HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

HỆ THỐNG QUẢN LÝ KHÁCH SẠN

Đồ án Lập trình Hướng đối tượng - Nhóm 204

Người thực hiện:

Trần Đình Khánh - 20237449
Nguyễn Hữu Linh - 20237455

ONE LOVE. ONE FUTURE.

Nội dung trình bày

1. Giới thiệu bài toán
2. Phân tích yêu cầu
3. Thiết kế hệ thống
4. Triển khai
5. Kết luận

1.1 Bối cảnh và bài toán

Bối cảnh thực tế:

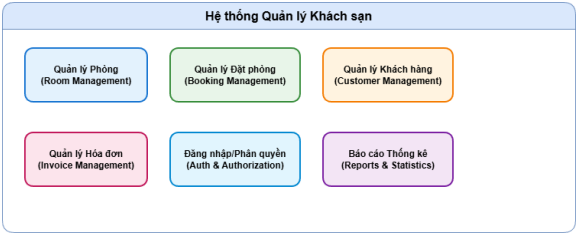
- Ngành khách sạn cần hệ thống phần mềm quản lý **chuyên nghiệp**
- Quản lý thủ công gây **sai sót**, **chậm trễ** và khó mở rộng
- Yêu cầu: Đặt phòng, Check-in/out, Hóa đơn, Báo cáo doanh thu

Mục tiêu dự án:

- Xây dựng hệ thống **CRUD** hoàn chỉnh cho Room, Booking, Customer, Invoice
- Áp dụng **4 tính chất OOP**: Encapsulation, Inheritance, Polymorphism, Abstraction
- Công nghệ: **Java 21**, Swing + FlatLaf, JSON Storage, Maven

6 Module chức năng:

1. Quản lý Phòng
2. Quản lý Đặt phòng
3. Quản lý Khách hàng
4. Quản lý Hóa đơn
5. Đăng nhập/Phân quyền
6. Báo cáo Thống kê



Hình: Sơ đồ phạm vi hệ thống

Công nghệ:

- Java 21, Swing + FlatLaf
- JSON Storage (Gson)
- Maven Build

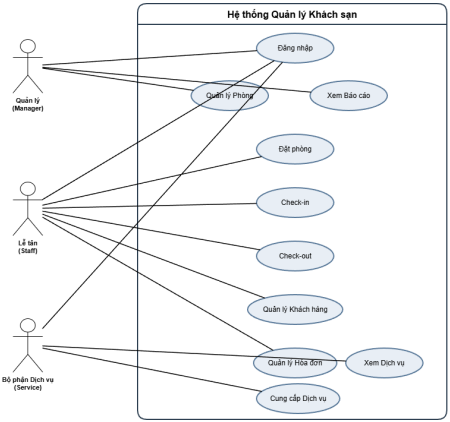
Hệ thống sử dụng 3 Role nghiệp vụ độc lập (không có kế thừa Actor)

Actor	**Role trong code**	**Chức năng chính**
Quản lý (Manager)	`MANAGER`	Quản lý phòng, xem báo cáo
Lễ tân (Staff)	`STAFF`	Đặt phòng, check-in/out, hóa đơn
Bộ phận Dịch vụ (Service)	`SERVICE`	Cung cấp dịch vụ khách hàng

Tại sao không dùng kế thừa Actor?

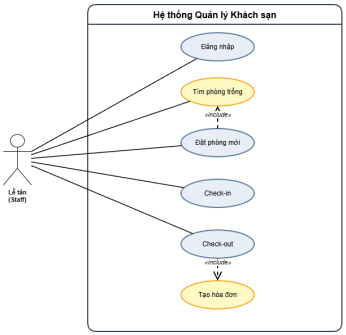
- Hệ thống dùng **Permission-based Access Control**
- Mỗi Role có tập quyền riêng biệt trong `PermissionManager.java`
- Linh hoạt hơn: Dễ thêm/bớt quyền mà không cần sửa cấu trúc

