

# **NHẬP MÔN CÔNG NGHỆ PHẦN MỀM**

**TÀI LIỆU**

**DÀNH CHO SINH VIÊN CÔNG NGHỆ THÔNG TIN**

**TRẦN ĐÌNH QUẾ  
NGUYỄN MẠNH HÙNG**

## GIỚI THIỆU

Mục tiêu của môn Công nghệ phần mềm là cung cấp cho sinh viên những kiến thức cơ bản về tất cả mọi hoạt động liên quan đến phát triển phần mềm và kiến thức cơ bản về UML trong phát triển phần mềm. Qua môn học này sinh viên có kỹ năng sử dụng công cụ phần mềm để thực hiện các pha trong quá trình phát triển phần mềm và qua đó nâng cao năng lực làm việc nhóm và kỹ năng mềm. Sinh viên tham dự lớp và thực hành đầy đủ đặc biệt tích cực tham gia thảo luận trình bày trên lớp là yêu cầu quan trọng.

Nội dung bao gồm các kiểu hệ thống thông tin, các mô hình phát triển phần mềm, lập kế hoạch và quản lý dự án; các pha phát triển phần mềm từ xác định yêu cầu, phân tích, thiết kế đến lập trình – tích hợp; các kiến thức cơ bản về mô hình phần mềm với UML.

**Mở đầu:** Các đặc trưng của phần mềm; Các dạng phần mềm; Các hoạt động trong phát triển phần mềm; Tiến hóa trong phát triển phần mềm

### **Chương 2: Các pha trong phát triển phần mềm.**

Các tác nhân trong quá trình phát triển phần mềm; Pha xác định yêu cầu; Pha phân tích; Pha thiết kế; Pha cài đặt và tích hợp; Pha bảo trì

### **Chương 3: Các mô hình vòng đời phần mềm**

Mô hình xây - sửa; Mô hình thác nước; Mô hình bản mẫu nhanh; Mô hình tiến hoá; Mô hình RUP; Mô hình xoắn ốc; So sánh các mô hình

### **Chương 4: Kiểm chứng**

Vấn đề chất lượng phần mềm; Kiểm chứng phần mềm; Các phương pháp kiểm chứng; Công cụ kiểm chứng

### **Chương 5: Lập kế hoạch và ước lượng**

Vấn đề lập kế hoạch và ước lượng dự án phần mềm; Ước lượng thời gian và chi phí; Các thành phần của việc lập kế hoạch dự án phần mềm; Lập kế hoạch cho các dự án phần mềm hướng đối tượng.

### **Chương 6: Xác định yêu cầu**

Các kỹ thuật xác định yêu cầu; Bản mẫu nhanh; Đặc tả dựa trên bản mẫu nhanh; Sử dụng lại bản mẫu; Đặc tả với bản mẫu; Kiểm thử pha yêu cầu

### **Chương 7: Các phương pháp phân tích truyền thống**

Viết tài liệu pha đặc tả; Đặc tả phi hình thức; Các kỹ thuật đặc tả nửa hình thức; Mô hình quan hệ thực thể; Máy trạng thái hữu hạn; Các kỹ thuật đặc tả hình thức; So sánh các kỹ thuật đặc tả; Kiểm thử pha đặc tả

### **Chương 8: Phân tích hướng đối tượng**

Tổng quan về phân tích hướng đối tượng; Mô hình use case; Mô hình lớp; Mô hình hành động; Kiểm thử pha phân tích hướng đối tượng

### **Chương 9: Thiết kế**

Tổng quan về pha thiết kế; Thiết kế hướng hành động; Phân tích dòng dữ liệu; Thiết kế hướng đối tượng

### **Chương 10: Cài đặt và tích hợp**

Các phương pháp cài đặt và tích hợp; Kiểm thử pha cài đặt và tích hợp; Kiểm thử sản phẩm; Kiểm thử chấp nhận

### **Chương 11: Bảo trì**

Pha bảo trì; Bảo trì hệ phần mềm hướng đối tượng

## **TÀI LIỆU THAM KHẢO**

- [1] Object-Oriented and Classical Software Engineering, Stephen R. Schach, Eighth Edition, McGraw Hill, 2010.
- [2] Giáo trình nhập môn UML, Huỳnh Văn Đức, Đoàn Thiện Ngân, NXB Lao động Xã hội, 2003.



## MỤC LỤC

<b>MỤC LỤC.....</b>	<b>4</b>
<b>CHƯƠNG 1: MỞ ĐẦU.....</b>	<b>9</b>
1.2 CÁC KIỀU PHẦN MỀM.....	9
1.3 KHÍA CẠNH LỊCH SỬ.....	9
1.4 KHÍA CẠNH KINH TẾ.....	10
1.5 KHÍA CẠNH BẢO TRÌ.....	10
1.6 KHÍA CẠNH PHÂN TÍCH VÀ THIẾT KẾ.....	11
1.7 KHÍA CẠNH LẬP TRÌNH NHÓM.....	12
1.8 PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG.....	12
1.9 CÂU HỎI ÔN TẬP.....	14
<b>CHƯƠNG 2: CÁC PHA PHÁT TRIỂN PHẦN MỀM.....</b>	<b>16</b>
2.1 TIẾN TRÌNH THÀNH PHẦN.....	16
2.2 SQA LÀ GÌ?.....	17
2.3 PHA YÊU CẦU.....	17
2.4 PHA ĐẶC TẨY.....	17
2.5 PHA THIẾT KẾ.....	18
2.6 PHA CÀI ĐẶT.....	18
2.7 TÍCH HỢP.....	19
2.8 CẢI TIẾN TIẾN TRÌNH PHẦN MỀM.....	19
2.9 CÂU HỎI ÔN TẬP.....	22
<b>CHƯƠNG 3.....</b>	<b>24</b>
<b>CÁC MÔ HÌNH VÒNG ĐỜI PHẦN MỀM.....</b>	<b>24</b>
3.1 PHÁT TRIỂN PHẦN MỀM.....	24
3.1.1 Theo lý thuyết phát triển phần mềm: .....	24
3.1.2 Thực tế phát triển phần mềm.....	24
3.1.3 Bài toán Winburg Mini.....	24
3.2 MÔ HÌNH XÂY SỬA.....	25
3.3 MÔ HÌNH TIẾN HÓA.....	25
3.4 MÔ HÌNH BẢN MÃ NHANH.....	26
3.5. MÔ HÌNH LẶP VÀ TĂNG.....	27
3.6 MÔ HÌNH UP.....	31
3.7 MÔ HÌNH XOĂN ÓC.....	36
3.8 MÔ HÌNH MÃ NGUỒN MỎ.....	37
3.9 CÂU HỎI ÔN TẬP.....	39
<b>CHƯƠNG 4: KIỂM THỬ.....</b>	<b>41</b>
4.1 VẤN ĐỀ CHẤT LƯỢNG PHẦN MỀM.....	41
4.1.1 Đảm bảo chất lượng phần mềm (SQA).....	41
4.1.2. Độc lập trong quản lý.....	41
4.2 KIỂM CHỨNG PHẦN MỀM.....	41
4.3 CÁC PHƯƠNG PHÁP KIỂM CHỨNG.....	42
4.3.1. Kiểm thử không có sự thực thi.....	42
4.3.2 Kiểm thử có dựa trên sự thực thi.....	46
4.4 NHỮNG VẤN ĐỀ TRONG KIỂM THỬ.....	46
4.4.1 Cái gì nên được kiểm thử?.....	46

4.4.2 Kiểm thử và sự kiểm chứng tính chính xác.....	48
4.4.3 Sự kiểm chứng tính chính xác và kỹ nghệ phần mềm.....	49
4.4.4 Ai thực hiện kiểm thử phần mềm.....	50
4.4.5 Khi nào kiểm thử dừng.....	51
4.5 CÂU HỎI ÔN TẬP.....	51
<b>CHƯƠNG 5: LẬP KẾ HOẠCH VÀ ƯỚC LUỢNG.....</b>	<b>52</b>
5.1 VẤN ĐỀ LẬP KẾ HOẠCH VÀ ƯỚC LUỢNG DỰ ÁN PHẦN MỀM.....	52
5.2 ƯỚC LUỢNG THỜI GIAN VÀ CHI PHÍ.....	53
5.2.1 Thước đo kích cỡ của sản phẩm phần mềm.....	53
5.2.2 Các kỹ thuật ước lượng chi phí.....	56
5.2.3 COCOMO trung gian.....	57
5.2.4 COCOMO II.....	59
5.2.5 Theo dõi ước lượng thời gian và chi phí.....	60
5.3 CÁC THÀNH PHẦN CỦA VIỆC LẬP KẾ HOẠCH PHÁT TRIỂN PHẦN MỀM.....	62
5.3.1 Khung kế hoạch quản lý dự án phần mềm(SPMP).....	62
5.3.2 Kế hoạch quản lý dự án phần mềm IEEE.....	62
5.3.3 Việc lập kế hoạch kiểm thử.....	63
5.3.4 Yêu cầu đào tạo.....	63
5.3.5 Các chuẩn tài liệu.....	63
5.3.6 Công cụ CASE cho việc lập kế hoạch và ước lượng.....	64
5.4 KIỂM THỬ VIỆC LẬP KẾ HOẠCH.....	64
5.5 LẬP KẾ HOẠCH CHO CÁC DỰ ÁN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG.....	64
5.6 CÂU HỎI ÔN TẬP.....	64
<b>CHƯƠNG 6: PHÁ XÁC ĐỊNH YÊU CẦU.....</b>	<b>66</b>
6.1 XÁC ĐỊNH YÊU CẦU CỦA KHÁCH HÀNG.....	66
6.2 TỔNG QUAN VỀ LUỒNG CÔNG VIỆC XÁC ĐỊNH YÊU CẦU.....	66
6.2.1 Hiểu lĩnh vực ứng dụng.....	67
6.2.2 Mô hình nghiệp vụ.....	67
6.2.3 Các use case.....	68
6.2.4 Các yêu cầu ban đầu.....	70
6.3 PHÁ XÁC ĐỊNH YÊU CẦU CỔ ĐIỀN.....	71
6.4 BẢN MẪU NHANH.....	72
6.5 NHÂN TỐ CON NGƯỜI.....	72
6.6 SỬ DỤNG LẠI BẢN MẪU NHANH.....	73
6.7 CÁC CÔNG CỤ CASE CHO XÁC ĐỊNH YÊU CẦU.....	73
6.8 CÁC THƯỚC ĐO CHO XÁC ĐỊNH YÊU CẦU.....	74
6.9 NHỮNG THỬ THÁCH CHO PHÁ XÁC ĐỊNH YÊU CẦU.....	74
6.10 CASE STUDY CHO PHÁ XÁC ĐỊNH YÊU CẦU.....	74
6.10.1 Bài toán.....	74
6.10.2 Mô tả hệ thống (bằng ngôn ngữ tự nhiên).....	79
6.10.3 Xây dựng sơ đồ use case tổng quan.....	83
6.10.4 Phân rã chi tiết các use case.....	86
<b>CHƯƠNG 7: PHƯƠNG PHÁP PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG.....</b>	<b>91</b>
7.1 LUỒNG CÔNG VIỆC PHÂN TÍCH.....	91
7.2 VIỆC TRÍCH RÚT CÁC LỚP THỰC THẾ.....	91
7.3 PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG CHO BÀI TOÁN THANG MÁY.....	92
7.4 MÔ HÌNH HÓA CHỨC NĂNG.....	92

