Review Chương 1 - 2 - 3:

I. <u>Câu hỏi trắc nghiệm:</u>

1. Pha nào trong tiến trình phát triển phần mềm tập trung vào việc thu thập yêu cầu từ khách hàng? A. Pha thiết kế
B. Pha kiểm thử
C. Pha lấy yêu cầu
D. Pha triển khai
2. Mô hình phát triển phần mềm nào có các vòng lặp liên tục và cải tiến sản phẩm sau mỗi lần lặp?
A. Mô hình thác nước
B. Mô hình xoắn ốc
C. Mô hình Agile
D. Mô hình kiểm thử
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?
A. Phỏng vấn
B. Khảo sát
C. Quan sát
D. Phân tích tài liệu
4. Trong chuẩn hóa cơ sở dữ liệu, dạng chuẩn nào loại bỏ các phụ thuộc bắc cầu?
A. 1NF
B. 2NF
C. 3NF
D. BCNF
5. Trong UML, sơ đồ nào giúp mô tả sự tương tác giữa các đối tượng theo thời gian?
A. Sơ đồ lớp
B. Sơ đồ tuần tự
C. Sơ đồ trạng thái
D. Sơ đồ hoạt động
6. UML, sơ đồ nào giúp mô tả sự tương tác giữa các đối tượng theo thời gian?
A. Sơ đồ lớp
B. Sơ đồ tuần tự
C. Sơ đồ trạng thái
D. Sơ đồ hoạt động
7. CMMI mức 5 – Optimizing tập trung vào điều gì?
A. Kiểm soát quy trình
B. Định lượng quy trình
C. Cải tiến quy trình
D. Xác định quy trình
8. Đâu là ưu điểm chính của mô hình phát triển phần mềm Agile?
A. Linh hoạt và thay đổi nhanh chóng
B. Yêu cầu chi tiết ngay từ đầu
C. Kiểm thử chỉ diễn ra ở giai đoạn cuối
D. Phù hợp với tất cả các loại dự án
9. Nguyên tắc DRY (Don’t Repeat Yourself) trong lập trình có ý nghĩa gì?
A. Hạn chế viết code lặp lại
B. Viết code dễ đọc hơn
C. Tăng tốc độ thực thi của chương trình
D. Giảm chi phí phát triển phần mềm
10. Yếu tố nào quan trọng nhất trong việc thiết kế một lớp trong lập trình hướng đối tượng?
A. Đặt tên biến dễ hiểu

B. Đảm bảo tính đóng gói và tái sử dụng
C. Viết phương thức càng nhiều càng tốt
D. Dùng số nguyên thay vì số thực
11. Phát biểu nào đúng về kiểm thử phần mềm?
A. Kiểm thử chỉ diễn ra sau khi phát triển xong sản phẩm
B. Kiểm thử giúp tìm ra tất cả lỗi trong phần mềm
C. Kiểm thử có thể thực hiện song song với phát triển phần mềm
D. Kiểm thử không quan trọng nếu phần mềm được viết tốt **II. <u>Câu hỏi trả lời ngắn:</u>**
1. Định nghĩa phần mềm và công nghệ phần mềm.
- Phần mềm là tập hợp các chương trình máy tính, dữ liệu liên quan và tài liệu hướng dẫn, giúp thực hiện các tác vụ cụ thể trên hệ thống máy tính
- Công nghệ phần mềm là lĩnh vực nghiên cứu và ứng dụng các nguyên tắc kỹ thuật để phát triển, bảo trì và quản lý phần mềm một cách hệ thống, hiệu quả và chất lượng cao.

2. Mô tả ngắn gọn về các mô hình vòng đời phát triển phần mềm phổ biến.
- Mô hình thác nước (Waterfall Model): quy trình tuần tự gồm các bước như lấy yêu cầu -> thiết kế -> lập trình -> kiểm thử -> triển khai
- Mô hình xoắn ốc (Spiral Model): Kết hợp phát triển lặp và quản lý rủi ro, thích hợp cho dự án lớn.
- Mô hình Agile: Chia nhỏ dự án thành các vòng lặp (iterations), phản hồi nhanh với thay đổi của khách hàng.
- Mô hình phát triển nhanh (RAD): Tập trung vào tạo mẫu nhanh (prototyping) và phát triển phần mềm trong thời gian ngắn
- Mô hình DevOps: Kết hợp phát triển và vận hành, giúp tự động hóa quy trình triển khai và bảo trì phần mềm.

3. Liệt kê ba loại yêu cầu phần mềm chính và giải thích từng loại.
- Yêu cầu chức năng: Mô tả chức năng mà hệ thống phải cung cấp
- Yêu cầu phi chức năng: Mô tả các tiêu chí về hiệu suất, bảo mật, khả năng mở rộng - Yêu cầu miền: Các yêu cầu đặc thù của ngành, lĩnh vực mà phần mềm phục vụ.

4. Mô tả vai trò của sơ đồ lớp UML trong thiết kế phần mềm.
- Mô tả cấu trúc tĩnh của hệ thống: Biểu diễn các lớp, thuộc tính, phương thức và mối quan hệ giữa chúng
- Giúp phân tích và thiết kế hệ thống một cách trực quan: Dễ dàng hiểu được cách các đối tượng tương tác với nhau
- Hỗ trợ chuyển đổi mã nguồn: Làm cơ sở để lập trình viên triển khai mã nguồn tương ứng.