2.2 Use Case tổng quan



Hình: Use Case Diagram tổng quan - 3 Actor nghiệp vụ

2.3 Use Case chi tiết - Lễ tân (Staff)



Hình: Use Case Lễ tân

6 Use Case chính:

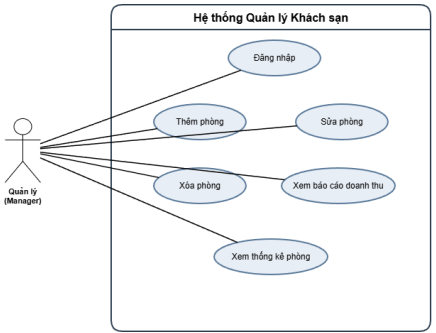
1. **Đăng nhập**: Xác thực qua AuthService
2. **Tìm phòng trống**: Lọc theo ngày
3. **Đặt phòng mới**: Tạo Booking
4. **Check-in**: Đổi status CHECKED_IN
5. **Check-out**: Đổi status COMPLETED
6. **Tạo hóa đơn**: Sinh Invoice

Quan hệ «include»:

- "Đặt phòng" → "Tìm phòng trống"
- "Check-out" → "Tạo hóa đơn"

Include = bắt buộc phải thực hiện

2.4 Use Case chi tiết - Quản lý (Manager)



Hình: Use Case Quản lý

6 Use Case chính:

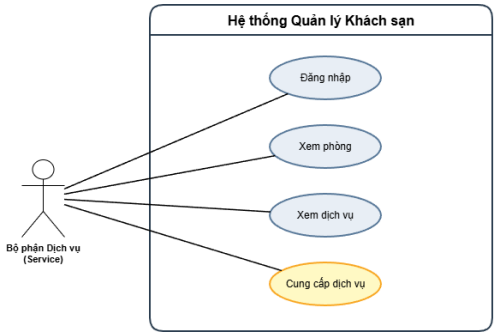
1. **Đăng nhập**: Xác thực quyền MANAGER
2. **Thêm/Sửa/Xóa phòng**: CRUD Room
3. **Xem báo cáo doanh thu**: Dashboard
4. **Xem thống kê phòng**: Biểu đồ

Quyền đặc biệt (PermissionManager):

- `MANAGE_ROOMS` - CRUD phòng
- `VIEW_REPORTS` - Xem báo cáo
- `EXPORT_REPORTS` - Xuất báo cáo
- `MANAGE_SERVICES` - Quản lý DV

Manager có gần như full quyền, chỉ thiếu MANAGE_ACCOUNTS

2.5 Use Case chi tiết - Bộ phận Dịch vụ (Service)



Hình: Use Case Bộ phận Dịch vụ

4 Use Case chính:

1. **Đăng nhập**: Xác thực quyền SERVICE
2. **Xem phòng**: Kiểm tra trạng thái
3. **Xem dịch vụ**: Danh sách dịch vụ
4. **Cung cấp dịch vụ**: Phục vụ khách

Quyền hạn (PermissionManager):

- `VIEW_ROOMS` - Chỉ xem
- `VIEW_SERVICES` - Xem dịch vụ
- `PROVIDE_SERVICES` - Cung cấp DV
- `VIEW_DASHBOARD` - Xem tổng quan

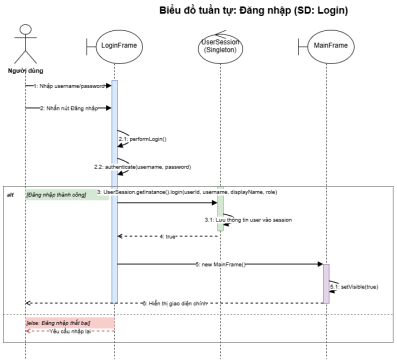
Service không có quyền thêm/sửa/xóa, chỉ phục vụ khách

2.6 Đặc tả Use Case "Đăng nhập"

Use Case chung cho tất cả Actor

Tác nhân	Tất cả (Manager, Staff, Service)
Mô tả	Xác thực người dùng để vào hệ thống
Tiền điều kiện	Có tài khoản trong hệ thống
Luồng chính	1. Mở LoginFrame 2. Nhập username và password 3. AuthService kiểm tra thông tin 4. Lưu User vào UserSession (Singleton) 5. Chuyển sang MainFrame
Luồng phụ	Sai thông tin $\rightarrow$ Hiển thị lỗi, cho nhập lại
Hậu điều kiện	UserSession chứa User đang đăng nhập