7.5 MÔ HÌNH HÓA LỚP THỰC THẾ .....	94
7.5.1 Trích rút danh từ.....	94
7.5.2 CRC Cards.....	96
7.6 KIỂM THỬ TRONG PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG.....	98
7.7 CÁC CÔNG CỤ CASE CHO PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG.....	102
7.8 CASE STUDY CHO PHA PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG.....	102
7.8.1. Các kịch bản.....	102
7.8.2 Trích các lớp thực thể.....	106
7.8.3 Phân tích chi tiết từng modul.....	109
<b>CHƯƠNG 8: PHA THIẾT KẾ.....</b>	<b>119</b>
8.1 TỔNG QUAN VỀ PHA THIẾT KẾ.....	119
8.1.1 Dữ liệu và các hành động.....	119
8.1.2 Thiết kế và trừu tượng.....	119
8.1.3 Thiết kế.....	120
8.1.4 Kiểm thử trong pha thiết kế.....	121
8.1.5 Kỹ thuật hình thức cho thiết kế chi tiết.....	122
8.1.6 Kỹ thuật thiết kế hệ thống thời gian thực .....	122
8.1.7 Công cụ CASE cho thiết kế.....	123
8.1.8 Thước đo cho thiết kế.....	123
8.1.9 Những thách thức của pha thiết kế.....	124
8.2 THIẾT KẾ HƯỚNG HÀNH ĐỘNG.....	124
8.3 PHÂN TÍCH VÀ THIẾT KẾ DÒNG DỮ LIỆU.....	125
8.3.1 Phân tích dòng dữ liệu.....	125
8.3.2 Thiết kế dòng dữ liệu.....	130
8.4 THIẾT KẾ HƯỚNG ĐỐI TƯỢNG.....	130
8.5 CASE STUDY CHO PHA THIẾT KẾ.....	133
8.5.1 Thiết kế lớp thực thể.....	133
8.5.2 Thiết kế cơ sở dữ liệu.....	134
8.5.3 Thiết kế chi tiết các modul.....	136
<b>CHƯƠNG 9: PHA CÀI ĐẶT VÀ TÍCH HỢP.....</b>	<b>148</b>
9.1 CÁC PHƯƠNG PHÁP CÀI ĐẶT VÀ TÍCH HỢP.....	148
9.1.1 Luồng công việc cài đặt.....	148
9.1.2 Tích hợp.....	157
9.2 KIỂM THỬ PHA CÀI ĐẶT VÀ TÍCH HỢP.....	161
9.2.1 Luồng công việc kiểm thử cài đặt.....	161
9.2.2 Kiểm thử tích hợp.....	175
9.3 KIỂM THỬ SẢN PHẨM.....	175
9.4 KIỂM THỬ CHẬP NHẬN.....	176
9.5 CASE STUDY CHO PHA CÀI ĐẶT.....	176
9.5.1 Cài đặt.....	177
9.5.2 Kiểm thử đơn vị - JUnit test.....	198
9.5.3 Kiểm thử chức năng - Blackbox test.....	214
<b>CHƯƠNG 10: PHA BẢO TRÌ.....</b>	<b>223</b>
10.1 PHA BẢO TRÌ SAU KHI CHUYÊN GIAO.....	223
10.1.1 Tại sao bảo trì sau khi chuyển giao là cần thiết.....	223
10.1.2 Người lập trình bảo trì sau khi chuyển giao yêu cầu những gì?.....	223
10.1.3 Quản lý bảo trì sau khi chuyển giao.....	225

---

10.1.4 Bảo trì sau khi chuyển giao với kỹ năng phát triển.....	227
10.1.5 Kỹ nghệ ngược.....	228
10.1.6 Công cụ CASE cho bảo trì sau khi chuyển giao.....	228
10.1.7 Thước đo của bảo trì sau khi chuyển giao.....	228
10.1.8 Những thách thức của bảo trì sau khi chuyển giao.....	229
10.2 BẢO TRÌ HỆ PHẦN MỀM HƯỚNG ĐÓI TUỢNG.....	229
10.3 KIỂM THỦ PHA BẢO TRÌ.....	230
<b>CHƯƠNG 11: BÀI TẬP DỰ ÁN.....</b>	<b>231</b>
11.1 HỆ THỐNG QUẢN LÍ BÁN VÉ VÀ LÊN LỊCH CHIẾU CỦA CHUỖI RẠP CHIẾU PHIM.....	231
11.2 HỆ THỐNG QUẢN LÍ ĐĂNG KÍ HỌC VÀ ĐIỂM TÍN CHỈ CỦA SINH VIÊN.....	234
11.3 HỆ THỐNG QUẢN LÍ MUỢN TRẢ SÁCH CỦA THƯ VIỆN QUỐC GIA.....	238
11.4 HỆ THỐNG QUẢN LÍ XUẤT NHẬP HÀNG TẠI MỘT ĐẠI LÝ TRUNG GIAN.....	240
11.5 HỆ THỐNG QUẢN LÍ BÁN VÉ TÀU HỎA.....	243
11.6 HỆ THỐNG QUẢN LÍ BÁN VÉ MÁY BAY CỦA MỘT ĐẠI LÝ.....	246
11.7 HỆ THỐNG QUẢN LÍ ĐẶT BÀN VÀ GỌI MÓN TRONG MỘT NHÀ HÀNG.....	249
11.8 HỆ THỐNG QUẢN LÍ TOUR DU LỊCH DẠNG OPEN TOUR.....	252
11.9 HỆ THỐNG QUẢN LÍ GIẢI VÔ ĐỊCH BÓNG ĐÁ QUỐC GIA.....	254
11.10 HỆ THỐNG QUẢN LÍ LỊCH THI ĐẤU VÀ KẾT QUẢ GIẢI VÔ ĐỊCH CỜ VUA THẾ GIỚI.....	258
11.11 HỆ THỐNG QUẢN LÍ LỊCH THI ĐẤU VÀ KẾT QUẢ GIẢI ĐUA CÔNG THỨC 1 F1.....	260
11.12 HỆ THỐNG QUẢN LÍ CHO THUÊ TRUYỆN CỦA CỬA HÀNG CHO THUÊ TRUYỆN.....	262
11.13 HỆ THỐNG QUẢN LÍ CHO THUÊ SÂN BÓNG MINI CỦA CHỦ SÂN BÓNG.....	265
11.14 HỆ THỐNG QUẢN LÍ CHO THUÊ PHÒNG TRỌ CỦA CHỦ NHÀ TRỌ.....	268
11.15 HỆ THỐNG QUẢN LÍ CHO THUÊ GIƯỜNG KÍ TÚC XÁ.....	271



## CHƯƠNG 1: MỞ ĐẦU

### 1.1 ĐẶC TRƯNG CỦA PHẦN MỀM

- Phần mềm không mòn
- Phần mềm được phát triển mà không được sản xuất theo nghĩa thông thường
- Mặc dù công nghiệp phần mềm đang hướng đến phát triển dựa trên thành phần nhưng phần lớn phần mềm phát triển dựa theo yêu cầu của khách hàng.
- Cho đến nay những đặc trưng của phần mềm vẫn còn là vấn đề tranh cãi. Chính điều này thể hiện sự chưa trưởng thành của ngành học CÔNG NGHỆ PHẦN MỀM.

### 1.2 CÁC KIỂU PHẦN MỀM

- **Phần mềm hệ thống:** Tập hợp các Chương trình được viết để phục vụ các chương trình khác, tương tác với phần cứng (ví dụ, biên dịch, trình soạn thảo, HĐH...)
- **Phần mềm thời gian thực:** Phần mềm điều phối/phân tích kiểm soát, đáp ứng thời gian
- **Phần mềm nghiệp vụ:** Các phần mềm tính lương, kế toán, quản lý kho...
- **Phần mềm khoa học và công nghệ:** Các ứng dụng trong thiên văn, sinh học phân tử, điều khiển tàu con thoi,...
- **Phần mềm nhúng:** Nằm trong bộ nhớ chỉ đọc điều khiển các sản phẩm và hệ thống
- **Phần mềm máy tính cá nhân:** Xử lý văn bản, đồ họa máy tính...
- **Phần mềm trên Web**
- **Phần mềm trí tuệ nhân tạo:** Dựa trên những kỹ thuật của Trí tuệ nhân tạo như hệ chuyên gia.

**Nhận xét:** Hiện nay web được xem là môi trường phổ biến để xây dựng giao diện với người sử dụng cho nhiều hệ thống phần mềm trên mạng.

### 1.3 KHÍA CẠNH LỊCH SỬ

- Năm 1967 nhóm NATO đưa ra thuật ngữ Công nghệ phần mềm (Software Engineering).
- Năm 1968 Hội nghị Software Engineering ở Garmisch, Đức đưa ra mục đích là giải quyết “Cuộc khủng hoảng phần mềm”:
  - Phần mềm hoàn thành không đúng thời hạn
  - Chi phí vượt dự toán ban đầu
  - Phần mềm còn nhiều lỗi
- Tại sao không thể sử dụng kỹ nghệ xây cất như xây dựng cầu để xây dựng các hệ điều hành?
  - Thái độ đối với việc sập cầu/hệ điều hành
  - Thông tin về CNPM thường không đầy đủ, không chắc chắn
  - Độ phức tạp

- Bảo trì

Công nghệ phần mềm không thể xem giống như các kỹ nghệ thông thường khác.

## 1.4 KHÍA CẠNH KINH TẾ

- CNPM và khoa học máy tính (tương tự như hóa học và kỹ nghệ hóa)
    - Khoa học có phần thực nghiệm và lý thuyết: Phần thực nghiệm của Hóa học là thí nghiệm còn của khoa học máy tính là lập trình.
    - Khoa học máy tính nghiên cứu những sách khác nhau để tạo ra phần mềm. Nhưng kỹ sư phần mềm chỉ quan tâm kỹ thuật có ý nghĩa kinh tế.
    - Ví dụ: Phương pháp mã hóa mới KTmới (lập trình hướng thành phần) nhanh hơn phương pháp đang sử dụng hiện thời KTCũ (lập trình hướng đối tượng) là 10%. Chúng ta nên sử dụng phương pháp mới hay không?
- Câu trả lời thông thường là: Tất nhiên! Câu trả lời Công nghệ phần mềm: xét hiệu quả của KTmới.