5. Tại sao kiểm thử phần mềm quan trọng trong phát triển phần mềm?
- Phát hiện lỗi sớm: Giúp tìm ra và sửa lỗi truóc khi phần mềm đưa vào sử dụng
- Đảm bảo chất lượng phần mềm: Giúp đảm bảo rằng phần mềm đáp ứng các yêu cầu và hoạt động ổn định.
- Tăng độ tin cậy và bảo mật: Giảm nguy cơ xảy ra lỗi nghiêm trọng hoặc lỗ hổng bảo mật khi triển khai thực tế.

6. Định nghĩa nguyên lý SOLID trong lập trình hướng đối tượng.
- S - Single Responsibility Principle (SRP): Mỗi lớp chỉ nên có một trách nhiệm duy nhất
- O - Open/Closed Principle (OCP): Mở rộng được nhưng không sửa đổi mã nguồn cũ - L - Liskov Substitution Principle (LSP): Lớp con có thể thay thế lớp cha mà không thay đổi hành vi mong đợi
- I - Interface Segregation Principle (ISP): Không nên ép buộc một lớp phụ thuộc vào các phương thức mà nó không sử dụng
- D - Dependency Inversion Principle (DIP): Các module cấp cao không nên phụ thuộc vào các module cấp thấp mà cả hai nên phụ thuộc abstraction

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

Tiêu chí	Kiểm thử hộp đen	Kiểm thử hộp trắng

Mục đích	Kiểm tra chức năng của hệ thống	Kiểm tra logic trong mã nguồn
Người thực hiện	Tester	Developer
Phương pháp	Dựa trên đầu vào - đầu ra	Dựa trên cấu trúc mã nguồn
Ví dụ	Kiểm thử form đăng nhập	Kiểm tra vòng lặp, điều kiện if-else

8. Mô tả quy trình thiết kế cơ sở dữ liệu từ sơ đồ lớp UML.
- Xác định các thực thể (Entity): dựa trên các lớp trong sơ đồ UML
- Xác định các thuộc tính (Attributes): Mỗi thuộc tính trong lớp trở thành một cột trong bảng
- Xác định khóa chính (Primary Key): Chọn thuộc tính duy nhất để định danh mỗi bản ghi
- Xác định quan hệ (Relationships): Dựa vào quan hệ giữa các lớp (1-1, 1-n , n-n)
- Chuẩn hóa cơ sở dữ liêu: Áp dụng các dạng chuẩn (1NF, 2NF, 3NF, BCNF) để tối ưu dữ liệu
- Tạo lược đồ cơ sở dữ liệu (Database Schema): Chuyển đổi sơ đồ lớp UML thành lược đồ thưck tế.

9. Nêu ba ưu điểm của việc sử dụng mô hình phát triển phần mềm Agile.
- Linh hoạt và phản hồi nhanh: dễ dàng thích ứngvới yêu cầu thay đổi từ khách hàng
- Cải thiện chất lượng phần mềm: Liên tục kiểm thử và tối ưu phần mềm qua từng vòng lặp - Tăng sự hợp tác và minh bạch: Nhóm phát triển làm việc chặt chẽ với khách hàng và các bên liên quan.

10.Liệt kê các giai đoạn chính trong quá trình chuẩn hóa cơ sở dữ liệu.
- 1NF: loại bỏ giá trị trùng lặp, mỗi ô trong bảng chỉ chứa giá trị duy nhất
- 2NF: Đảm bảo mọi thuộc tính không khóa phụ thuộc hoàn chỉnh vào khóa chính
- 3NF: Loại bỏ các phụ thuộc bắc cầu
- BCNF: một phiên bản nâng cao của 3NF, đảm bảo mỗi phụ thuộc hàm phi tầm thường có dạng X->Y thì X phải là siêu khóa.

III. <u>Câu hỏi thảo luận nhóm:</u>

1. So sánh mô hình phát triển phần mềm Agile và mô hình Waterfall.

Tiêu chí	Agile	Waterfall
Cách tiếp cận	Linh hoạt, phát triển lặp	Tuần tự, từng giai đoạn
Khả năng thay đổi yêu cầu	Dễ thay đổi	Khó thay đổi sau khi bắt đầu
Tương tác với khách hàng	Liên tục, khách hàng tham gia vào quy trình	Chủ yếu ở giai đoạn đầu và cuối
Kiểm thử	Diễn ra liên tục trong quá trình phát triển	Kiểm thử chỉ diễn ra sau khi hoàn thành phát triển
Khi nào phù hợp?	Dự án thay đổi nhanh, cần phản hồi liên tục	Dự án có yêu cầu rõ ràng, ít thay đổi

2. Lợi ích của việc sử dụng UML trong thiết kế phần mềm là gì?
- Trực quan hóa hệ thống: Biểu diễn cấu trúc và hành vi của phần mềm dễ hiểu hơn.
- Chuẩn hóa thiết kế: UML là ngôn ngữ chung giúp các nhóm phát triển làm việc nhất quán.
- Hỗ trợ lập trình: Chuyển đổi sơ đồ UML thành mã nguồn dễ dàng.
- Dễ bảo trì và mở rộng: Giúp tài liệu hóa hệ thống, hỗ trợ bảo trì lâu dài.

3. Làm thế nào để đảm bảo phần mềm có thể bảo trì tốt trong tương lai? -
Tuân theo nguyên tắc SOLID để viết code dễ bảo trì.
- Viết tài liệu đầy đủ để giúp lập trình viên hiểu rõ hệ thống.
- Viết code sạch (Clean Code) và có cấu trúc rõ ràng.
- Kiểm thử kỹ lưỡng để đảm bảo hệ thống ổn định.
- Sử dụng thiết kế hướng đối tượng và mô-đun hóa để dễ dàng thay đổi, mở rộng.

