☒ Phân tách chức năng xử lý và truy vấn trạng thái
- ☐ Tăng độ liên kết giữa client và service
- ☐ Gộp chung mọi hành vi liên quan vào một interface

✓ **Đúng** 1/1 Điểm


29

Mục đích của sơ đồ hoạt động là? 

- ☐ Giúp nắm được mục đích cơ bản của lớp, tốt cho việc khám phá việc cài đặt use case như thế nào
- ☐ Biểu diễn cách bố trí các thành phần trên các nút phần cứng
- ☐ Biểu diễn cấu trúc tĩnh của các khái niệm, các loại và các lớp
- ☒ Biểu diễn hành vi với cấu trúc điều khiển. Sơ đồ hoạt động có thể biểu diễn nhiều đối tượng trong một use case

✓ **Đúng** 1/1 Điểm


30

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Giao diện IPaymentProcessor trong thiết kế nhằm mục tiêu gì? 

- ☐ Loại bỏ sự cần thiết của dependency injection
- ☐ Cho phép kế thừa từ nhiều lớp
- ☒ Cho phép đa hình và thay thế các implementation cụ thể
- ☐ Áp dụng nguyên lý "đóng gói"

✓ **Đúng** 1/1 Điểm


31

Hoàn chỉnh phát biểu sau về mối quan hệ giữa các Usecases: Sử dụng khi chúng ta muốn giảm các bước trùng lặp giữa các usecase, lấy những bước chung đó để tạo nên usecase phụ mà luôn được gọi thực thi trong usecase cơ sở. 

- ☐ Extend
- ☐ Generalization
- ☒ Include
- ☐ Delegation

✓ **Đúng** 1/1 Điểm


32

Đặc điểm nào không đúng với hồ sơ đặc tả phi hình thức? 

- ☐ Viết bằng ngôn ngữ tự nhiên
- ☐ Không yêu cầu biểu diễn toán học phức tạp
- ☐ Thích hợp cho việc mô tả tổng quát yêu cầu
- ☒ Dựa trên ký pháp đồ họa với cú pháp chặt chẽ

✓ **Đúng** 1/1 Điểm


33

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Sự phụ thuộc trực tiếp của PaymentProcessor vào lớp cụ thể PaymentGateway có thể gây ra hệ quả nào nếu hệ thống mở rộng thêm nhiều gateway khác? 

- ☐ Làm tăng khả năng mở rộng hệ thống
- ☐ Không ảnh hưởng gì vì đã có đa hình
- ☒ Làm giảm khả năng kiểm thử độc lập
- ☐ Làm tăng tính ổn định khi thay đổi

✓ **Đúng** 1/1 Điểm

34

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Lý do chính để tách IPaymentProcessor thành một interface thay vì để các phương thức trực tiếp trong lớp OrderProcessor là gì? 

- ☐ Tăng khả năng truy cập các thuộc tính private của OrderProcessor
- ☐ Giảm số lượng phương thức trong OrderProcessor
- ☐ Dễ dàng triển khai đa luồng trong OrderProcessor

☒ Áp dụng nguyên lý phân tách giao diện và giảm phụ thuộc vào chi tiết

✓ **Đúng** 1/1 Điểm


35

Biểu đồ nào dùng để biểu diễn cách thức dữ liệu trung chuyển giữa các tiến trình? 

- ☒ Biểu đồ luồng dữ liệu (DFD)
- ☐ Sơ đồ lớp (Class Diagram)
- ☐ Máy trạng thái hữu hạn (FSM)
- ☐ Sơ đồ thực thể liên kết (ERD)

✓ **Đúng** 1/1 Điểm


36

Loại kiến trúc nào tập trung vào các thành phần phối hợp và truyền thông dữ liệu giữa chúng? 

- ☐ Kiến trúc hướng đối tượng
- ☐ Kiến trúc phân lớp
- ☐ Kiến trúc gọi và trả về
- ☒ Kiến trúc luồng dữ liệu

✓ **Đúng** 1/1 Điểm

37


Phát biểu sau đây là đúng hay sai: "Một tác nhân (Actor) trong một Usecase luôn là một con người" 

☐ Đúng

☒ Sai

✓ **Đúng** 1/1 Điểm

38

Trong biểu đồ Usecase, mối quan hệ <<include>> mang ý nghĩa nào? 

