**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

Trường Công nghệ thông tin và Truyền thông

A red and white logo

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN**

ĐỀ TÀI: THIẾT KẾ HƯỚNG ĐỐI TƯỢNG CHO BÀI TOÁN QUẢN LÝ PHÒNG GYM

Môn: Lập trình hướng đối tượng – IT3103 (Mã lớp: 151964)

Nhóm: 7

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn | : | Thầy Lê Đức Hậu |
| Họ và tên – MSSV | : | Nguyễn Lê Minh – 20225651  Trần Quang Huy – 20225860  Phạm Lê Thành – 20225765 |

**PHÂN CÔNG CÔNG VIỆC VÀ ĐÓNG GÓP**

|  |  |  |  |
| --- | --- | --- | --- |
| Họ và tên | MSSV | Đóng góp (%) | Công việc |
| Nguyễn Lê Minh | 20225651 | 30 | GUI, làm báo cáo |
| Trần Quang Huy (Leader) | 2022 | 40 | Thiết kế hệ thống, Thiết kế usecase. |
| Phạm Lê Thành | 20225765 | 30 | Làm slide, Thiết kế giao diện. |

Mục lục

[**Lời nói đầu** 4](#_Toc186011640)

[**1. Giới thiệu** 5](#_Toc186011641)

[1.1. Đặt vấn đề: 5](#_Toc186011642)

[1.2. Yêu cầu hệ thống 5](#_Toc186011643)

[1.3. Công nghệ sử dụng 5](#_Toc186011644)

[**2. Thiết kế hệ thống** 6](#_Toc186011645)

[2.1. Biểu đồ UML 7](#_Toc186011646)

[2.1.1. Usecase Diagram 7](#_Toc186011647)

[2.1.2. Class Diagram 7](#_Toc186011648)

[**3. Phân tích và mô tả hệ thống** 10](#_Toc186011649)

[**4. Kỹ thuật lập trình hướng đối tượng đã áp dụng** 15](#_Toc186011650)

[**5. Demo các chức năng của chương trình** 20](#_Toc186011651)

[**6. Kết luận** 24](#_Toc186011652)

# 

# **Lời nói đầu**

Trong bối cảnh hiện đại, việc quản lý phòng gym không chỉ dừng lại ở việc cung cấp thiết bị tập luyện mà còn đòi hỏi một hệ thống tổ chức thông tin hiệu quả, minh bạch và chính xác. Đặc biệt, với sự gia tăng số lượng người sử dụng dịch vụ cùng các yêu cầu về chất lượng trải nghiệm ngày càng cao, việc ứng dụng công nghệ vào quản lý phòng gym trở thành một nhu cầu thiết yếu.

Đề tài **“Thiết kế hướng đối tượng cho bài toán Quản lý phòng gym”** được thực hiện nhằm mục tiêu xây dựng một mô hình hệ thống quản lý thông tin phòng gym cơ bản nhưng toàn diện. Thông qua việc áp dụng các nguyên lý thiết kế hướng đối tượng, bài báo cáo không chỉ hướng đến việc giải quyết các vấn đề hiện tại mà còn đảm bảo tính linh hoạt, khả năng mở rộng và dễ dàng bảo trì hệ thống trong tương lai.

Bài báo cáo bao gồm các phần từ phân tích bài toán, xây dựng mô hình lớp cho đến thiết kế các phương thức quản lý cụ thể. Trong đó, các thực thể chính như hội viên, gói dịch vụ, lịch tập, và thiết bị tập luyện được phân tích chi tiết, đảm bảo phản ánh sát thực tế hoạt động của phòng gym.

Chúng em hy vọng rằng bài báo cáo này sẽ không chỉ cung cấp những kiến thức nền tảng về thiết kế hệ thống mà còn là tài liệu tham khảo hữu ích cho các nhà quản lý và các cá nhân, tổ chức quan tâm đến việc hiện đại hóa công tác quản lý phòng gym.

Trân trọng cảm ơn!