## 1.5 KHÍA CẠNH BẢO TRÌ

- Vòng đời phần mềm: Một loạt các pha mà phần mềm phải trải qua từ khám phá các khái niệm đến loại bỏ hoàn toàn.
- Mô hình vòng đời
  - Pha yêu cầu
  - Pha đặc tả
  - Pha thiết kế
  - Pha cài đặt
  - Pha tích hợp
  - Pha bảo trì
  - Loại bỏ
- **Bảo trì:** Mọi thay đổi đối với sản phẩm một khách hàng đã đồng ý sản phẩm mua tài liệu đặc tả.
- Phần mềm tốt được bảo trì khác với phần mềm tồi bị loại bỏ
- Các dạng bảo trì”
  - Bảo trì sửa lỗi [17,5%]: sửa chữa lỗi nhưng đặc tả không đổi
  - Bảo trì nâng cao: sửa chữa theo thay đổi của đặc tả. Bảo trì hoàn thiện [60,5%]: thêm chức năng để cải tiến sản phẩm. Bảo trì thích nghi [18%]: thay đổi phần mềm để đáp ứng thay đổi của môi trường như quy định chính phủ, CPU, công nghệ mới...

**Ví dụ 1:** Tỷ lệ thuế GTGT thay đổi từ 6% đến 7%

- C++

```
const float salesTax = 6.0;
```

- JAVA

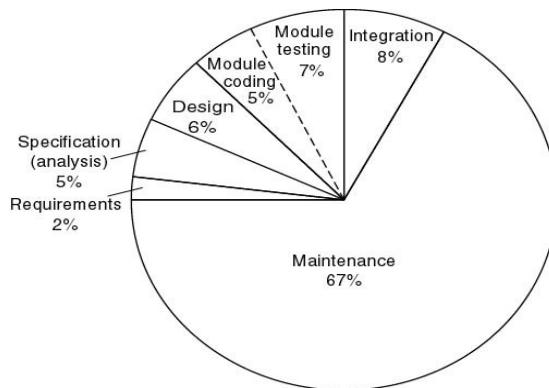
**public static float salesTax = 6.0;**

**Ví dụ 2** Tổ chức y tế thay đổi  $\Rightarrow$  Hoạt động thay đổi

**Ví dụ 3** Các hệ thời gian thực/hệ nhúng

Thế giới thực thay đổi  $\Rightarrow$  hệ thay đổi

- Chi phí cho các pha: Nguồn dữ liệu của các năm 1976-1981



- Hewlett-Packard (1992)
  - 60-80% nguồn nhân lực để phát triển và nghiên cứu dành cho bảo trì
  - 40-60% chi phí phần mềm dành cho bảo trì
- Nhiều tổ chức dành 80% thời gian và công sức cho bảo trì [Yourdon, 1996]

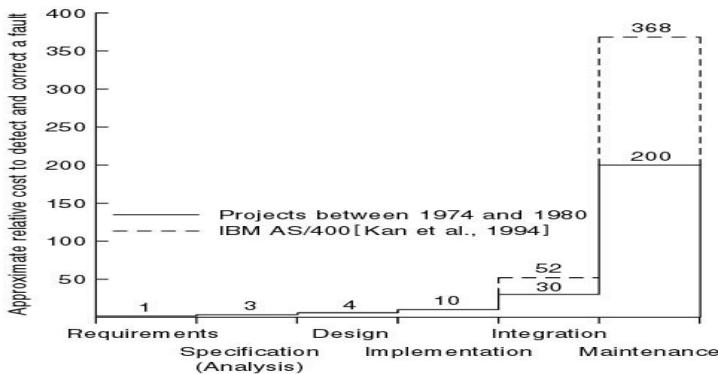
**Kết luận:** Bảo trì là pha tốn kém nhiều thời gian và chi phí

## 1.6 KHÍA CẠNH PHÂN TÍCH VÀ THIẾT KẾ

- 60 đến 70% lỗi của phần mềm là những lỗi do đặc tả và thiết kế
- Dữ liệu của Kelly, Sherif and Hops [1992] về chương trình không gian liên hành tinh của Nasa
  - 1,9 lỗi trên một trang đặc tả
  - 0,9 lỗi trên một trang thiết kế
  - 0,3 lỗi trên một trang chương trình nguồn
- Dữ liệu của Bhandari et al. [1994]: Lỗi trong cuối pha thiết kế của phiên bản mới của sản phẩm
  - 13% lỗi từ phiên bản trước
  - 16% lỗi trong pha đặc tả mới
  - 71% lỗi trong pha thiết kế mới

**Kết luận:** Xây dựng những kỹ thuật sinh ra thiết kế và đặc tả tốt hơn là một vấn đề quan trọng trong công nghệ phần mềm.

Chi phí cho việc phát hiện và sửa chữa lỗi:



## 1.7 KHÍA CẠNH LẬP TRÌNH NHÓM

Phần cứng rẻ, khả năng tăng, kích thước giảm

Phần mềm lớn, đắt. Nhiều phần mềm quá lớn nên một người không thể phát triển được trong thời gian có hạn.

**Ví dụ:** Lan và Minh viết code cho hai Mô đun p và q với mô đun p gọi q. Khi viết p Lan đã viết một hàm gọi q với 5 đối số. Minh cũng code q với 5 đối số nhưng thứ tự khác Lan.

- Compiler của C không thể phát hiện sai sót đó
- Java chỉ có thể phát hiện khi biến khác kiểu nhưng nếu cùng kiểu thì không thể phát hiện

## 1.8 PHƯƠNG PHÁP HƯỚNG ĐỐI TƯỢNG

Trước năm 1975, phần lớn các tổ chức phần mềm có những kỹ thuật riêng, cá nhân có cách làm việc riêng. Giữa những năm 1975-1985: Phương pháp cấu trúc đạt nhiều thành công ban đầu. Kỹ thuật cấu trúc phù hợp với chương trình 5.000 đến 50.000 dòng mã.

### Hạn chế của phương pháp cấu trúc:

- Không phù hợp với phần mềm lớn ( $> 50.000$  dòng lệnh)
- 80% chi phí và sức lực dành cho bảo trì nhưng cách tiếp cận của phương pháp cấu trúc không giải quyết được vấn đề này.

### Đặc trưng của phương pháp cấu trúc:

- Hướng hành động (máy trạng thái hữu hạn, sơ đồ dòng dữ liệu); hay
- Hướng dữ liệu (sơ đồ quan hệ thực thể, phương pháp Jackson);
- Không cả hai

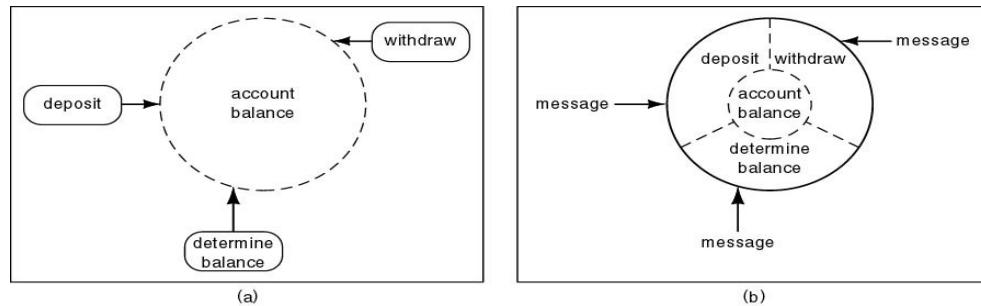
**Phương pháp hướng đối tượng:** Cả hai dữ liệu và hành động đều quan trọng như nhau

**Đối tượng:** Thành phần phần mềm kết hợp cả hai dữ liệu và các hành động thực hiện trên dữ liệu đó.

**Ví dụ:** Quản lý trương mục ngân hàng

- Dữ liệu: tiền gửi vào, rút ra
- Hành động: gửi vào, rút ra, xác định tiền gửi vào rút ra

**Phương pháp lập trình hướng đối tượng khác phương pháp cấu trúc**



- Ân dấu thông tin
- Thiết kế dựa trên trách nhiệm
- Ảnh hưởng đối với bảo trì và phát triển
- **Phương pháp hướng đối tượng:** Một đối tượng gửi message đến một đối tượng khác để yêu cầu hành động/phương pháp.
- **Từ phân tích đến thiết kế**

Structured Paradigm	Object-Oriented Paradigm
1. Requirements phase 2. Specification (analysis) phase 3. Design phase 4. Implementation phase 5. Integration phase 6. Maintenance phase 7. Retirement	1. Requirements phase 2'. Object-oriented analysis phase 3'. Object-oriented design phase 4'. Object-oriented programming phase 5. Integration phase 6. Maintenance phase 7. Retirement

- Phương pháp cấu trúc: Tạo “xúc” giữa phân tích (what) và thiết kế “how”
- Phương pháp hướng đối tượng: (chuyển pha “em dịu” nên giảm được sai sót) Các đối tượng đưa vào ngay từ đầu của vòng đời. Đối tượng được đề xuất trong pha thiết kế, được xây dựng trong pha thiết kế, được mã hóa trong pha lập trình.
- Phân tích hệ thống: Xác định điều cần phải làm WHAT
- Thiết kế xác định cách làm HOW: thiết kế kiến trúc xác định các mô đun và thiết kế chi tiết từng mô đun.
- **Lợi ích của HDT:**
  - Phân tích hướng đối tượng: Xác định cái gì phải thực hiện và xác định các đối tượng
  - Thiết kế hướng đối tượng: Xác định cách thực hiện hành động và xây dựng các đối tượng.
- So sánh phương pháp cấu trúc và phương pháp hướng đối tượng

Structured Paradigm	Object-Oriented Paradigm
2. Specification (analysis) phase <ul style="list-style-type: none"> <li>• Determine what the product is to do</li> </ul>	2'. Object-oriented analysis phase <ul style="list-style-type: none"> <li>• Determine what the product is to do</li> <li>• Extract the objects</li> </ul>
3. Design phase <ul style="list-style-type: none"> <li>• Architectural design (extract the modules)</li> <li>• Detailed design</li> </ul>	3'. Object-oriented design phase <ul style="list-style-type: none"> <li>• Detailed design</li> </ul>
4. Implementation phase <ul style="list-style-type: none"> <li>• Implement in appropriate programming language</li> </ul>	4'. Object-oriented programming phase <ul style="list-style-type: none"> <li>• Implement in appropriate object-oriented programming language</li> </ul>

### Kết luận:

- Công nghệ phần mềm: ngành học nhằm nghiên cứu cách sản xuất ra phần mềm thỏa mãn nhu cầu khách hàng, không có lỗi, giao đúng thời hạn và trong chi phí hạn định.
- Công nghệ phần mềm xem xét các pha của vòng đời phần mềm và phân phối các khía cạnh khác nhau của tri thức con người từ kinh tế, kỹ thuật đến khoa học xã hội, tâm lý...
- Phương pháp hướng đối tượng được xem là cách tiếp cận phổ biến trong phát triển phần mềm hiện nay.