4. Tại sao các công ty phần mềm thường sử dụng mô hình phát triển lặp (Iterative Development)?
- Phát hiện lỗi sớm: Mỗi lần lặp giúp kiểm tra và cải tiến sản phẩm.

- Thích ứng với thay đổi: Có thể điều chỉnh yêu cầu theo phản hồi của khách hàng.
- Giảm rủi ro: Phát triển từng phần nhỏ giúp kiểm soát rủi ro tốt hơn.
- Nâng cao chất lượng phần mềm: Kiểm thử liên tục giúp sản phẩm ổn định hơn.

5. Vai trò của kiến trúc phần mềm trong việc xây dựng một hệ thống phần mềm phức tạp. - Cung cấp cấu trúc tổng thể để đảm bảo hệ thống dễ mở rộng và bảo trì.
- Hỗ trợ khả năng mở rộng (Scalability) khi hệ thống cần phát triển lớn hơn.
- Giúp tối ưu hiệu suất bằng cách thiết kế hợp lý ngay từ đầu.
- Cải thiện bảo mật bằng cách áp dụng các mô hình bảo mật chuẩn.

6. Làm thế nào để đảm bảo rằng một hệ thống phần mềm đáp ứng được yêu cầu bảo mật?
- Sử dụng mã hóa dữ liệu để bảo vệ thông tin quan trọng.
- Xác thực và phân quyền người dùng để kiểm soát truy cập.
- Kiểm thử bảo mật thường xuyên để phát hiện và sửa lỗi bảo mật.
- Áp dụng các nguyên tắc bảo mật như Principle of Least Privilege (PoLP) để giảm thiểu rủi ro.

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

Tiêu chí	Unit Testing	Intergration Testing
Mục tiêu	Kiểm tra từng phần nhỏ (hàm, class)	Kiểm tra sự kết hợp giữa các module
Ai thực hiện	Lập trình viên	Tester hoặc lập trình viên
Thời điểm thực hiện	Ngay khi viết xong một phần nhỏ của code	Sau khi các module được phát triển xong
Công cụ	JUnit, NUnit, PTest	Selenium, TestNG, Postman

8. Những thách thức chính trong việc thu thập yêu cầu phần mềm là gì? - Khách hàng không biết chính xác họ cần gì → Cần tư vấn kỹ lưỡng.
- Yêu cầu liên tục thay đổi → Cần linh hoạt trong phát triển.
- Sự mơ hồ trong yêu cầu → Cần tài liệu hóa rõ ràng và xác nhận lại với khách hàng. - Mâu thuẫn giữa các bên liên quan → Cần đàm phán và thống nhất yêu cầu 9. Cách áp dụng quy trình Scrum vào dự án phát triển phần mềm thực tế.
- Lập kế hoạch Sprint: Xác định các nhiệm vụ cần làm trong 2-4 tuần.
- Họp Daily Standup: Cập nhật tiến độ hàng ngày.
- Thực hiện phát triển và kiểm thử liên tục trong mỗi Sprint.
- Họp Sprint Review: Đánh giá sản phẩm sau mỗi Sprint.
- Họp Sprint Retrospective: Rút kinh nghiệm và cải thiện quy trình cho Sprint tiếp theo.

10.Tại sao việc tối ưu hóa mã nguồn lại quan trọng trong phát triển phần mềm?
- Tăng hiệu suất: Giúp phần mềm chạy nhanh hơn và sử dụng ít tài nguyên hơn.
- Giảm chi phí bảo trì: Code tối ưu dễ đọc, dễ sửa lỗi và mở rộng.
- Cải thiện trải nghiệm người dùng: Ứng dụng mượt mà, phản hồi nhanh hơn.
- Giảm nguy cơ lỗi và lỗ hổng bảo mật: Mã nguồn sạch sẽ giúp dễ kiểm soát hơn.

IV. <u>Câu hỏi tình huống:</u>

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?
- Sử dụng mô hình phát triển linh hoạt (Agile, Scrum) để dễ dàng thích ứng với thay đổi.
- Xác định mức độ ưu tiên của yêu cầu: Thảo luận với khách hàng để xem yêu cầu nào quan trọng nhất.
- Tài liệu hóa thay đổi: Mỗi thay đổi cần được ghi nhận rõ ràng để đánh giá tác động đến tiến độ và chi phí.
- Thương lượng về chi phí và thời gian: Nếu thay đổi lớn, cần điều chỉnh ngân sách và lịch trình.

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?

- Báo cáo chi tiết về mức độ ảnh hưởng: Cung cấp bằng chứng (log, test case) để chứng minh lỗi nghiêm trọng.
- Thảo luận với đội ngũ kỹ thuật: Nếu lỗi ảnh hưởng đến bảo mật hoặc dữ liệu, cần thuyết phục trưởng nhóm xem xét lại.
- Báo cáo lên cấp trên hoặc khách hàng: Nếu lỗi có thể gây thiệt hại nghiêm trọng, cần báo cáo cho người có thẩm quyền 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?
- Áp dụng phương pháp Agile với Sprint ngắn hơn để thích ứng với thay đổi mà không ảnh hưởng lớn đến tiến độ.
- Thiết lập phạm vi yêu cầu rõ ràng: Xác định yêu cầu cốt lõi trước khi phát triển. - Sử dụng hợp đồng linh hoạt: Thỏa thuận với khách hàng về phạm vi thay đổi để tránh phát sinh quá nhiều yêu cầu.
- Tạo một hệ thống ưu tiên yêu cầu: Chỉ thay đổi những yêu cầu quan trọng nhất.