# **1. Giới thiệu**

## 1.1. Đặt vấn đề:

- Quản lý các phòng gym hiện nay là một bài toán phổ biến nhưng vẫn gặp nhiều thách thức trong việc số hóa và tối ưu hóa quy trình quản lý.  
- Các phòng gym cần hệ thống để:

+ Theo dõi thiết bị, lịch sử bảo trì.

+ Quản lý khóa học, lớp học, huấn luyện viên và học viên.

+ Đáp ứng nhu cầu phân quyền rõ ràng giữa các vai trò (Admin, Member, Trainer).  
+ Hệ thống phải đảm bảo tính dễ mở rộng, bảo trì và hỗ trợ tính năng quản lý tập trung.

## 1.2. Yêu cầu hệ thống

- Hệ thống phải đáp ứng các yêu cầu sau:

1. **Quản lý vai trò:**

* Admin: Thực hiện quản lý người dùng, thiết bị, khóa học, lớp học và thống kê dữ liệu.
* Trainer: Quản lý lớp học của mình, xem danh sách học viên.
* Member: Đăng ký membership, tham gia lớp học, xem lịch sử thanh toán.

1. **Quản lý thiết bị và tài nguyên:** Theo dõi trạng thái thiết bị và khả năng sử dụng.
2. **Tối ưu hóa quy trình đăng ký:** Đảm bảo học viên không đăng ký trùng giờ học hoặc vượt quá giới hạn.
3. **Hỗ trợ phân quyền:** Đảm bảo các vai trò chỉ truy cập được chức năng được phân quyền.
4. **Mở rộng dễ dàng:** Hỗ trợ thêm chức năng hoặc vai trò mới mà không làm ảnh hưởng hệ thống hiện tại.
5. **Tính tái sử dụng:** Nhiều chức năng chung (như CRUD) nhưng thực hiện trên các thực thể khác nhau (User, Equipment, Course...).

## 1.3. Công nghệ sử dụng

- Front-end: **JavaFX**

- Back-end: **Java**

# **2. Thiết kế hệ thống**

- Giới thiệu chung:

1. **Entities (Thực thể):**
   * User: Lớp cha, gồm các trường cơ bản (username, password, role).
     + Member: Kế thừa User, có thêm thông tin Membership.
     + Trainer: Kế thừa User, có thêm lương và chuyên môn (specialization).
   * Course: Thông tin khóa học (tên, mô tả, giá, số lượng học viên tối đa).
   * GymClass: Thông tin lớp học thuộc Course (giảng viên, lịch học, số lượng tối đa học viên).
   * Equipment: Thông tin thiết bị (tên, trạng thái, số lượng, giá, vị trí).
   * MembershipCard: Thông tin thẻ thành viên (loại thẻ, ngày bắt đầu, ngày kết thúc).
   * Payment: Thông tin thanh toán (người dùng, số tiền, mục đích).
2. **Services (Dịch vụ):**
   * UserService: Quản lý người dùng (CRUD).
   * MembershipService: Quản lý thẻ Membership.
   * CourseService: Quản lý khóa học.
   * ClassService: Quản lý lớp học.
   * EquipmentService: Quản lý thiết bị.
   * PaymentService: Quản lý thanh toán.
3. **Command Pattern:**
   * Mỗi hành động (use case) được triển khai dưới dạng một Command độc lập.
   * Ví dụ:
     + AddUserCommand: Thêm người dùng.
     + ViewAllCoursesCommand: Xem danh sách khóa học.
     + RegisterMembershipCommand: Đăng ký Membership.