### 1.9 CÂU HỎI ÔN TẬP

1. Phần mềm là gì? Nêu đặc trưng của nó. Có những loại ngôn ngữ nào để phát triển phần mềm?
2. Phân loại phần mềm và nội dung cơ bản mỗi loại.
3. Định nghĩa kỹ nghệ phần mềm? Những yếu tố chủ chốt trong kỹ nghệ phần mềm là gì?
4. Có các dạng bảo trì nào? Nêu và phân biệt.
5. Tại sao không có pha kiểm thử?
6. Tại sao không có pha làm tài liệu?
7. Tại sao không có pha lập kế hoạch?



## CHƯƠNG 2: CÁC PHA PHÁT TRIỂN PHẦN MỀM

### 2.1 TIẾN TRÌNH THÀNH PHẦN

- Tiến trình phần mềm là “phương cách” sản xuất ra phần mềm. Tiến trình phần mềm nghiên cứu các cách kết hợp:
  - Mô hình vòng đời
  - Các công cụ CASE
  - Các cá nhân xây dựng phần mềm
  - Các công nghệ
- **Tiến trình phần mềm = Khía cạnh kỹ thuật + Khía cạnh quản lý**
- Các tổ chức khác nhau có những tiến trình phần mềm khác nhau.
  - Khâu viết tài liệu (quan trọng/không quan trọng)
  - Chi phí dành cho kiểm thử (1/2 chi phí/không quan trọng)
  - Tập trung khaua nghiên cứu, phát triển phần mềm. Khâu bảo trì dành cho người khác.
- Vì sao như vậy?
  - Thiếu kỹ năng về kỹ nghệ phần mềm
  - Nhiều người quản lý phần mềm thiếu kiến thức về bảo trì và phát triển phần mềm (do đó trễ hạn!)
  - Quan điểm về quản lý (giao đúng hạn hay kiểm tra kỹ trước khi giao)
  - Phụ thuộc vào cá nhân

#### Có pha kiểm thử không?

- KHÔNG có pha kiểm thử. Vì sao? Kiểm thử là hoạt động thực hiện trong mọi pha của sản xuất phần mềm.
- Check = test
- Testing: Thường được hiểu sau khi coding
- Kiểm tra (Verification): Thực hiện vào cuối mỗi pha
- Kiểm chứng (Validation): Thực hiện trước khi giao sản phẩm cho khách hàng

#### Có pha viết tài liệu không?

- KHÔNG có pha viết tài liệu
- Mọi pha phải được viết tài liệu trước khi khởi đầu một pha mới.
- Một số lý do:
  - Tài liệu bị hoãn lại thì sẽ không bao giờ hoàn thành
  - Cá nhân chịu trách nhiệm trong pha trước có thể đã chuyển sang bộ phận khác.
  - Sản phẩm thường xuyên thay đổi trong khi phát triển vì thế ta cần tài liệu để ghi lại điều này. Ví dụ, thiết kế thường phải sửa đổi trong khi cài đặt. Việc sửa đổi như vậy chỉ có thể thực hiện được khi có tài liệu của nhóm thiết kế.

### Kiểm thử và viết tài liệu được tiến hành trong mọi pha của tiến trình phần mềm.

#### 2.2 SQA LÀ GÌ?

- Nhóm đảm bảo chất lượng phần mềm (SQA: Software quality assurance):
  - Có trách nhiệm đảm bảo sản phẩm được xây dựng đúng (theo đặc tả) và theo đơn đặt hàng.
  - Nhóm SQA phải đóng đúng vai trò ngay từ đầu của tiến trình và hoạt động trong mọi pha của tiến trình phần mềm.
  - SQA kiểm tra với khách hàng xem phiên bản cuối cùng thỏa mãn hoàn toàn chưa.

#### 2.3 PHA YÊU CẦU

- Tiến trình phát triển phần mềm bắt đầu khi khách hàng tiếp xúc với công ty phần mềm và cho rằng: Phần mềm có khả năng thích hợp với khách hàng và giá thành hợp lý.
- Khám phá khái niệm:
  - Trong lần gặp đầu tiên, khách hàng phát họa sản phẩm mà họ hình dung. Theo quan điểm của người phát triển, mô tả này không rõ ràng, không hợp lý, mâu thuẫn hay không thể xây dựng phần mềm như thế được.
  - Người phát triển xác định nhu cầu và ràng buộc của khách hàng

#### Kiểm thử pha yêu cầu:

- Làm bản mẫu nhanh: là mẫu phần mềm kết hợp nhiều chức năng của sản phẩm cuối cùng nhưng bỏ qua những khía cạnh mà khách hàng không thấy được như cập nhật file hay xử lý lỗi.
- Bản mẫu nhanh phải được kiểm tra bởi khách hàng và người sử dụng.
- Bản mẫu nhanh có thể thay đổi cho đến khi khách hàng và người sử dụng cho rằng nó có những chức năng mà họ mong muốn.

#### Viết tài liệu pha yêu cầu:

- Tài liệu pha yêu cầu:
  - Có bản mẫu: Bản mẫu nhanh. Bản ghi thỏa thuận với khách hàng và người sử dụng về cơ sở xây dựng và sửa đổi bản mẫu.
  - Không có bản mẫu (nhóm quyết định không xây dựng bản mẫu): Mô tả nhu cầu khách hàng. Tài liệu này phải được khách hàng, người sử dụng và nhóm phát triển kiểm tra trước khi nhóm SQA xem xét.

#### 2.4 PHA ĐẶC TẢ

- Khi khách hàng cho rằng nhóm phát triển hiểu được yêu cầu, nhóm đặc tả viết tài liệu đặc tả để mô tả chức năng của sản phẩm (những gì sản phẩm cần có + ràng buộc).
- Đặc tả bao gồm những input của sản phẩm và output được yêu cầu
- Ví dụ: Khách hàng cần tính bảng lương thì input bao gồm mức trả cho mỗi nhân viên, thông tin từ hồ sơ cá nhân để tính thuế... và output là số lương thuế, chi bảo hiểm,...

- Yêu cầu của đặc tả
  - Không nhập nhằng: không sử dụng thuật ngữ như tiện lợi, đầy đủ chức năng, nhanh, 98%...
  - Đầy đủ: Thể hiện mọi yêu cầu của khách hàng
  - Phi mâu thuẫn: không chứa mâu thuẫn
  - Theo dõi được (Traceability): có thể lần theo phán đoán trong đặc tả trở lại phán đoán đưa ra bởi khách hàng trong pha yêu cầu. Nếu đặc tả được trình bày đúng phương pháp, có đánh chỉ số,... thì nhóm SQA sẽ ít gặp khó khăn. Nếu có bản mẫu thì phán đoán liên quan của đặc tả phải theo dõi được đến bản mẫu.
- Một khi đặc tả được hoàn thành và đã thông qua thì hình thành kế hoạch quản lý quá trình sản xuất phần mềm (SPMP : The software product management plan).
- Yêu cầu kế hoạch:
  - SPMP cần nêu lên thời gian thực hiện, chi phí cho từng pha, gán trách nhiệm cá nhân cho từng pha, thời hạn hoàn thành cho mỗi pha.
  - Mô hình vòng đời nào sẽ sử dụng, cấu trúc tổ chức, kỹ thuật và CASE sử dụng, lịch tinh, chi phí...

#### Kiểm thử pha đặc tả:

- Nguồn gốc chính của lỗi trong các phần mềm đã phân phối đến nay là những lỗi trong tài liệu đặc tả và những lỗi này chỉ được phát hiện khi tổ chức khách hàng sử dụng nó.
- Duyệt xét lại (Review): là cách tốt nhất để kiểm tra đặc tả. Mục đích là xác định đặc tả có đúng không. Nhóm SQA chủ trì cuộc họp với đại diện nhóm đặc tả và khách hàng.

## 2.5 PHA THIẾT KẾ

- Đặc tả - What
- Thiết kế - How
- Giữ lại quyết định thiết kế
  - Thời điểm kết thúc
  - Cho nhóm bảo trì
  - Thiết kế nên mở (open-ended)
- Thiết kế kiến trúc : Phân rã sản phẩm thành mô đun
- Thiết kế chi tiết: Thiết kế các mô đun: cấu trúc dữ liệu, thuật toán

#### Kiểm thử pha thiết kế

- Tài liệu viết sao cho dễ theo dõi
- Duyệt tài liệu

## 2.6 PHA CÀI ĐẶT

- Cài đặt thiết kế chi tiết thành chương trình

#### Kiểm thử pha cài đặt:

- Rà soát

- Các test case
  - Test không hình thức (desk checking)
  - Test hình thức (Formal testing) do nhóm SQA

## 2.7 TÍCH HỢP

- Kết hợp các mô đun và kiểm thử toàn bộ sản phẩm

### Tài liệu pha tích hợp

- Mã nguồn có chủ thích
- Các test cases

## 2.8 CẢI TIẾN TIẾN TRÌNH PHẦN MỀM

- Bắt đầu từ Bộ Quốc Phòng Mỹ
- Software Engineering Institute (SEI)
- Ván đề:
  - Quản lý tiến trình phần mềm kém
- Cải tiến tiến trình phần mềm
  - Capability maturity model (CMM)
  - ISO 9000-Series
  - ISO/IEC 15504

### CMM: Capability Maturity Model

- Không phải mô hình vòng đời
- Tập các chiến lược cải tiến tiến trình phần mềm
  - SW-CMM for software
  - P-CMM for human resource (“people”)
  - SE-CMM for systems engineering
  - IPD-CMM for integrated product development
  - SA-for software acquisition
- Các chiến lược được thống nhất thành CMMI (Capability maturity model integration)

### SW-CMM

- A strategy for improving the software process
  - Put forward in 1986 by the SEI
  - Fundamental idea:
  - Improving the software process leads to
    - Improved software quality
    - Delivery on time, within budget
  - Improved management leads to
    - Improved techniques
- Five levels of “maturity” are defined

- Organization advances stepwise from level to level

#### Level 1: Initial Level

- Ad hoc approach
  - Entire process is unpredictable
  - Management consists of responses to crises
- Most organizations world-wide are at level 1

#### Level 2: Repeatable Level

- Basic software management
  - Management decisions should be made on the basis of previous experience with similar products
  - Measurements (“metrics”) are made
  - These can be used for making cost and duration predictions in the next project
  - Problems are identified, immediate corrective action is taken