2.7 Sequence Diagram - Đăng nhập



Hình: Biểu đồ tuần tự "Đăng nhập"

Các bước thực hiện:

1. User nhập username/password
2. Nhấn nút Đăng nhập
3. LoginFrame.performLogin()
4. authenticate() kiểm tra credentials
5. UserSession.login() lưu thông tin
6. Mở MainFrame

Class liên quan:

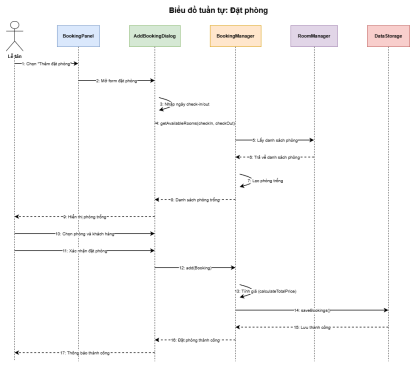
- `LoginFrame`
- `UserSession` (Singleton)
- `MainFrame`

2.8 Đặc tả Use Case "Đặt phòng"

Use Case quan trọng nhất - Nghiệp vụ cốt lõi của hệ thống

Tác nhân	Lễ tân (Staff)
Mô tả	Tạo booking mới cho khách hàng, hệ thống tự động tính giá
Tiền điều kiện	Đã đăng nhập với quyền CREATE_BOOKING
Luồng chính	1. Lễ tân mở AddBookingDialog 2. Chọn ngày check-in/check-out 3. Hệ thống gọi `getAvailableRooms()` 4. Chọn khách hàng từ CustomerManager 5. Chọn phòng và xác nhận 6. Hệ thống tạo Booking với status PENDING
Luồng phụ	Không có phòng trống → Thông báo lỗi
Hậu điều kiện	Booking được lưu, Room status = OCCUPIED

2.9 Sequence Diagram - Đặt phòng



Hình: Biểu đồ tuần tự "Đặt phòng"

Luồng xử lý:

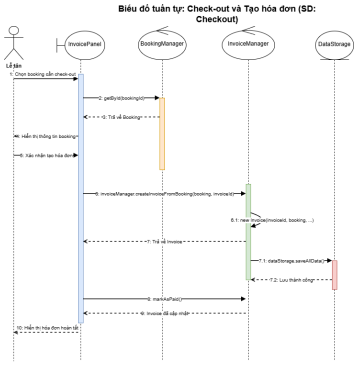
1. Lễ tân chọn "Thêm đặt phòng"
2. openAddDialog() mở form
3. Nhập ngày, refreshAvailableRooms()
4. bookingManager.isRoomAvailable(
5. Chọn Customer + Room
6. saveBooking() → add(booking)
7. DataStorage.saveAllData()

2.10 Đặc tả Use Case "Check-out và Tạo hóa đơn"

Use Case kết hợp Check-out và Tạo hóa đơn

Tác nhân	Lễ tân (Staff)
Mô tả	Hoàn tất booking và tạo hóa đơn thanh toán
Tiền điều kiện	Đã đăng nhập, Booking có status CHECKED_IN
Luồng chính	1. Lễ tân chọn booking cần check-out 2. Xác nhận check-out 3. Hệ thống đổi status = COMPLETED 4. «include» Tạo hóa đơn tự động 5. Tính tiền phòng + thuế 6. Hiển thị Invoice
Luồng phụ	Booking không hợp lệ → Thông báo lỗi
Hậu điều kiện	Booking = COMPLETED, Invoice được tạo

2.11 Sequence Diagram - Check-out và Tạo hóa đơn



Hình: Biểu đồ tuần tự "Check-out và Tạo hóa đơn"

Luồng xử lý:

1. Lễ tân chọn booking từ InvoicePanel
2. getById() lấy thông tin booking
3. Xác nhận tạo hóa đơn
4. createInvoiceFromBooking()
5. markAsPaid()
6. saveAllData() lưu dữ liệu

3.1 Kiến trúc MVC - 4 tầng

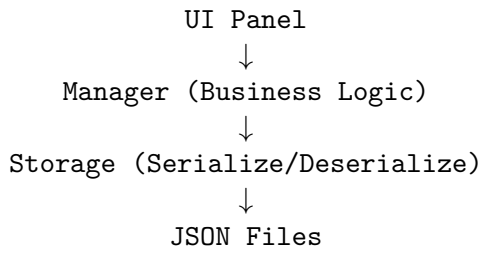
Tại sao dùng kiến trúc MVC?