## 2.1. Biểu đồ UML

### 2.1.1. Usecase Diagram

Ảnh có chứa hình vẽ, biểu đồ, văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

### 2.1.2. Class Diagram

**- Domain Layer (Model Layer):**

**Mục đích:** Đại diện cho các đối tượng cốt lõi trong hệ thống, chứa logic liên quan đến từng đối tượng.

Ảnh có chứa văn bản, biểu đồ, Kế hoạch, ảnh chụp màn hình

Mô tả được tạo tự động

* **Persistence Layer (lớp trừu tượng):**

**Mục đích:** Kết nối hệ thống với cơ sở dữ liệu hoặc lưu trữ dữ liệu.

Ảnh có chứa văn bản, biểu đồ, Song song, Phông chữ

Mô tả được tạo tự động

* **Persistence Layer (Lớp Implement):**

**Mục đích:** Kết nối hệ thống với cơ sở dữ liệu hoặc lưu trữ dữ liệu.

Ảnh có chứa văn bản, Song song, ảnh chụp màn hình, tài liệu

Mô tả được tạo tự động

* **Command Layer (Pattern-Specific Layer):**

**Mục đích:** Đóng gói các hành động và logic nghiệp vụ riêng biệt thành từng lệnh

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Song song

Mô tả được tạo tự động

* **Utilities Layer:**

**Mục đích:** Chứa các lớp tiện ích hoặc cấu trúc chung không thuộc về logic nghiệp vụ cụ thể nào.

Ảnh có chứa văn bản, ảnh chụp màn hình, Song song, Phông chữ

Mô tả được tạo tự động

# **3. Phân tích và mô tả hệ thống**

**Danh sách các package trong hệ thống quản lý phòng gym:**

**3.1. Package model**

* **Chứa các lớp đại diện cho các đối tượng cốt lõi trong hệ thống.**
  + **User**
    - Đại diện cho người dùng trong hệ thống (Admin, Trainer, Member).
    - Chứa các thông tin cơ bản như username, password, role.
  + **Member, Trainer, Admin**
    - Kế thừa từ lớp User, bổ sung các thuộc tính riêng như membershipCard (Member), salary (Trainer).
  + **Equipment**
    - Đại diện cho thiết bị trong phòng gym.
    - Các thuộc tính gồm tên thiết bị, trạng thái, giá, vị trí.
  + **Course**
    - Thông tin khóa học: tên, mô tả, giá tiền, số lượng học viên tối đa.
  + **GymClass**
    - Đại diện cho lớp học, liên kết với Course.
    - Các thuộc tính như lịch học, số lượng học viên tối đa.
  + **MembershipCard**
    - Chứa thông tin thẻ thành viên: loại thẻ, ngày bắt đầu, ngày kết thúc.
  + **Payment**
    - Lưu trữ thông tin thanh toán: người thanh toán, số tiền, ngày thanh toán, mục đích.

**3.2 Package service**

* **Chứa các lớp xử lý logic nghiệp vụ của hệ thống.**
  + **UserService**
    - Quản lý thông tin người dùng (CRUD).
    - Bao gồm tìm kiếm, thêm mới, cập nhật, và xóa người dùng.
  + **MembershipService**
    - Xử lý nghiệp vụ liên quan đến thẻ thành viên.
    - Quản lý việc đăng ký mới hoặc kiểm tra tính hợp lệ của thẻ.
  + **CourseService**
    - Quản lý khóa học: thêm, sửa, xóa thông tin khóa học.
    - Lấy danh sách tất cả khóa học.
  + **ClassService**
    - Quản lý lớp học: thêm, sửa, xóa lớp học.
    - Kiểm tra danh sách học viên đã đăng ký lớp học.
  + **EquipmentService**
    - Quản lý thiết bị: thêm mới, sửa thông tin, xóa thiết bị.
    - Thống kê số lượng thiết bị hiện có.
  + **PaymentService**
    - Xử lý thanh toán: thêm giao dịch thanh toán, xem lịch sử thanh toán.