#### Level 3: Defined Level

- The software process is fully documented
  - Managerial and technical aspects are clearly defined
  - Continual efforts are made to improve quality, productivity
  - Reviews are performed to improve software quality
  - CASE tools are applicable *now* (and not at levels 1 or 2)

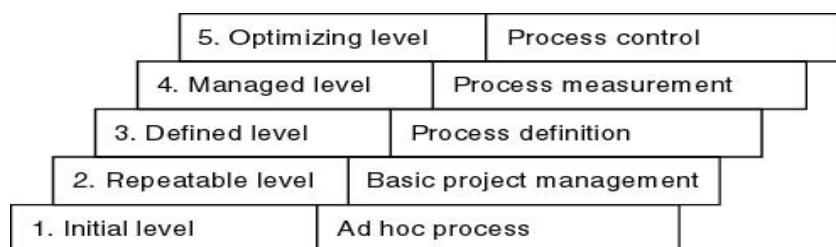
#### Level 4: Managed Level

- Quality and productivity goals are set for each project
  - Quality, productivity are continually monitored
  - Statistical quality controls are in place

#### Level 5: Optimizing level

- Continuous process improvement
  - Statistical quality and process controls
  - Feedback of knowledge from each project to the next

Kết luận:



#### Tiến trình chính

- Có các tiến trình chính (key Process Area) cho mỗi mức
- Mức 2:

- Quản lý yêu cầu
- Lập kế hoạch dự án
- Theo dõi dự án
- Quản lý cấu hình
- Bảo đảm chất lượng
- So sánh
  - Mức 2: Phát hiện và sửa lỗi
  - Mức 3: Ngăn chặn lỗi

### Kinh nghiệm

- Cân:
  - 3 đến 5 năm để từ mức 2 lên mức 2
  - 1.5 đến 3 năm từ mức 2 lên mức 3
- Original idea: Defense contracts would be awarded only to capable firms
- Profitability
  - Hughes Aircraft (Fullerton, CA) spent \$500K (1987–90)
    - Savings: \$2M per year, moving from level 2 to level 3
  - Raytheon moved from level 1 in 1988 to level 3 in 1993
    - Productivity doubled
    - Return of \$7.70 per dollar invested in process improvement

### Other SPI Initiatives

- Other software process improvement (SPI) initiatives:
  - ISO 9000-series
  - ISO/IEC 15504

### ISO 9000

- Set of five standards for industrial activities
  - ISO 9001 for quality systems
  - ISO 9000-3, guidelines to apply ISO 9001 to software
  - There is an overlap with CMM, but they are not identical
  - *Not* process improvement
  - Stress on documenting the process
  - Emphasis on measurement and metrics
  - ISO 9000 is required to do business with the E.U.
  - Also by many U.S. businesses, for example, GE
  - More and more U.S. businesses are ISO 9000 certified

### ISO/IEC 15504

- Original name: Software Process Improvement Capability dEtermination (SPICE)
  - International process improvement initiative

- Started by British Ministry of Defence (MOD)
- Includes process improvement, software procurement
- Extends and improves CMM, ISO 9000
- Framework, not a method
  - CMM, ISO 9000 conform to this framework
- Now referred to as ISO/IEC 15504
- Or just 15504 for short

### Process Improvement Data

Category	Range	Median	Number of Data Points
Years engaged in software process improvement (SPI)	1–9	3.5	24
Yearly cost of SPI per software engineer	\$490–\$2004	\$1375	5
Productivity gain per year	9%–67%	35%	4
Early defect detection gain per year	6%–25%	22%	3
Yearly reduction in time to market	15%–23%	19%	2
Yearly reduction in post-release defect reports	10%–94%	39%	5
Business value (saving/cost of SPI)	4.0–8.8:1	5.0:1	5

- SEI report on 13 organizations in the original study
- They used a variety of process improvement techniques, not just SW-CMM

CMM Level	Number of Projects	Relative Decrease in Duration	Faults per MEASL Detected during Development	Relative Productivity
Level 1	3	1.0	—	—
Level 2	9	3.2	890	1.0
Level 3	5	2.7	411	0.8
Level 4	8	5.0	205	2.3
Level 5	9	7.8	126	2.8

## 2.9 CÂU HỎI ÔN TẬP

1. Tiến trình phần mềm là gì? Mô hình tiến trình là gì? Hãy trình bày mô hình của một số tiến trình cơ bản.
2. Các bước tổng quát của tiến trình phần mềm gồm những giai đoạn nào? Nêu các hoạt động của tiến trình phần mềm và tài liệu mà nó sinh ra?
3. Chất lượng phần mềm là gì? Các tiêu chí của chất lượng phần mềm.
4. Có các dạng bảo trì nào? Nêu và phân biệt.
5. Thế nào là portability của phần mềm?
6. Thế nào là reuseability của phần mềm?
7. Thế nào là một bản thiết kế còn omission?
8. Thế nào là một bản thiết kế còn contradiction?

9. Thế nào là một phần mềm COTS?
10. SPMP là viết tắt của từ gì? Ý nghĩa?
11. Mô hình CMM là gì? Có những mức tăng trưởng nào trong mô hình CMM? Nội dung của mỗi mức?
12. Làm thế nào để một tổ chức đạt được các mức tăng trưởng trong CMM? Đâu là giải pháp và thước đo về các mức tăng trưởng?
13. Nêu các chuẩn quốc tế về phần mềm. Trình bày sự khác nhau giữa mô hình CMM và các chuẩn đó.

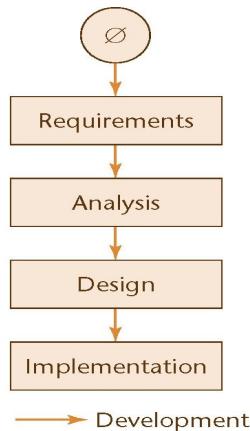
## CHƯƠNG 3

### CÁC MÔ HÌNH VÒNG ĐỜI PHẦN MỀM

#### 3.1 PHÁT TRIỂN PHẦN MỀM

##### 3.1.1 Theo lý thuyết phát triển phần mềm:

Theo lý tưởng, phần mềm được phát triển: tuyến tính và bắt đầu từ con số 0



##### 3.1.2 Thực tế phát triển phần mềm

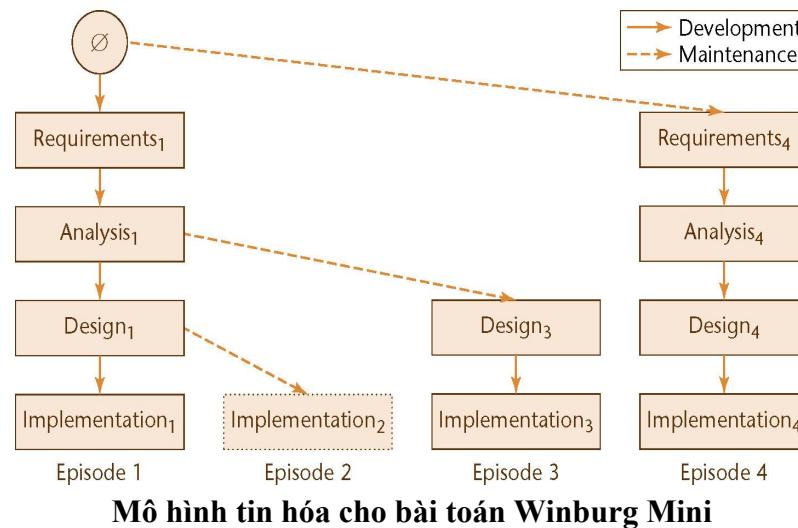
Trong thực tế, phát triển phần mềm hoàn toàn khác:

- Người phát triển có thể mắc lỗi trong quá trình phát triển phần mềm
- Yêu cầu của khách hàng thay đổi trong khi hệ thống phần mềm đang được xây dựng

##### 3.1.3 Bài toán Winburg Mini

- **Episode 1:** Phiên bản đầu tiên được cài đặt
- **Episode 2:** Tìm ra các lỗi
  - Hệ thống phần mềm được xây dựng quá lâu bởi vì việc sử lỗi cài đặt
  - Thay đổi trong cài đặt được bắt đầu.
- **Episode 3:** Thiết kế mới được chấp nhận
  - Thuật toán nhanh hơn được sử dụng
- **Episode 4:** Các yêu cầu thay đổi
  - Yêu cầu độ chính xác cao hơn

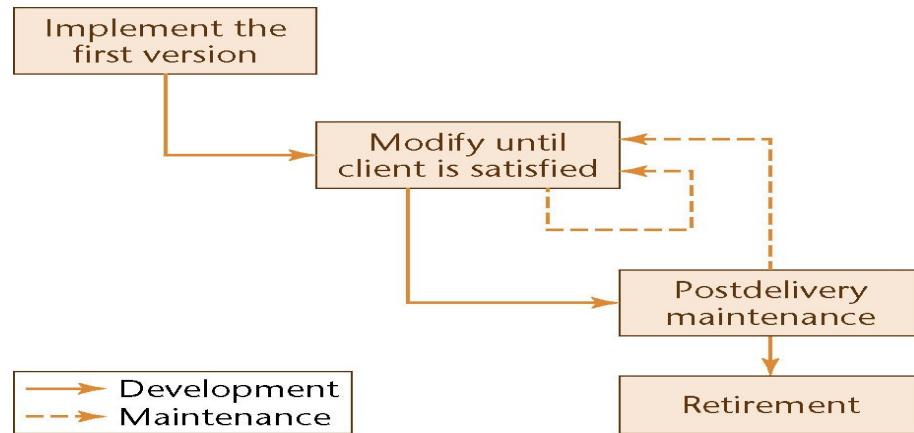
**Kết luận:** Một vài năm sau, những vấn đề này lại xảy ra



Mô hình tin hóa cho bài toán Winburg Mini

### 3.2 MÔ HÌNH XÂY SỬA

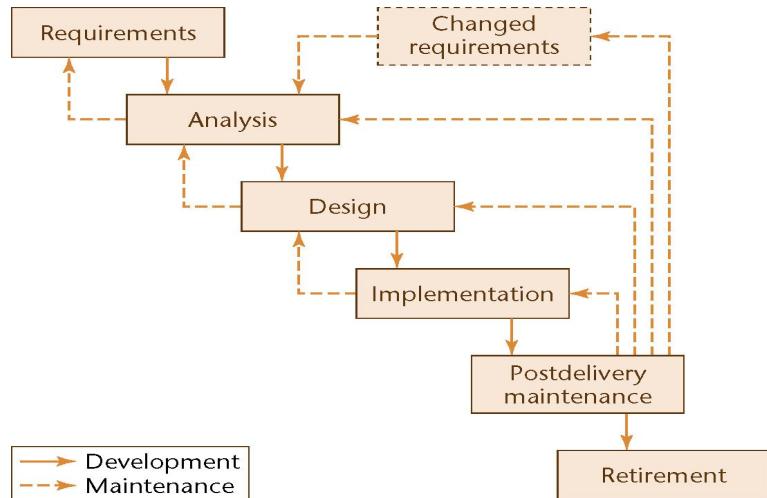
- Không thiết kế
- Không đặc tả
  - Bảo trì là ác mộng
- Là cách dễ dàng nhất để phát triển phần mềm
- Là cách đắt nhất