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?
- Áp dụng chuẩn hóa dữ liệu (Normalization) đến ít nhất 3NF để loại bỏ dữ liệu dư thừa.
- Xác định đúng các thực thể và quan hệ để tránh lặp lại dữ liệu.
- Thiết kế chỉ mục (Indexing) hợp lý để tối ưu truy vấn mà không làm tăng kích thước cơ sở dữ liệu không cần thiết.
- Sử dụng công cụ mô hình hóa dữ liệu (ERD, UML) để kiểm tra mối quan hệ giữa các bảng

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 Agile, vì:
- Startup cần thích ứng nhanh với thị trường.
- Dễ dàng thay đổi và cập nhật sản phẩm theo phản hồi của khách hàng.
- Giúp phát triển sản phẩm theo từng giai đoạn mà không cần yêu cầu chi tiết ngay từ đầu.

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 hiện tại bằng công cụ như New Relic, JProfiler.
- Tối ưu truy vấn cơ sở dữ liệu bằng indexing, caching (Redis, Memcached).
- Cải tiến thuật toán và cấu trúc dữ liệu để giảm độ phức tạp.
- Sử dụng load balancing và scaling nếu hệ thống gặp vấn đề về tải.
- Giảm tải cho server bằng cách sử dụng CDN, nén dữ liệu (Gzip, Brotli).

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?
- Cập nhật bảo mật dần dần: Triển khai bản vá nhỏ thay vì thay đổi lớn.
- Mã hóa dữ liệu quan trọng: Đảm bảo dữ liệu người dùng được bảo vệ (AES, TLS).
- Tăng cường xác thực (MFA) để giảm nguy cơ bị tấn công tài khoản.
- Kiểm thử bảo mật thường xuyên để phát hiện và sửa lỗi trước khi bị khai thác

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 đánh giá các phương án dựa trên tiêu chí chung (hiệu suất, bảo trì, bảo mật).
- Chạy thử nghiệm A/B: Kiểm tra hai giải pháp để xem phương án nào tối ưu hơn.
- Tham khảo ý kiến chuyên gia hoặc kiến trúc sư phần mềm nếu cần.
- Dựa vào nguyên tắc SOLID, DRY, KISS để đảm bảo thiết kế tốt nhất.

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?
- Thực hiện khảo sát người dùng để hiểu nhu cầu và hành vi của họ.
- Sử dụng nguyên tắc UI/UX (Material Design, Human Interface Guidelines) để tạo trải nghiệm tốt.

- Làm nguyên mẫu (Prototyping) và thử nghiệm với người dùng thực tế trước khi phát triển chính thức.
- Đảm bảo hệ thống phản hồi nhanh, dễ sử dụng trên nhiều thiết bị

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: Xác định kiến trúc hệ thống, thư viện sử dụng.
- Dùng công cụ Reverse Engineering như Doxygen, Understand để tạo tài liệu tự động.
- Liên hệ với đội ngũ phát triển trước (nếu có thể) để hiểu thêm về hệ thống.
- Viết lại tài liệu kỹ thuật: Ghi nhận thông tin về API, cơ sở dữ liệu, quy trình triển khai.
- Tạo test case để kiểm tra hệ thống trước khi thực hiện bất kỳ thay đổi nào. **V. <u>Bài tập thực hành:</u>**

1. Viết một chương trình Java để quản lý thông tin sinh viên sử dụng lập trình hướng đối tượng.

```java
import java.util.ArrayList;
import java.util.Scanner;

// Lớp Student đại diện cho sinh viên
class Student {
    private String id;
    private String name;
    private int age;
    private double gpa;

    public Student(String id, String name, int age, double gpa) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gpa = gpa;
    }

    public String getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public double getGpa() {
        return gpa;
    }

    public void setName(String name) {
        this.name = name;
    }
    public void setAge(int age) {
        this.age = age;
    }

    public void setGpa(double gpa) {
        this.gpa = gpa;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Age: " + age + ", GPA: " + gpa;
    }
}

// Lớp StudentManager để quản lý danh sách sinh viên
class StudentManager {
    private ArrayList<Student> students = new ArrayList<>();

    public void addStudent(Student student) {
        students.add(student);
    }

    public boolean removeStudent(String id) {
        return students.removeIf(student -> student.getId().equals(id));
    }

    public Student findStudent(String id) {
        for (Student student : students) {
            if (student.getId().equals(id)) {
                return student;
            }
        }
        return null;
    }
```