- **Tách biệt** View - Controller - Model
- **Dễ bảo trì**: Sửa UI không ảnh hưởng logic
- **Dễ test**: Mock Storage để test Manager

4 Layer:

1. **UI Layer** (Swing Panels)
2. **Service Layer** (Managers)
3. **Model Layer** (Entities)
4. **Storage Layer** (JSON)

Luồng dữ liệu



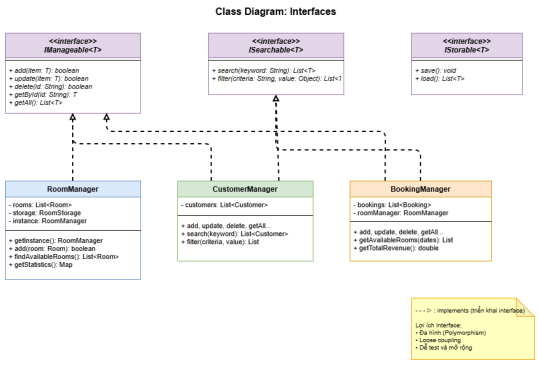
Manager không biết dữ liệu lưu ở đâu → Có thể đổi sang Database sau!

Tại sao dùng Interface?

- `IManageable<T>`: CRUD chung
- `ISearchable<T>`: Tìm kiếm
- `IStorable<T>`: Lưu/tải

Lợi ích (Abstraction):

- Định nghĩa "hợp đồng"
- Dễ mở rộng Manager mới
- Hỗ trợ Dependency Injection



Hình: Class Diagram - Interfaces

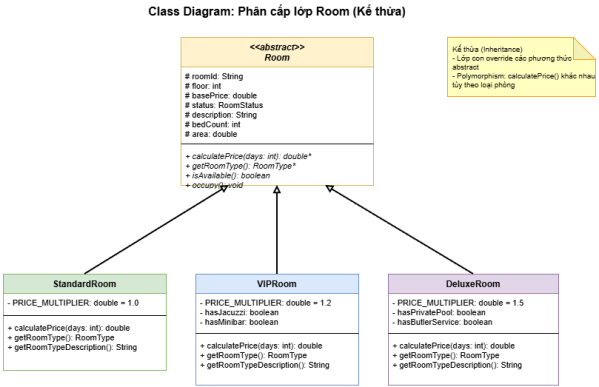
3.3 Class Diagram - Room (Inheritance)

OOP - Inheritance:

- `Room` là abstract class
- 3 subclass: Standard, VIP, Deluxe

OOP - Polymorphism:

- Override `calculatePrice()`
- Standard: $\times 1.0$
- VIP: $\times 1.2$
- Deluxe: $\times 1.5$



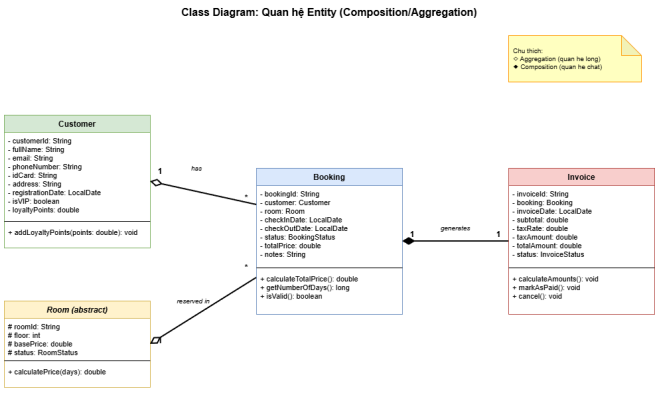
Hình: Abstract Room và 3 subclass

Quan hệ Composition:

- Booking chứa Customer + Room
- Invoice chứa Booking
- Khi xóa Booking → Invoice bị ảnh hưởng