### 3.3 MÔ HÌNH TIẾN HÓA

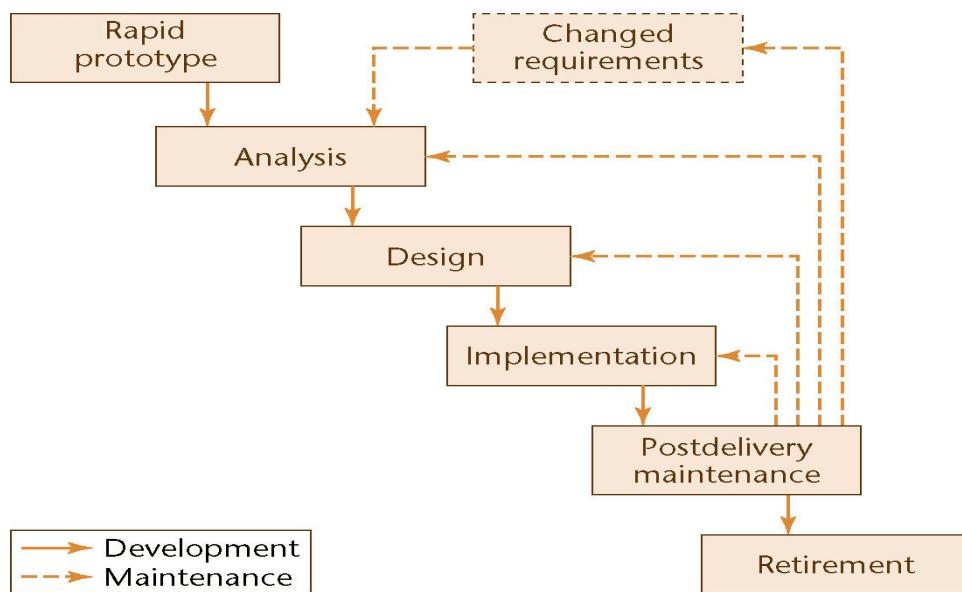
- Đặc trưng bởi
  - Các vòng lặp phản hồi
  - Hướng tài liệu
  - Mô hình thác nước không biểu hiện thứ tự các sự kiện
- Thuận lợi
  - Tài liệu

- Bảo trì dễ dàng
- Bất lợi
  - Tài liệu đặc tả
    - Joe và Jane Johnson
    - Mark Marberry

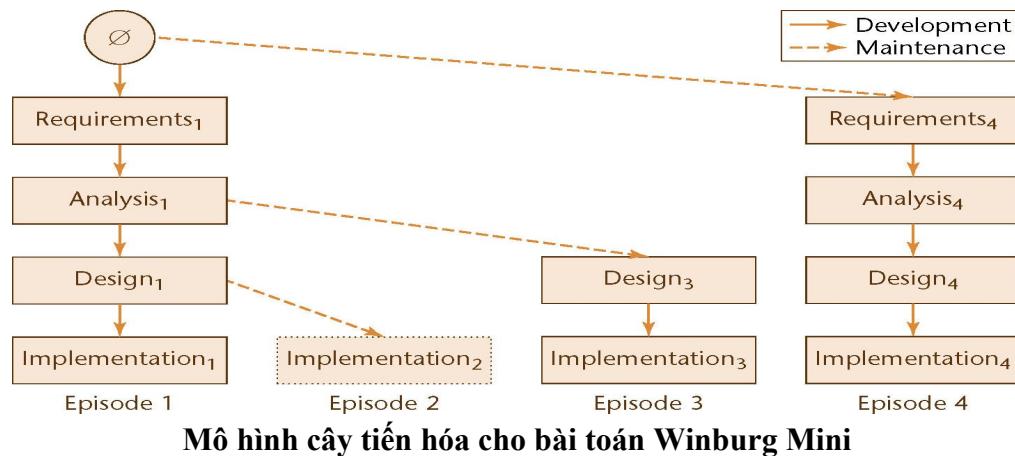


### 3.4 MÔ HÌNH BẢN MẪU NHANH

- Mô hình tuyến tính
- “Nhanh”



### Mô hình tiến hóa



- Thứ tự của các sự kiện được chỉ ra rõ ràng
- Kết thúc của mỗi Episode:
  - Có một đường cơ sở chỉ rõ tập các tài liệu phải được hoàn thiện
- Chặng hạn:
  - Đường cơ sở ở cuối Episode 3 là:
    - Các yêu cầu<sub>1</sub>, Tài liệu phân tích<sub>1</sub>, Tài liệu thiết kế<sub>3</sub>, cài đặt<sub>3</sub>
- Các bài học rút ra từ bài toán Winburg Mini
- Trong thực tế, phát triển phần mềm nhiều hơn hỗn độn hơn bài toán Winburg mini.
- Thay đổi luôn luôn cần thiết
  - Hệ thống phần mềm là một mô hình của thế giới thực và nó luôn luôn thay đổi
  - Chuyên gia phần mềm là con người nên có thể mắc lỗi trong quá trình phát triển phần mềm

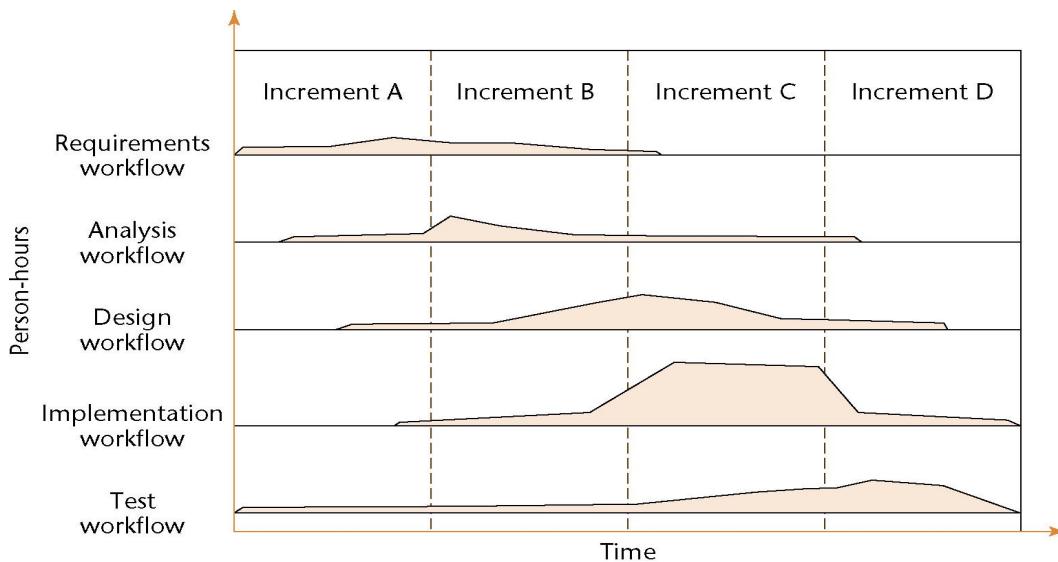
### 3.5. MÔ HÌNH LẶP VÀ TĂNG

- Trong thực tế, chúng ta không thể nói về “pha phân tích”
  - Mặc dù, các thao tác của pha phân tích trải rộng xuyên suốt vòng đời phần mềm.
- Tiến trình phát triển phần mềm là lặp
  - Mỗi phiên bản kế tiếp gần với hệ thống đích hơn phiên bản trước đó.

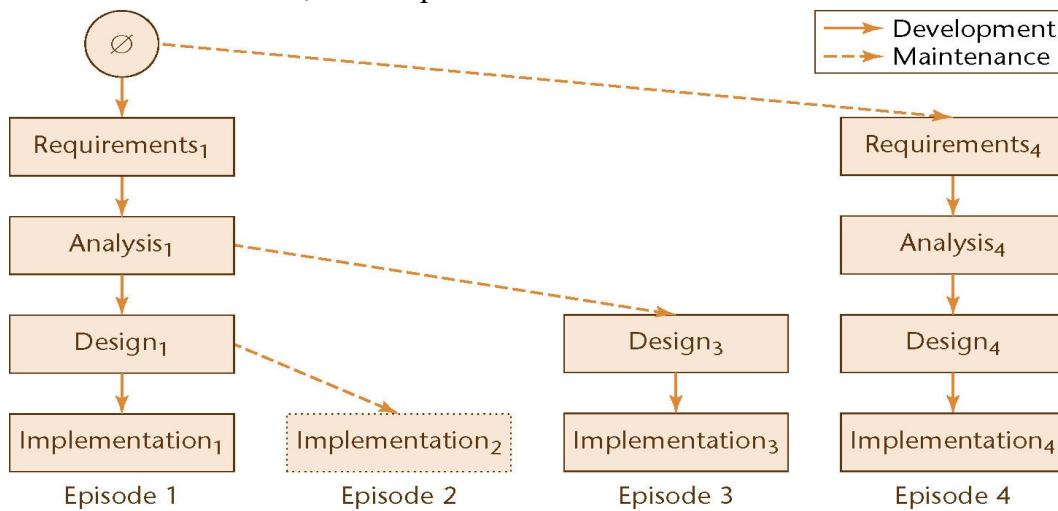
Luật Miller:

- Ở bất cứ thời điểm nào, chúng ta chỉ có thể tập trung vào khoảng 7 chunks (tương ứng 7 đơn vị thông tin)
- Để xử lý lượng thông tin lớn hơn yêu cầu sử dụng bước làm mìn theo kiểu bậc thang
  - Tập trung vào các khía cạnh quan trọng nhất hiện thời
  - Các khía cạnh trì hoãn thường ít quan trọng hơn
  - Cuối cùng mọi khía cạnh đều được xử lý nhưng theo thứ tự mức độ quan trọng hiện thời

Đây là tiến trình tăng



- Lặp và tăng được sử dụng chung với các mô hình khác.
  - Không có Episode nào mà chỉ có pha “xác định yêu cầu” hoặc “pha thiết kế”
  - Có nhiều thể hiện ở mỗi pha



- Số lượng của sự gia tăng sẽ thay đổi – không phải là 4
- Và số lượng vòng lặp cũng thay đổi không phải luôn bằng ba.