**3.3 Package command**

* **Chứa các lớp Command thực hiện hành động cụ thể.**
  + **MemberCommand**
    - Các lệnh dành cho Member:
      * Đăng ký lớp học (RegisterForClassCommand).
      * Xem thông tin lớp học (ViewGymClassFromCourseCommand).
      * Xem lịch sử thanh toán (ViewPaymentHistoryCommand).
  + **AdminCommand**
    - Các lệnh dành cho Admin:
      * Quản lý thiết bị (AddEquipmentCommand, UpdateEquipmentCommand).
      * Quản lý khóa học (CreateCourseCommand, DeleteCourseCommand).
      * Thống kê dữ liệu (GetStatisticsCommand).
  + **TrainerCommand**
    - Các lệnh dành cho Trainer:
      * Xem lớp học của mình (ViewTrainerClassCommand).
      * Xem thanh toán nhận được (ViewReceivedPaymentsCommand).

**3.4 Package utils**

* **Chứa các tiện ích hỗ trợ hệ thống.**
  + **ServiceContainer**
    - Quản lý các service trong hệ thống theo Singleton Pattern.
    - Đảm bảo chỉ khởi tạo một lần và được dùng lại xuyên suốt ứng dụng.
  + **SessionManager**
    - Quản lý trạng thái người dùng hiện tại.
    - Đảm bảo mỗi phiên đăng nhập chỉ gắn với một người dùng cụ thể.
  + **CommandFactory**
    - Tạo ra các Command tương ứng cho từng role (Admin, Member, Trainer).
    - Đảm bảo mở rộng dễ dàng khi thêm chức năng mới.

**3.5. Package constants**

* **Chứa các hằng số dùng chung trong hệ thống.**
  + **RoleType**
    - Xác định vai trò: MEMBER, TRAINER, ADMIN.
  + **MembershipCardType**
    - Loại thẻ thành viên: BASIC, PREMIUM.
  + **MemberMenuConstants**
    - Các ID ánh xạ với các lệnh của Member như VIEW\_PROFILE, REGISTER\_CLASS.

**3.6. Package interfaces**

* **Chứa các interface định nghĩa hành động và nghiệp vụ.**
  + **ActionCommand**
    - Định nghĩa interface cho mọi lệnh (Command) trong hệ thống.
  + **UserService, MembershipService, ...**
    - Định nghĩa các chức năng cơ bản cần thiết cho từng thực thể.
    - Giúp che giấu logic triển khai cụ thể và tăng khả năng mở rộng.

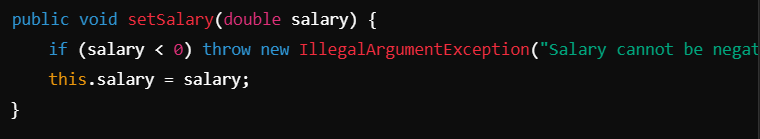
**3.7. Package controller**

* **Chứa các lớp điều khiển logic luồng xử lý của hệ thống, liên kết giữa service, command, và giao diện.**
* **AdminDashboardController**
  + Điều phối các hành động của Admin.
  + Chức năng bao gồm:
    - Quản lý thiết bị: thêm mới, cập nhật, xóa.
    - Quản lý người dùng (Member, Trainer).
    - Quản lý khóa học và lớp học.
    - Xem báo cáo thống kê.
* **MemberDashboardController**
  + Điều phối các hành động của Member.
  + Chức năng bao gồm:
    - Đăng ký lớp học.
    - Xem thông tin khóa học, lịch sử thanh toán, Membership.
    - Cập nhật hồ sơ cá nhân.
* **TrainerDashboardController**
  + Điều phối các hành động của Trainer.
  + Chức năng bao gồm:
    - Quản lý lớp học của mình.
    - Xem danh sách học viên.
    - Nhận thanh toán.
* **AuthPortalController**
  + Điều phối các hành động liên quan đến tạo tài khoản, đăng nhập, đăng ký .
  + Chức năng bao gồm:
    - Đăng ký, đăng nhập tài khoản.
    - Đăng xuất tài khoản.

# **4. Kỹ thuật lập trình hướng đối tượng đã áp dụng**

**4.1. Tính đóng gói:**

* Đóng gói bảo vệ dữ liệu bằng cách che giấu chi tiết và chỉ cho phép truy cập thông qua các phương thức cụ thể.