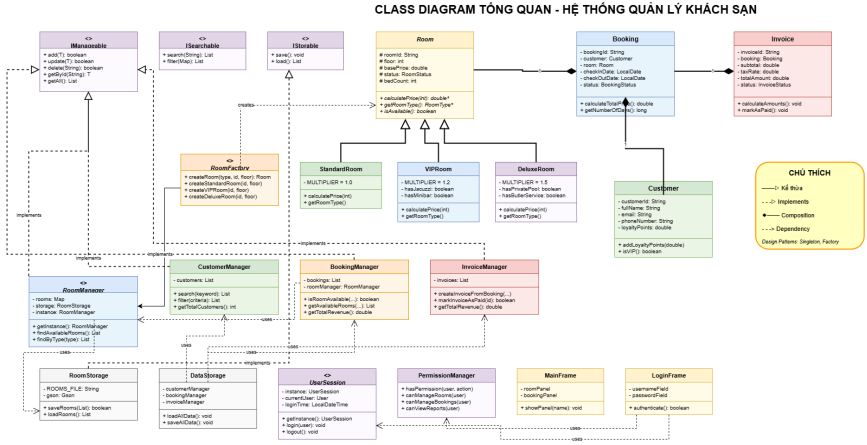
OOP - Encapsulation:

- Thuộc tính `private`
- Truy cập qua getter/setter



Hình: Quan hệ Composition giữa Entity

3.5 Class Diagram - Tổng thể



Hình: Class Diagram tổng thể hệ thống

3.6 Design Patterns sử dụng

Factory Pattern

`RoomFactory`

- Tạo Room theo type
- Ẩn logic khởi tạo
- Dễ thêm loại mới

Singleton Pattern

`UserSession`

- 1 instance duy nhất
- Lưu thông tin login
- Truy cập toàn cục

MVC Pattern

Toàn bộ kiến trúc

- View: UI Panels
- Controller: Managers
- Model: Entities

Tại sao dùng Design Patterns? → Code dễ đọc và bảo trì

Cấu trúc thư mục:

- `src/com/hotel/`
  - `model/` - 11 files
  - `service/` - 7 files
  - `storage/` - 2 files
  - `ui/` - 17 files
  - `auth/` - 2 files
  - `util/` - 2 files

Tổng cộng: ∼41 files, ∼4500 LOC

Data Files (JSON)

- `rooms.json` - Danh sách phòng
- `customers.json` - Khách hàng
- `bookings.json` - Đặt phòng
- `invoices.json` - Hóa đơn
- `users.json` - Tài khoản

Tại sao dùng JSON?

- Đơn giản, không cần DB server
- Human-readable, dễ debug
- Portable, đi kèm project

4.2 Bốn nguyên lý OOP áp dụng

1. Encapsulation (Đóng gói)

- Thuộc tính `private`
- Truy cập qua getter/setter
- Ví dụ: `Room.basePrice`

2. Inheritance (Kế thừa)

- `Room` → `StandardRoom`
- `Room` → `VIPRoom`
- `Room` → `DeluxeRoom`

3. Polymorphism (Đa hình)

- `calculatePrice()` trả về giá khác nhau
- VIPRoom: $\times 1.2$
- DeluxeRoom: $\times 1.5$

4. Abstraction (Trừu tượng)

- Abstract class: `Room`
- Interface: `IManageable<T>`
- Interface: `ISearchable<T>`

5.1 Tổng kết

Đã hoàn thành:

- ✓ Hệ thống CRUD hoàn chỉnh
- ✓ 4 tính chất OOP
- ✓ 3 Design Patterns
- ✓ Kiến trúc MVC 4 tầng
- ✓ Phân quyền Permission-based
- ✓ UI hiện đại (FlatLaf)

Hướng phát triển:

- ▶ Chuyển sang Database thực
- ▶ Thêm module Dịch vụ phòng
- ▶ Tích hợp thanh toán online
- ▶ Responsive Web UI

Demo

DEMO ỨNG DỤNG

THÔNG TIN LIÊN HỆ

Nhóm 204:
- Trần Đình Khánh - 20237449
- Nguyễn Hữu Linh - 20237455

- **GitHub:** `https://github.com/IvanLyVodka11/hotel-management`

Xin chân thành cảm ơn Thầy/Cô và các bạn đã lắng nghe!

CẢM ƠN QUÝ THẦY CÔ!

Mời Thầy/Cô đặt câu hỏi

THANK YOU !