☒ A không thể được thực hiện mà không có B


☐ Hành vi của B có thể hoặc không thể được chèn vào hành vi của A

☐ B có thể được thực hiện thay vì A

☐ Hành vi của A luôn phải được chèn vào hành vi của B

✓ **Đúng** 1/1 Điểm

39

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Giao diện IPaymentProcessor nên tuân thủ nguyên lý nào để tránh quá chồng kênh và khó mở rộng? 

☒ Interface Segregation Principle

☐ Least Privilege Principle


☐ Law of Demeter

☐ Liskov Substitution Principle

✓ **Đúng** 1/1 Điểm

Một công ty thương mại điện tử toàn cầu đang xây dựng hệ thống với các yêu cầu sau:

- Hệ thống cần phục vụ khách hàng ở nhiều khu vực địa lý khác nhau với độ trễ thấp
- Mỗi trung tâm dữ liệu cần có thể hoạt động độc lập khi mất kết nối
- Dữ liệu cần được đồng bộ giữa các trung tâm khi có kết nối
- Hệ thống cần đảm bảo tính nhất quán cuối cùng của dữ liệu
- Cần hỗ trợ khả năng mở rộng theo chiều ngang


Trong tình huống này, đâu là tổ hợp kiến trúc phù hợp nhất? 

- ☐ Kiến trúc gọi và trả về với cơ chế cache phân tán
- ☒ Kiến trúc kết hợp giữa gọi và trả về, phân tán với CSDL phi tập trung và đồng bộ không đồng thời
- ☐ Kiến trúc phân lớp đơn thuần với CSDL tập trung
- ☐ Kiến trúc lấy dữ liệu làm trung tâm với một CSDL chủ

✓ **Đúng** 1/1 Điểm

Một công ty fintech đang xây dựng hệ thống xử lý giao dịch tài chính với các yêu cầu sau:


- Cần xử lý đồng thời nhiều luồng dữ liệu từ các nguồn khác nhau (giao dịch ngân hàng, ví điện tử, crypto)
- Dữ liệu sau xử lý cần được lưu trữ tập trung để phân tích
- Các module phân tích hoạt động độc lập và không ảnh hưởng đến luồng xử lý chính
- Kết quả cuối cùng cần được tổng hợp từ nhiều luồng xử lý

Kiến trúc nào KHÔNG phù hợp cho hệ thống này? 

- ☒ Kiến trúc phân lớp truyền thống
- ☐ Kiến trúc hướng sự kiện với bộ đệm message
- ☐ Kiến trúc microservices với CSDL chia sẻ
- ☐ Kiến trúc kết hợp giữa luồng dữ liệu và lấy dữ liệu làm trung tâm

✓ **Đúng** 1/1 Điểm


42

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, nếu cần kiểm thử đơn vị (unit test) cho OrderProcessor, điều gì là bắt buộc để đảm bảo tính dễ kiểm thử? 

- ☒ Dùng các mock object cho các service phụ thuộc
- ☐ Tách riêng phần xử lý dữ liệu ra khỏi hàm gọi
- ☐ Gọi trực tiếp API thật để kiểm tra tính đúng đắn
- ☐ Không sử dụng interface mà dùng trực tiếp class

✓ **Đúng** 1/1 Điểm

43

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Sự hiện diện của interface IPaymentProcessor làm tăng khả năng nào sau đây? 

- ☒ Khả năng mở rộng và kiểm thử độc lập
- ☐ Tự động tạo mã nguồn cho lớp giao diện
- ☐ Khả năng xử lý đồng thời các đơn hàng
- ☐ Hiệu suất thực thi của hệ thống

✓ **Đúng** 1/1 Điểm

44

Một thiết kế phần mềm có tính modular cao cần đảm bảo yếu tố nào? 

- ☒ Các module chứa đầy đủ dữ liệu và chức năng cần thiết
- ☐ Các module dễ dàng mở rộng nhưng khó bảo trì
- ☐ Các module phụ thuộc lẫn nhau để hoạt động tốt hơn
- ☐ Tích hợp các module vào hệ thống dễ dàng

✓ **Đúng** 1/1 Điểm


45

Thành phần nào không phải là đặc điểm của kiến trúc phần mềm? 

- ☒ Mô tả chi tiết mã nguồn của từng thành phần
- ☐ Cách tích hợp các thành phần thành hệ thống
- ☐ Tập hợp các kết nối giữa các thành phần
- ☐ Tập hợp các thành phần thực hiện chức năng cần thiết

✓ **Đúng** 1/1 Điểm

46


Trong một hệ thống thanh toán trực tuyến, một thành phần xử lý thanh toán được thiết kế với các lớp PaymentProcessor và PaymentGateway. Dựa vào thiết kế chi tiết thành phần, đâu là nhận xét KHÔNG chính xác? 