****

**Ứng dụng trong dự án:**

* Các thuộc tính của User, Course, Equipment được khai báo private, chỉ có thể truy cập thông qua getter/setter.

→ Bảo vệ dữ liệu: Trainer không thể có lương âm.

**Service Interface:** Các service như UserService chỉ định nghĩa API (addUser, updateUser) mà không tiết lộ chi tiết dữ liệu bên trong (UserServiceImpl lưu trong bộ nhớ hoặc cơ sở dữ liệu).

**4.2. Tính kế thừa:**

* Tái sử dụng code qua việc chia sẻ các thuộc tính và hành vi chung trong lớp cha.

**Ứng dụng trong dự án:**

* User là lớp cha của Member, Trainer, Admin.
* Các thuộc tính chung (username, password, role,…) được định nghĩa trong User, các lớp con chỉ thêm logic riêng.

A screen shot of a computer program

Description automatically generated

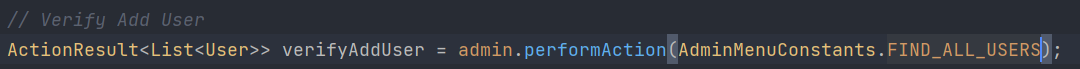
* Kế thừa giúp giảm lặp lại code, mở rộng dễ dàng (ví dụ, thêm role Manager).

**4.3. Tính đa hình:**

* Cho phép cùng một hành vi được thực hiện khác nhau tùy theo ngữ cảnh hoặc đối tượng.

**Ứng dụng trong dự án:**

* Phương thức performAction() linh hoạt xử lý cho các role khác nhau (Admin, Member, Trainer):



→ Nếu user là Admin, actionId = 1 sẽ gọi AddEquipmentCommand.

→ Nếu user là Member, actionId = 1 sẽ gọi ViewProfileCommand.

* **Lợi ích**: Hệ thống dễ mở rộng mà không làm thay đổi các hành vi cũ.

**4.4. Tính trừu tượng:**

* Tập trung vào "cái gì cần làm", không phải "làm như thế nào".

**Ứng dụng trong dự án:**

* Các **Service Interface** như UserService định nghĩa các hành động cần thiết (addUser, updateUser), chi tiết xử lý được giấu trong UserServiceImpl.



→ Các layer khác chỉ cần gọi userService.addUser(user) mà không cần quan tâm dữ liệu được lưu ở đâu.

* Tương lai dễ mở rộng: Nếu lưu trữ dữ liệu chuyển từ bộ nhớ sang SQL, chỉ cần thay UserServiceImpl bằng UserServiceSQLImpl. Các thành phần khác không cần thay đổi.

**4.5. Tối ưu hệ thống với Command Design Pattern:**

**4.5.1. Phân tích bài toán và yêu cầu**

* **3 role với các chức năng khác nhau:**
  + **Admin:** Quản lý thiết bị, người dùng (CRUD), khóa học và lớp học, thống kê dữ liệu.
  + **Trainer:** Hướng dẫn các lớp học, quản lý danh sách học viên.
  + **Member:** Đăng ký membership, tham gia lớp học, xem thông tin cá nhân.
* **Tính chất phân quyền:** Mỗi role chỉ được phép thực hiện các hành động cụ thể, không được truy cập chức năng của role khác.
* **Hệ thống cần mở rộng dễ dàng:** Bổ sung các chức năng mới cho từng role mà không làm thay đổi logic cốt lõi.
* **Tính tái sử dụng:** Nhiều chức năng chung (như CRUD) nhưng thực hiện trên các thực thể khác nhau (User, Equipment, Course...).

**Kết luận:** Bài toán này đòi hỏi một cách tiếp cận có khả năng mở rộng, dễ bảo trì và hỗ trợ phân quyền. Command Pattern là giải pháp hoàn hảo.