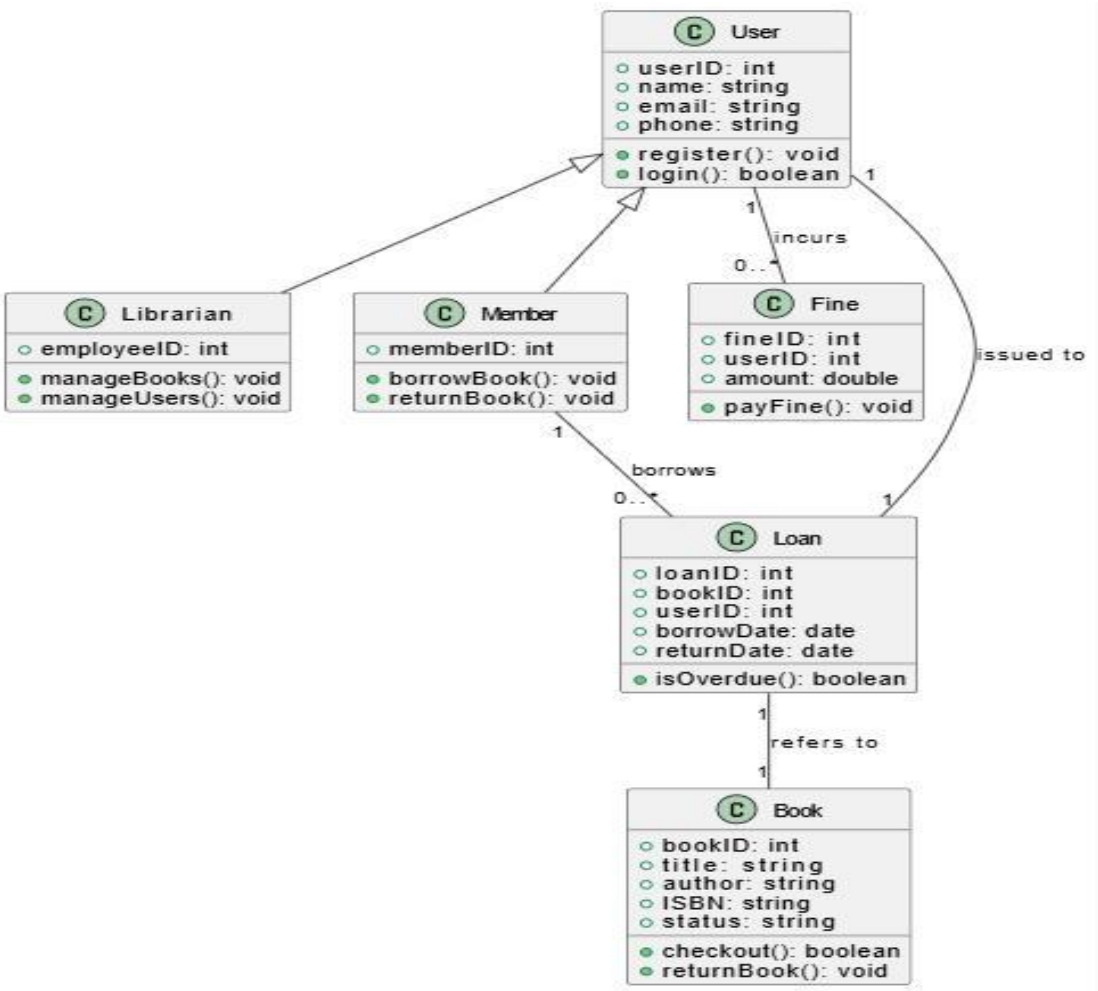
```java
    public void displayStudents() {
        for (Student student : students) {
            System.out.println(student);
        }
    }
}

// Lớp Main để chạy chương trình
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StudentManager manager = new StudentManager();

        while (true) {
            System.out.println("\nSTUDENT MANAGEMENT SYSTEM");
            System.out.println("1. Add Student");
            System.out.println("2. Remove Student");
            System.out.println("3. Find Student");
            System.out.println("4. Display Students");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Enter ID: ");
                    String id = scanner.nextLine();
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Age: ");
                    int age = scanner.nextInt();
                    System.out.print("Enter GPA: ");
                    double gpa = scanner.nextDouble();
                    scanner.nextLine();
                    manager.addStudent(new Student(id, name, age, gpa));
                    break;
                case 2:
                    System.out.print("Enter ID to remove: ");
                    String removeId = scanner.nextLine();
                    if (manager.removeStudent(removeId)) {
                        System.out.println("Student removed successfully.");
                    } else {
                        System.out.println("Student not found.");
                    }
                    break;
                case 3:
                    System.out.print("Enter ID to find: ");
                    String findId = scanner.nextLine();
                    Student student = manager.findStudent(findId);
                    if (student != null) {
                        System.out.println("Student found: " + student);
                    } else {
                        System.out.println("Student not found.");
                    }
                    break;
                case 4:
                    manager.displayStudents();
                    break;
                case 5:
                    System.out.println("Exiting program...");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}
```

2. Xây dựng sơ đồ lớp UML cho hệ thống quản lý thư viện.



3. Viết test case cho tính năng đăng ký tài khoản trong một hệ thống website.

STT	Tên Test Case	Mô tả	Dữ liệu đầu vào	Kết quả mong đợi	Trạng thái
1	Đăng ký thành công với dữ liệu hợp lệ	Kiểm tra xem hệ thống có cho phép đăng ký với dữ liệu hợp lệ không	Username: testuser Email: test@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo đăng ký thành công	Pass/Fail
2	Đăng ký với username đã tồn tại	Kiểm tra nếu username đã tồn tại thì có thể đăng ký không.	Username:existinguser Email: new@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo lỗi “Useranme đã tồn tại”	Pass/Fail