Các pha cổ điển với các workflow

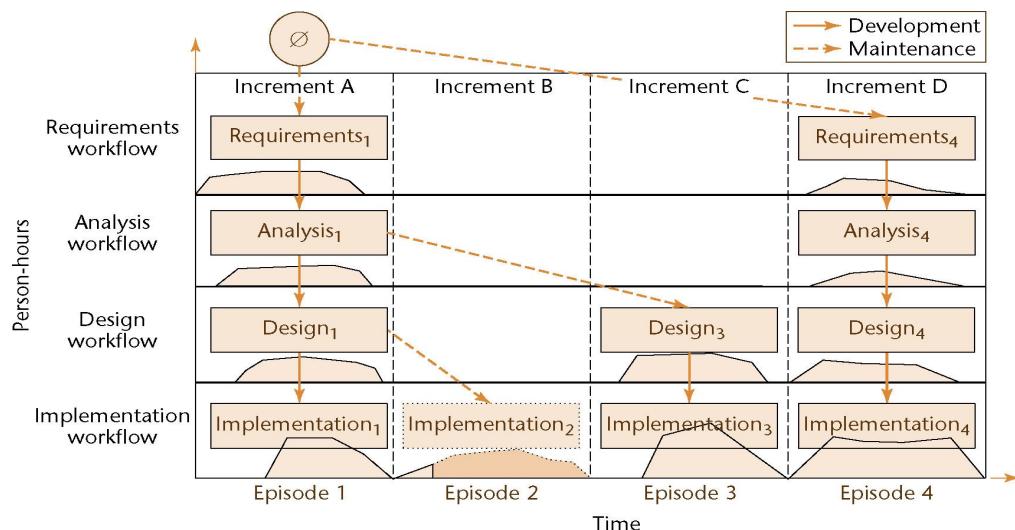
- Các pha tuần tự không có trong thế giới thực
- Mặc dù có 5 workflow (luồng công việc) chính được thực hiện ở trên toàn vòng đời phát triển phần mềm
  - Luồng công việc xác định yêu cầu - Requirements workflow
  - Luồng công việc phân tích - Analysis workflow
  - Luồng công việc thiết kế - Design workflow
  - Luồng công việc cài đặt - Implementation workflow
  - Luồng công việc kiểm thử - Test workflow

### Các luồng công việc

- Cả năm luồng công việc chính được thực hiện trên toàn bộ vòng đời phần mềm
- Tuy nhiên, ở mỗi một thời điểm có một luồng công việc chiếm ưu thế.
- Chẳng hạn:
  - Ở đầu mỗi vòng đời phát triển phần mềm
    - Luồng công việc đặc tả yêu cầu chiếm ưu thế
  - Ở cuối mỗi vòng đời phát triển phần mềm
    - Luồng công việc cài đặt và kiểm thử chiếm ưu thế
- Các hoạt động lập kế hoạch và viết tài liệu được thực hiện xuyên suốt vòng đời phát triển phần mềm

Xem xét lại bài toán Winburg Mini

- Mô hình cây tiến hóa đã được thêm vào mô hình vòng đời lặp và tăng
- Luồng kiểm thử đã bị bỏ quên – mô hình cây tiến hóa giả sử rằng kiểm thử liên tục



- Đối với quá trình tăng:
  - Mỗi Episode tương ứng với một sự gia tăng
  - Không phải mỗi sự gia tăng đều bao gồm mọi luồng công việc
  - Sự gia tăng B không được hoàn thiện
  - Những đường nét đứt biểu diễn sự bảo trì
    - Episodes 2, 3: Bảo trì sửa lỗi
    - Episode 4: Bảo trì hoàn thiện chức năng

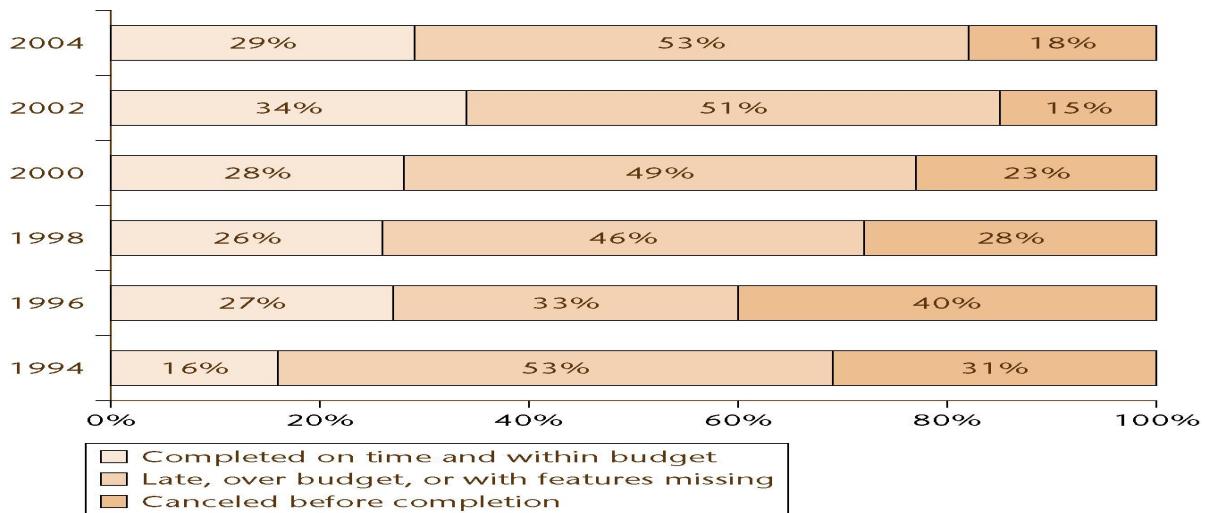
Rõ ràng và những khía cạnh khác của mô hình lặp và tăng

- Chúng ta có thể xem xét toàn bộ dự án như là một tập các dự án nhỏ (quá trình tăng)
- Mỗi dự án nhỏ gồm
  - Tài liệu yêu cầu
  - Tài liệu phân tích

- Tài liệu thiết kế
- Tài liệu cài đặt
- Tài liệu kiểm thử
- Tập tài liệu cuối cùng là sản phẩm phần mềm hoàn thiện
- Trong suốt dự án nhỏ: During each mini project we
  - Mở rộng các tài liệu (Sự gia tăng);
  - Kiểm tra tài liệu (luồng công việc kiểm thử); và
  - Nếu cần, thay đổi các tài liệu liên quan (vòng lặp)
- Mỗi vòng lặp có thể được xem xét như là một mô hình vòng đời thác nước nhỏ nhưng hoàn thiện
- Trong suốt mỗi vòng lặp chúng ta lựa chọn một phần của hệ thống phần mềm
- Trong mỗi phần đó cần thực hiện:
  - Pha xác định yêu cầu cốt điểm
  - Pha phân tích cốt điểm
  - Pha thiết kế cốt điểm
  - Pha cài đặt cốt điểm

Điểm mạnh của mô hình vòng đời lặp và tăng

- Có nhiều ưu điểm cho việc kiểm tra tính đúng đắn của sản phẩm
  - Mỗi vòng lặp có tích hợp luồng công việc kiểm thử
  - Các lỗi có thể được phát hiện và sửa chữa sớm
- Sự mạnh mẽ của kiến trúc có thể được xác định sớm trong vòng đời
  - Kiến trúc – các module thành phần khác nhau và cách chúng kết hợp với nhau
  - *Sự mạnh mẽ - có khả năng xử lý việc mở rộng và thay đổi mà hệ thống không bị tách thành từng mảnh*
- Chúng ta có thể giảm bớt rủi ro sớm hơn
  - Lúc nào cũng vậy rủi ro luôn liên quan tới quá trình phát triển phần mềm và bảo trì
- Chúng ta có một phiên bản hệ thống phần mềm đang làm việc
  - Khách hàng và người dùng có thể thử nghiệm với phiên bản này để xác định cái họ cần thay đổi
  - Mức độ thay đổi: các phiên bản từng phần đưa ra để làm cho sự giới thiệu sản phẩm mới tới tổ chức khách hàng dễ dàng hơn.
- Có bằng chứng thực nghiệm chứng tỏ mô hình vòng đời làm việc
- Các bản tường trình CHAOS của nhóm Standish đã chỉ ra phần trăm phần mềm thành công tăng



- Lý do của sự suy giảm của các dự án thành công năm 2004 bao gồm:
  - Nhiều dự án lớn trong năm 2004 hơn năm 2002
  - Sử dụng mô hình thác nước
  - Thiếu sự quan tâm của người dùng
  - Thiếu sự hỗ trợ từ những người điều hành lâu năm

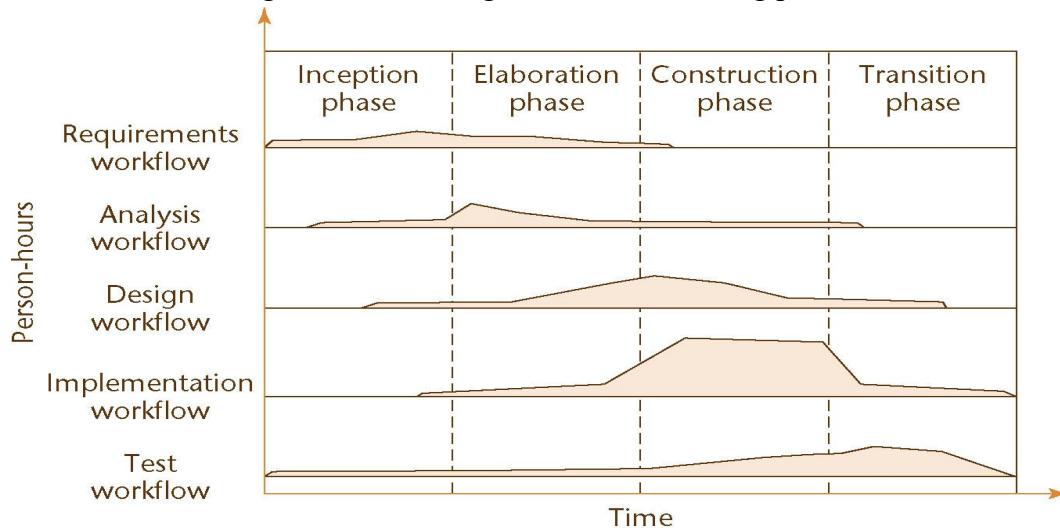
Việc quản lý lặp và tăng

- Mô hình vòng đời lặp và tăng là tập hợp các mô hình thác nước...
- ... bởi vì mô hình vòng đời lặp và tăng là mô hình thác nước, được áp dụng liên tiếp
- Mỗi sự tăng là một dự án nhỏ theo mô hình vòng đời thác nước