- ☒ PaymentProcessor và PaymentGateway có coupling chặt chẽ và không thể tách rời

- ☐ PaymentGateway là một dependency của PaymentProcessor, thể hiện quan hệ has-a
- ☐ Interface IPaymentProcessor giúp giảm sự phụ thuộc và tăng khả năng thay thế implementation
- ☐ Thiết kế tuân thủ nguyên tắc đóng gói bằng cách ẩn chi tiết triển khai của phương thức encryptData

✓ **Đúng** 1/1 Điểm


47

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Giải pháp nào giúp đảm bảo tính mở rộng khi thêm loại PaymentGateway mới (ví dụ: PayPal, Stripe) mà không ảnh hưởng đến PaymentProcessor? 

- ☐ Sử dụng biến tĩnh để chọn PaymentGateway
- ☐ Sử dụng kế thừa từ PaymentGateway
- ☒ Sử dụng Dependency Injection và một factory để khởi tạo
- ☐ Gộp tất cả các loại gateway vào một lớp lớn

✓ **Đúng** 1/1 Điểm

48

Trong sơ đồ Usecase sau, tác nhân T giao tiếp với các trường hợp sử dụng nào - bất kể các tác nhân khác có tham gia hay không? 

- ☐ Chỉ có D
- ☐ D và E
- ☐ D và F
- ☒ E và F

✓ **Đúng** 1/1 Điểm


49

Quan hệ nào sau đây **làm tăng cohesion nhưng giảm coupling** trong thiết kế hướng đối tượng? 

- ☒ Tạo interface cho các service và inject qua constructor
- ☐ Kế thừa từ lớp cha có nhiều chức năng phụ trợ
- ☐ Gộp nhiều lớp vào một module xử lý chính
- ☐ Truy cập trực tiếp biến toàn cục chứa các dependency

✓ **Đúng** 1/1 Điểm


50

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Mối quan hệ giữa PaymentProcessor và IPaymentProcessor là mối quan hệ gì trong thiết kế hướng đối tượng? 

- ☐ Giao tiếp (Interaction)
- ☐ Kế thừa đa hình
- ☐ Hợp thành (Composition)
- ☒ Cài đặt (Implementation)

✓ **Đúng** 1/1 Điểm


51

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, nếu ShippingService có thêm phương thức trackDelivery(orderId: string), điều gì cần xem xét để không làm tăng tính **phụ thuộc không cần thiết**? 

- ☐ Đặt logic tracking vào OrderProcessor để tiện truy cập
- ☐ Chỉ sử dụng khi cần thiết, không inject vào constructor
- ☐ Đảm bảo OrderProcessor gọi trực tiếp phương thức đó
- ☒ Tách thành một lớp DeliveryTracker riêng biệt nếu phức tạp

✓ **Đúng** 1/1 Điểm


52

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Việc OrderProcessor gọi các phương thức từ InventoryService, PaymentProcessor và ShippingService là ví dụ của kiểu thiết kế nào? 

- ☐ Service Oriented Architecture
- ☐ Template Method
- ☐ Mediator Pattern
- ☒ Orchestrator trong thiết kế dịch vụ

✓ **Đúng** 1/1 Điểm

53


Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, nếu ta cần ghi log chi tiết cho từng lần xử lý thanh toán, nên thực hiện ở đâu để **không vi phạm nguyên lý SRP**? 

- ☐ Trong PaymentGateway
- ☐ Trong OrderProcessor
- ☒ Trong Logger được injected vào PaymentProcessor

☐ Trong processOrder() gọi System.out.println()

✓ **Đúng** 1/1 Điểm

54

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Nếu cần đo hiệu suất cho toàn bộ chu trình đặt hàng trong OrderProcessor, cách tiếp cận nào **tốt nhất** mà không can thiệp vào lớp gốc? 

☒ Dùng decorator hoặc proxy pattern để theo dõi

☐ Sửa code bên trong processOrder()

☐ Thêm log thủ công tại mỗi điểm xử lý

☐ Kế thừa và override toàn bộ phương thức

✓ **Đúng** 1/1 Điểm

55

Kỹ thuật nào sau đây nên được ưu tiên để **kiểm soát sự phụ thuộc giữa các module** trong sơ đồ lớp? 