3	Đăng ký với email đã tồn tại	Kiểm tra nếu email đã được sử dụng thì có thể đăng ký không	Username:newuser Email: existing@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thông báo lỗi "Email đã được sử dụng"	Pass/Fail
4	Đăng ký với mật khẩu đã trùng khớp	Kiểm tra xác nhận mật khẩu	Username: testuser1 Email: test1@example.com Password: Test@1234 Confirm Password: Test@12345	Hiển thị thông báo lỗi "Mật khẩu không trùng khớp"	Pass/Fail
5	Đăng ký với mật khẩu yếu	Kiểm tra hệ thống có chấp nhận mật khẩu yếu không	Username: testuser2 Email: test2@example.com Password: 12345 Confirm Password: 12345	Hiển thị cảnh báo "Mật khẩu quá yếu, vui lòng sử dụng mật khẩu mạng hơn"	Pass/Fail
6	Đăng ký không nhập email	Kiểm tra hệ thống có bắt buộc nhập email không	Username: testuser3 Email: Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo lỗi "Email không được để trông"	Pass/Fail
7	Đăng ký với email không hợp lệ	Kiểm tra hệ thống có xác thực định dạng email không	Username: testuser4 Email: invalid - email Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo lỗi "Email không hợp lệ"	Pass/Fail
8	Đăng ký không nhập mật khẩu	Kiểm tra hệ thống có bắt buộc nhập mật khẩu không	Username: testuser5 Email: test5@example.com Password: Confirm Password:	Hiển thị thông báo lỗi "Mật khẩu không được để trống"	Pass/Fail
9	Đăng ký không nhập username	Kiểm tra hệ thống có bắt buộc nhập username không	Username: Email: test6@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo lỗi "Username không được để trống"	Pass/Fail

10	Đăng ký với username chứa ký tự đặc biệt	Kiểm tra hệ thống có chấp nhận username chứa ký tự đặc biệt không	Username: user@123 Email: test7@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo lỗi "Username chỉ được chứa chữ cái và số"	Pass/Fail
11	Đăng ký khi server bị lỗi	Kiểm tra hệ thống có xử lý lỗi server khi đăng nhập không	Username: testuser8 Email: test7@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo "Hệ thống đang gặp lỗi, vui lòng thử lại sau"	Pass/Fail
12	Đăng ký với mật khẩu chỉ có chữ hoặc số	Kiểm tra hệ thống có yêu cầu mật khẩu mạnh không	Username: testuser9 Email: test9@example.com Password: passwordonly Confirm Password: passwordonly	Hiển thị thông báo "Mật khẩu phải chứa ít nhất một chữ cái, một số và một ký tự đặc biệt"	Pass/Fail
13	Đăng ký với username quá dài	Kiểm tra hệ thống có giới hạn độ dài username không	Username: usernametooooooooooooooooooloong Email: test1@example.com Password: Test@1234 Confirm Password: Test@1234	Hiển thị thông báo "Username không được dài quá 20 ký tự"	Pass/Fail

4. Thiết kế cơ sở dữ liệu cho hệ thống bán hàng trực tuyến và chuẩn hóa đến 3NF.
- User: userID, username, email, passwordHash, phone, address
- Category: categoryID, categoryName, description
- Product: productID, productName, description, price, stock, categoryID, imageURL
- Order: orderID, userID, orderDate, totalAmount
- OrderDetail: orderDetailID, orderID, productID, quantity, price, subtotal
- Payment: paymentID, orderID, paymentMethod, paymentStatus, paymentDate
- Shipping: shippingID, orderID, trackingNumber, carrier, shippingDate, estimateDeliveryDate, deliveryStatus

5. Viết một đoạn mã Java đơn giản để minh họa nguyên tắc đóng gói trong lập trình hướng đối tượng.

```java
class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        setAge(age); // Sử dụng setter để kiểm tra hợp lệ
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    // Setter để thiết lập giá trị age (có kiểm tra hợp lệ)
    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("Tuổi không hợp lệ, đặt về giá trị mặc định là 1.");
            this.age = 1;
        }
    }

    // Phương thức hiển thị thông tin
    public void displayInfo() {
        System.out.println("Tên: " + name + ", Tuổi: " + age);
    }
}
```

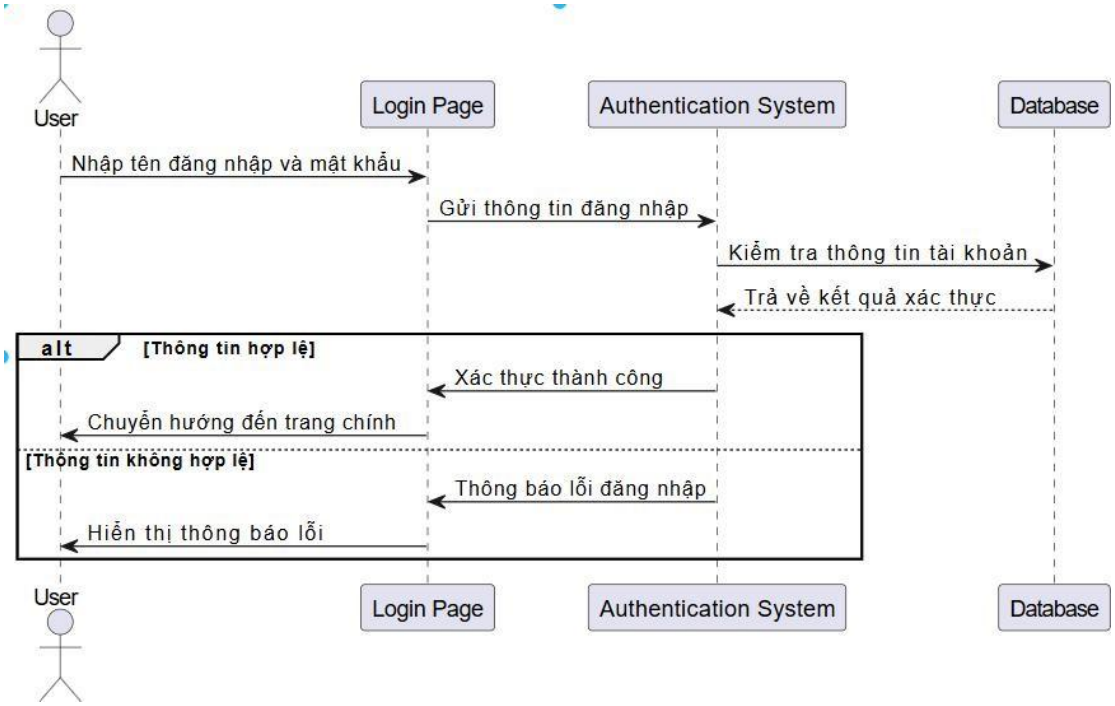
```java
public class EncapsulationExample {
    public static void main(String[] args) {
        Person person = new Person("Alice", 25);
        person.displayInfo();

        // Thử thiết lập tuổi không hợp lệ
        person.setAge(-5);

        // Hiển thị lại thông tin sau khi cập nhật
        person.displayInfo();

        // Truy xuất thông tin bằng getter
        System.out.println("Tên của người dùng: " + person.getName());
    }
}
```