**4.5.2. Tại sao chọn Command Pattern?**

* **Đáp ứng tính năng phân quyền:**
  + Mỗi role có một tập lệnh riêng (MemberCommand, AdminCommand, TrainerCommand).
  + Các command được đăng ký vào MenuOption tương ứng, đảm bảo rằng mỗi role chỉ thấy được các chức năng mà họ được phép thực hiện.
* **Dễ dàng mở rộng:** Thêm chức năng mới chỉ cần:
  + Tạo một class Command mới (tuân theo giao diện ActionCommand).
  + Đăng ký lệnh đó vào CommandFactory.
  + **Ví dụ:** Khi bổ sung chức năng "Xem danh sách thiết bị," chỉ cần thêm ViewAllEquipmentCommand và không phải thay đổi logic của hệ thống.
* **Độc lập và tái sử dụng:** Các command được tách biệt, dễ kiểm thử và tái sử dụng trong các ngữ cảnh khác nhau.
* **Hỗ trợ kiểm soát truy cập:** Role-based access control được triển khai một cách tự nhiên thông qua cấu trúc CommandFactory.

**4.5.3. Lợi ích nổi bật khi dùng Command Pattern**

Liệt kê các lợi ích cụ thể áp dụng trong bài toán:

* **Phân tách rõ ràng giữa logic xử lý và giao diện người dùng (Separation of Concerns):**
  + Logic của mỗi lệnh (như thêm thiết bị, cập nhật người dùng) được gói gọn trong các class riêng.
  + Giao diện chính chỉ cần gọi các lệnh mà không cần biết chi tiết bên trong.
* **Dễ bảo trì và mở rộng:**
  + Thêm role mới (ví dụ, "Manager") chỉ cần thêm một CommandFactory mới.
  + Thêm chức năng mới không ảnh hưởng đến chức năng cũ.
* **Giảm sự phụ thuộc giữa các thành phần:**
  + Mã nguồn chính không cần biết chi tiết về UserService, EquipmentService, hoặc bất kỳ service nào khác. Tất cả đều được đóng gói trong các command.

# **5. Demo các chức năng của chương trình**

**5.1. AuthPortal:** đăng nhập, đăng ký

**A screenshot of a login screen

Description automatically generatedA screenshot of a login

Description automatically generatedA screenshot of a login screen

Description automatically generated**

**5.2. MemberDashboard**

- Hiển thị thẻ thành viên, chức năng add thẻ và gia hạn thẻ:

A screenshot of a membership card

Description automatically generated

- Hiển thị giao dịch và các lớp đã đăng ký:

A screenshot of a computer

Description automatically generated

- Đăng ký lớp, tập với PT:

A screenshot of a computer

Description automatically generated

- Cập nhập thông tin cá nhân:

A screenshot of a computer

Description automatically generated

**5.3. TrainerDashboard**

- Hiển thị các lớp phụ trách, thông tin cơ bản

A screenshot of a computer

Description automatically generated

- Đăng ký đảm nhiệm lớp:

A screenshot of a computer

Description automatically generated

**5.4. Giao diện Admin:**

- Quản lý User:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

- Quản lý Course, GymClass

A screenshot of a computer

Description automatically generated

- Quản lý Thiết bị:

A screenshot of a computer

Description automatically generated

# **6. Kết luận**

Dự án quản lý phòng Gym đã cung cấp một giải pháp hiệu quả và toàn diện, đáp ứng nhu cầu quản lý hoạt động tại phòng tập. Với các tính năng như quản lý hội viên, theo dõi lịch tập, quản lý thiết bị và nhân sự, hệ thống đã góp phần tối ưu hóa quy trình vận hành, tiết kiệm thời gian và nâng cao trải nghiệm người dùng.

Sự thành công của dự án không chỉ đến từ việc áp dụng các công nghệ tiên tiến mà còn từ sự phối hợp chặt chẽ giữa các thành viên trong nhóm. Đây sẽ là nền tảng vững chắc để tiếp tục phát triển và mở rộng hệ thống trong tương lai, đáp ứng nhu cầu ngày càng cao của ngành dịch vụ thể dục thể thao.