☐ Service Locator pattern

☒ Dependency Injection

☐ Global variable

☐ Static factory

✓ **Đúng** 1/1 Điểm

Theo em mối quan hệ giữa các UseCase nào sau đây là mối quan hệ generalization 

- ☐ Usecase Đăng bài với Usecase Xoá bài và Usecase Sửa bài
- ☐ Usecase Đăng nhập với Usecase Đăng xuất
- ☒ Usecase Rút tiền với Usecase Rút ATM và Usecase Rút tại quầy
- ☐ Usecase Quét vân tay với Usecase Quét thẻ và Usecase Quét võng mạc

✓ **Đúng** 1/1 Điểm

57


Ngân hàng Agribank (Bank) có quản lý nhiều loại tài khoản và giao dịch khác nhau. Mỗi tài khoản (Account) có thông tin: số tài khoản, số tiền. Một tài khoản đều ứng với một người chủ tài khoản (Person) với thông tin tên, tuổi và chứng minh thư. Một cá nhân có thể có tối đa 5 tài khoản trong ngân hàng.

Một chủ tài khoản có thể là khách hàng (Customer) hoặc nhân viên ngân hàng (Staff). Trong đó khách hàng có thêm thông tin về mã khách hàng; nhân viên có thêm thông tin về mã nhân viên.

Trong số các tài khoản còn có một loại tài khoản nữa là tài khoản bên ngoài (Outside), tài khoản này chỉ có thuộc tính đại diện cho số tài khoản (của ngân hàng khác) và tên chủ tài khoản, và tên ngân hàng khác. Tài khoản này không do ngân hàng Agribank lập ra mà của ngân hàng khác (chẳng hạn BIDV, VCB, MB...), vốn được gửi hoặc nhận bởi tài khoản của khách/nhân viên.

Khi một khoản tiền được gửi từ tài khoản này đến tài khoản khác, giao dịch (Transaction) đó có thông tin về tài khoản phía gửi và phía nhận; về số tiền, nội dung.

Ngân hàng có nhiệm vụ tìm ra khách hàng may mắn trong số các giao dịch trong tuần, và để làm được điều đó thì lớp Bank cần một phương thức getLuckyPerson() trả về một đối tượng kiểu Outside hoặc Customer.


Theo em phát biểu nào đúng về các tác nhân trong biểu đồ Usecase mô hình hóa cho hệ thống trên? 

- ☐ Giao dịch là một tác nhân
- ☒ Người nhân viên là một tác nhân
- ☐ Ngân hàng Agribank là một tác nhân

☐ Bảo vệ ngân hàng là một tác nhân

✓ **Đúng** 1/1 Điểm


58

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Việc PaymentProcessor triển khai IPaymentProcessor là ví dụ về cơ chế gì trong thiết kế hướng đối tượng? 

- ☐ Ủy quyền (Delegation)
- ☒ Đa hình (Polymorphism)
- ☐ Đa kế thừa
- ☐ Kế thừa triển khai

✓ **Đúng** 1/1 Điểm

59

Trong thiết kế chi tiết của một hệ thống thanh toán trực tuyến, Phương thức checkStock(productId: string) trong InventoryService nên được thiết kế thế nào để **tối thiểu hóa tính phụ thuộc (coupling)** từ lớp sử dụng? 

- ☒ Trả về một boolean để biểu thị tình trạng còn hàng
- ☐ Truy cập trực tiếp cơ sở dữ liệu trong lớp OrderProcessor
- ☐ Gửi phản hồi trực tiếp qua giao diện người dùng
- ☐ Trả về một exception khi không tìm thấy sản phẩm

✓ **Đúng** 1/1 Điểm

60

Phát biểu nào sau đây về sơ đồ Usecase đã cho là đúng? 

- ☐ A và B cùng thực hiện Y
- ☒ A và B có thể thực hiện Z riêng biệt
- ☐ Tất cả các phát biểu đều sai
- ☐ A và B cùng thực hiện Z



Nội dung này được tạo bởi chủ sở hữu của biểu mẫu. Dữ liệu bạn gửi sẽ được gửi đến chủ sở hữu biểu mẫu. Microsoft không chịu trách nhiệm về quyền riêng tư hoặc thực tiễn bảo mật của khách hàng, bao gồm cả các biện pháp bảo mật của chủ sở hữu biểu mẫu này. Không bao giờ đưa ra mật khẩu của bạn.

Microsoft Forms | Các cuộc khảo sát, câu đố và cuộc thăm dò do AI cung cấp [Tạo biểu mẫu riêng của tôi](#)

[Quyền riêng tư và cookie](#) | [Điều khoản sử dụng](#)