6. Xây dựng sơ đồ tuần tự UML cho chức năng "Đăng nhập hệ thống".



7. Viết chương trình kiểm thử đơn vị (Unit Test) cho một phương thức tính tổng hai số trong Java.

```
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

```java
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class CalculatorTest {
    private Calculator calculator;

    @BeforeEach
    void setUp() {
        calculator = new Calculator();
    }

    @Test
    void testAddition() {
        int result = calculator.add(3, 5);
        assertEquals(8, result, "3 + 5 should be 8");
    }

    @Test
    void testAdditionWithZero() {
        int result = calculator.add(0, 5);
        assertEquals(5, result, "0 + 5 should be 5");
    }

    @Test
    void testAdditionWithNegativeNumbers() {
        int result = calculator.add(-3, -7);
        assertEquals(-10, result, "-3 + (-7) should be -10");
    }
}
```

```xml
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

8. Tạo kịch bản kiểm thử chức năng thanh toán trong hệ thống thương mại điện tử.

❖ Thông tin chung:

- Chức năng: Thanh toán người dùng
- Mô tả: Người dùng có thể thanh toán đơn hàng bằng nhiều phương thức khác nhau (thẻ tín dụng, PayPal, COD, ví điện tử,...) ❖ Test Case chi tiết:
- Test Case 1: Thanh toán bằng thẻ tín dụng:

ID	TC001
Tên test case	Thanh toán thành công bằng thẻ tín dụng
Điều kiện tiền đề	Giỏ hàng có ít nhất một sản phẩm, người dùng đã đăng nhập
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán thẻ tín dụng 3. Nhập thông tin thẻ hợp lệ 4. Nhấn “Thanh toán”
Kết quả mong đợi	Hệ thống xác nhận thanh toán thành công và hiển thị thông báo xác nhận

- Test Case 2: Thanh toán thất bại do nhập sai thông tin thẻ

ID	TC002

Tên test case	Thanh toán thất bại do nhập sai thông tin thẻ
Điều kiện tiền đề	Giỏ hàng có ít nhất một sản phẩm, người dùng đã đăng nhập
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán thẻ tín dụng 3. Nhập thông tin thẻ hợp lệ 4. Nhấn "Thanh toán"
Kết quả mong đợi	Hệ thống hiển thị lỗi "Thông tin thẻ không hợp lệ"

- Test Case 3: Thanh toán bằng thẻ PayPal:

ID	TC003
Tên test case	Thanh toán thành công bằng PayPal
Điều kiện tiền đề	Giỏ hàng có ít nhất một sản phẩm, người dùng đã đăng nhập
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán PayPal 3. Đăng nhập tài khoản PayPal 4. Xác nhận "Thanh toán"
Kết quả mong đợi	Hệ thống xác nhận thanh toán thành công và hiển thị thông báo xác nhận

- Test Case 4: Thanh toán COD:

ID	TC004
Tên test case	Thanh toán thành công bằng COD
Điều kiện tiền đề	Giỏ hàng có ít nhất một sản phẩm, người dùng đã đăng nhập
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán "COD" 3. Xác nhận đơn hàng
Kết quả mong đợi	Hệ thống xác nhận thanh toán thành công và thông báo "Thanh toán khi nhận hàng"

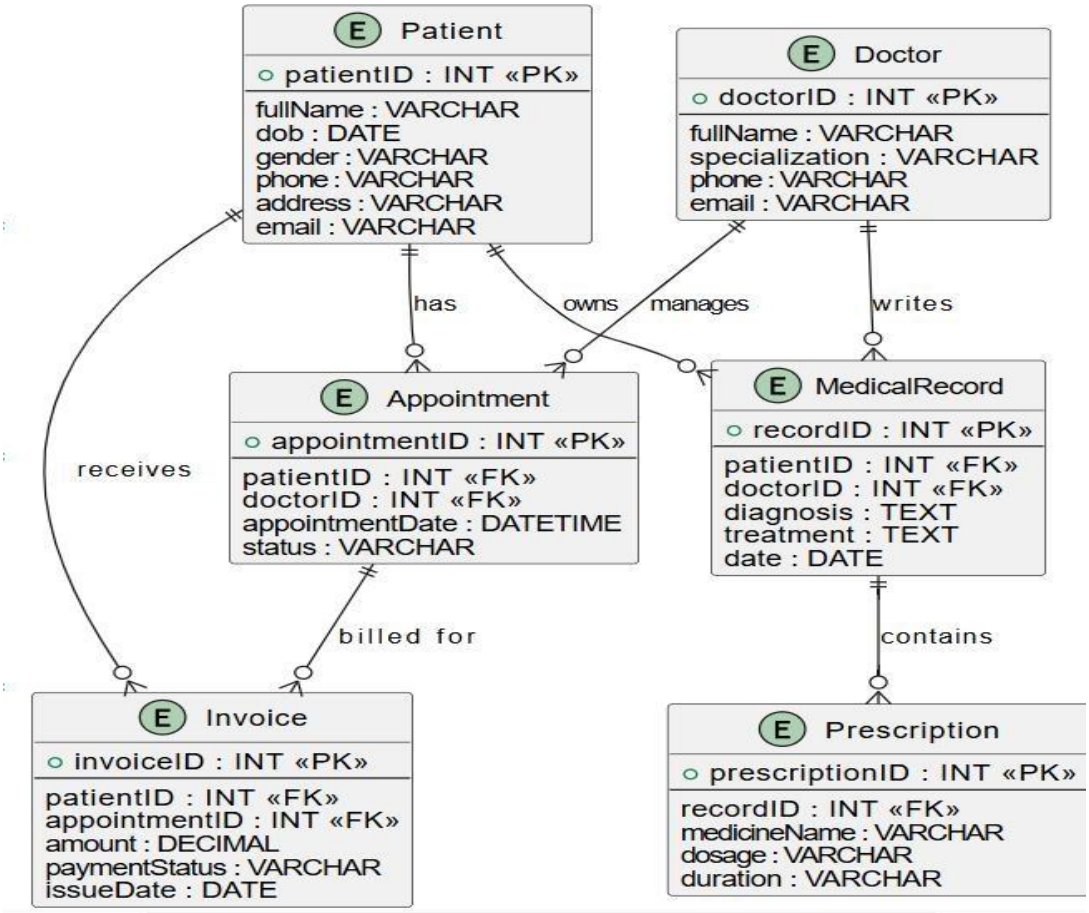
- Test Case 5: Hủy thanh toán giữa chừng:

ID	TC005
Tên test case	Hủy thanh toán giữa chừng
Điều kiện tiền đề	Giỏ hàng có ít nhất một sản phẩm, người dùng đã đăng nhập
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán 3. Nhấn "Hủy thanh toán"trước khi xác nhận
Kết quả mong đợi	Hệ thống đưa người dùng trở lại giỏ hàng mà không trừ tiền

- Test Case 6: Kiểm thử giao dịch khi mất kết nối Internet:

ID	TC006
Tên test case	Kiểm thử giao dịch khi mất kết nối Internet
Điều kiện tiền đề	Người dùng đang thực hiện thanh toán
Bước thực hiện	1. Truy cập giỏ hàng 2. Chọn phương thức thanh toán 3. Tắt kết nối Internet 4. Nhấn "Thanh toán"
Kết quả mong đợi	Hệ thống hiển thị thông báo "Lỗi kết nối, vui lòng thử lại"

9. Xây dựng mô hình thực thể (ERD) cho hệ thống quản lý bệnh viện.



10. Phân tích và mô tả một use case cụ thể trong hệ thống đặt vé máy bay.
- Tên use case: Đặt vé máy bay
- Mô tả: Use Case mô tả quy trình hành khách tìm kiếm chuyến bay, chọn vé, nhập thông tin cá nhân, thanh toán và nhận vé điện tử - Tác nhân liên quan:
+ Hành khách: Người sử dụng hệ thống để đặt vé
+ Hệ thống thanh toán: Xử lý giao dịch thanh toán + Hệ thống
quản lý đặt xe: Xác nhận và lưu trữ thông tin đặt vé - Luồng sự
kiện chính:
+ Hành khách truy cập vào hệ thống đặt vé máy bay
+ Hành khách nhập thông tin chuyến bay cần tìm kiếm (điểm đi, điểm đến, ngày bay, số lượng hành khách)
+ Hệ thống hiển thị danh sách chuyến bay phù hợp
+ Hành khách chọn chuyến bay mong muốn
+ Hành khách nhập thông tin cá nhân
+ Hành khách chọn phương thức thanh toán và tiến hành thanh toán
+ Hệ thống thanh toán xác nhận giao dịch
+ Hệ thống đặt vé xác nhận thông tin đặt chỗ và gửi tiền vé điện tử qua email +
Hành khách nhận được vé điện tử.
- Luồng sự kiện phụ:
+ TH1: Không có chuyến bay phù hợp -> Hệ thống thông báo không tìm thấy chuyên bay + TH2: Thanh toán thất bại -> Hệ thống thông báo lỗi và yêu cầu thử lại với phương thức khác
+ TH3: Nhập sai thông tin cá nhân -> Hệ thống yêu cầu nhập lại thông tin chính xác -
Điều kiện tiên quyết:
+ Hành khách có tài khoản hoặc có thể đặt vé mà không cần đăng nhập +
Hệ thống có kết nối với cơ sở dữ liệu chuyến bay và cổng thanh toán
- Kết quả: Hành khách đặt vé thành công và nhận được vé điện tử qua email.