### 3.6 MÔ HÌNH UP

- Cho đến gần đây, ba phương pháp luận hướng đối tượng thành công nhất:
  - Phương pháp của Booch
  - Jacobson's Objectory
  - Rumbaugh's OMT
- Năm 1999, Booch, Jacobson, và Rumbaugh đã công bố phương pháp luận phân tích và thiết kế hướng đối tượng hoàn thiện, đó là sự hợp nhất của ba phương pháp tách biệt.
  - Tên đầu tiên : *Rational Unified Process* (RUP)
  - Tiếp theo: *Unified Software Development Process* (USDP)
  - Tên được sử dụng ngày nay: *Unified Process* (for brevity)
- Quy trình hợp nhất không phải là một chuỗi các bước xây dựng hệ thống phần mềm
  - Không tồn tại phương pháp luận “một phù hợp với tất cả”
  - Có sự khác biệt lớn giữa các loại phần mềm
- Quy trình hợp nhất là một phương pháp luận có thể thích hợpThe Unified Process is an adaptable methodology
  - Nó phải được chỉnh sửa tùy theo từng phần mềm cụ thể được phát triển

- UML là đồ họa
  - Một bức tranh đáng giá ngàn từ
- Các biểu đồ UML cho phép kỹ sư phần mềm giao tiếp nhanh hơn và chính xác hơn
- Quy trình hợp nhất là một kỹ thuật mô hình hóa
  - Một mô hình là một tập các biểu đồ UML biểu diễn các khía cạnh khác nhau của sản phẩm phần mềm mà chúng ta muốn phát triển
- UML là viết tắt của ngôn ngữ mô hình hợp nhất (Unified Modeling Language)
  - UML là công cụ được sử dụng để mô hình hệ thống phần mềm đích.



- Bốn quá trình tăng mang tên:
  - Pha khởi đầu – Inception Phase
  - Pha khảo sát tỉ mỉ - Elaboration phase
  - Pha xây dựng - Construction phase
  - Pha chuyển giao - Transition phase
- Các pha của tiến trình hợp nhất là tăng
- Theo lý thuyết, số lượng quá trình tăng là bất kỳ
  - Trong thực tế, quá trình phát triển thường gồm 4 quá trình tăng
- Mỗi bước thực hiện trong tiến trình hợp nhất được chia thành
  - Một trong 5 luồng công việc chính và cũng có thể
  - Một trong bốn pha
- Tại sao mỗi bước phải được xem xét hai lần?
- Luồng công việc
  - Là ngữ cảnh kỹ thuật của một bước
- Pha
  - Là ngữ cảnh nghiệp vụ của mỗi bước

#### *Pha khởi đầu*

Mục đích của pha này là xác định liệu sản phẩm phần mềm đã đề xuất có thể làm được về mặt tài chính

1. Hiểu được lĩnh vực xây dựng phần mềm
  2. Xây dựng mô hình nghiệp vụ
  3. Phân định phạm vi của dự án đã đề xuất
    - Tập trung vào tập con của mô hình nghiệp vụ mà đã được bao phủ bởi sản phẩm phần mềm đề xuất
  4. Bắt đầu thực hiện những trường hợp nghiệp vụ ban đầu
- Các câu hỏi cần được trả lời bao gồm:
    - Có phải sản phẩm phần mềm đề xuất ước tính chi phí hiệu quả?
    - Bao lâu sẽ thu được vốn đầu tư??
    - Về mặt giải pháp, chi phí sẽ lấy từ đâu nếu công ty quyết định không phát triển sản phẩm phần mềm đã đề xuất?
    - Nếu sản phẩm phần mềm không được bán trên thị trường thì có phải việc nghiên cứu thị trường cần thiết được thực hiện không?
    - Sản phẩm phần mềm được đề xuất có thể được chuyển giao đúng thời gian không?
    - Nếu sản phẩm phần mềm được phát triển để hỗ trợ các hoạt động của tổ chức khách hàng, cái gì sẽ bị ảnh hưởng nếu sản phẩm phần mềm được chuyển giao muộn?
    - Những rủi ro nào liên quan đến việc phát triển phần mềm?
    - Những rủi ro này có thể giảm nhẹ đi như thế nào?
      - Có phải đội phát triển sản phẩm phần mềm đã đề xuất là những người có kinh nghiệm?
      - Có phải sản phẩm phần mềm này cần phần cứng mới?
      - Nếu như vậy, thì có phải có một rủi ro thì sản phẩm phần mềm đề xuất sẽ không được chuyển giao đúng thời gian?
      - Có phải có một cách để giảm nhẹ rủi ro, đó là bằng cách đưa ra phần cứng sao chép dự phòng từ nhà cung cấp khác??
      - Các công cụ phần mềm cần được yêu cầu (chương 5)? Are software tools (Chapter 5) needed?
      - Có phải chúng luôn sẵn có?
      - Có phải chúng có tất cả những chức năng cần thiết?

Tất cả các câu trả lời đều được yêu cầu ở cuối pha khởi đầu để trường hợp nghiệp vụ khởi đầu có thể được tạo ra

- Năm được phiên bản đầu tiên của trường hợp nghiệp vụ là mục đích nói chung của pha khởi đầu
- Các phiên bản đầu tiên kết hợp chặt chẽ
  - Bản miêu tả phạm vi của sản phẩm phần mềm
  - Chi tiết về mặt tài chính
  - Nếu sản phẩm phần mềm được đề xuất có thể được kinh doanh thì trường hợp nghiệp vụ sẽ bao gồm:

- Đưa ra kế hoạch thu nhập, ước lượng thị trường, ước lượng chi phí ban đầu
- Nếu sản phẩm phần mềm được sử dụng nội bộ thì trường hợp nghiệp vụ bao gồm
  - Phân tích lợi nhuận và chi phí ban đầu

Các rủi ro:

- Có ba loại rủi ro chính:
  - Rủi ro kỹ thuật
    - Xem ở slide trước đó
  - Rủi ro do xác định yêu cầu không đúng
    - Được giảm nhẹ đi bằng việc thực hiện luồng công việc đặc tả yêu cầu một cách chính xác
  - Rủi ro do thực hiện kiến trúc không đúng
    - Kiến trúc không thể đủ mạnh mẽ
- Làm giảm nhẹ ba loại rủi ro
  - Các rủi ro cần được xếp hạng để những rủi ro quan trọng được làm giảm nhẹ trước
- Công việc này kết thúc các bước của pha khởi đầu đối với luồng công việc xác định yêu cầu

Luồng công việc phân tích, thiết kế

- Một phần nhỏ luồng công việc phân tích có thể được thực hiện trong suốt pha khởi đầu
  - Thông tin cần thiết cho thiết kế kiến trúc cần được trích rút.
- Theo đó, một phần nhỏ luồng công việc của thiết kế cũng được thực hiện

Luồng công việc cài đặt

- Cài đặt nói chung được thực hiện trong suốt pha khởi đầu
- Tuy nhiên, đôi khi proof-of-concept-prototype được xây dựng để kiểm thử tính khả thi của việc xây dựng mỗi phần của sản phẩm phần mềm

Luồng công việc kiểm thử

- Luồng công việc kiểm thử gần như bắt đầu ở giai đoạn đầu của pha khởi đầu
  - Mục đích để đảm bảo rằng các yêu cầu được xác định một cách chính xác

Lập kế hoạch ở pha khởi đầu

- Bắt đầu pha khởi đầu không có đủ thông tin để lập kế hoạch cho toàn bộ quá trình phát triển phần mềm
  - Việc lập kế hoạch duy nhất được thực hiện ở đầu dự án là việc lập kế hoạch cho chính pha khởi đầu
- Cùng với lý do đó, việc lập kế hoạch duy nhất được thực hiện ở cuối pha khởi đầu là bản kế hoạch cho pha tiếp theo (pha khảo sát kỹ mĩ)

Tài liệu của pha khởi đầu

- Những sản phẩm chuyên giao ở pha khởi đầu bao gồm::
  - Phiên bản đầu tiên của mô hình lĩnh vực hoạt động
  - Phiên bản đầu tiên của mô hình nghiệp vụ
  - Phiên bản đầu tiên của tài liệu xác định yêu cầu
  - Phiên bản sơ bộ của tài liệu phân tích
  - Phiên bản sơ bộ của kiến trúc
  - Danh sách ban đầu về các rủi ro
  - Đưa ra các use case
  - Lập kế hoạch cho pha khảo sát tỉ mỉ
  - Phiên bản đầu tiên của các trường hợp nghiệp vụ

#### *Pha khảo sát tỉ mỉ*

- Mục đích của pha khảo sát tỉ mỉ là làm mịn những yêu cầu ban đầu
  - Làm mịn kiến trúc
  - Giám sát rủi ro và làm mịn độ ưu tiên của chúng
  - Làm mịn trường hợp nghiệp vụ
  - Đưa ra kế hoạch quản lý dự án
- Các hoạt động chính của pha khảo sát tỉ mỉ là làm mịn hoặc khảo sát tỉ mỉ các pha trước đó
- Những công việc của pha khảo sát tỉ mỉ bao gồm:
  - Tất cả những hoàn thành luồng công việc xác định yêu cầu
  - Thực hiện một cách trực quan luồng công việc phân tích toàn thể
  - Bắt đầu thiết kế kiến trúc
- Sản phẩm chuyển giao của pha khảo sát tỉ mỉ bao gồm:
  - Mô hình lĩnh vực hoàn thiện
  - Mô hình nghiệp vụ hoàn thiện
  - Các tài liệu xác định yêu cầu hoàn thiện
  - Các tài liệu phân tích hoàn thiện
  - Phiên bản cập nhật kiến trúc
  - Danh sách cập nhật các rủi ro
  - Kế hoạch quản lý dự án (cho những phần còn lại của dự án)
  - Trường hợp nghiệp vụ hoàn thiện

#### *Pha xây dựng*

- Mục đích của pha này là đưa ra phiên bản chất lượng – sẵn sàng hoạt động đầu tiên của sản phẩm phần mềm
  - Đôi khi gọi đây là sự phát hành phiên bản beta
- Pha này tập trung vào những công việc
  - Cài đặt và
  - Kiểm thử

- Kiểm thử đơn vị của các mô đun
- Kiểm thử tích hợp của các hệ thống con
- Kiểm thử sản phẩm của toàn hệ thống
- Sản phẩm chuyển giao của pha xây dựng bao gồm:
  - Sổ tay người dùng ban đầu và các sổ tay khác
  - Tất cả các tài liệu được tạo ra (các phiên bản phát hành beta)
  - Kiến trúc hoàn thiện
  - Danh sách rủi ro đã được cập nhật
  - Kế hoạch quản lý dự án (Cho những phần còn lại của dự án)
  - Nếu cần thiết cập nhật trường hợp nghiệp vụ

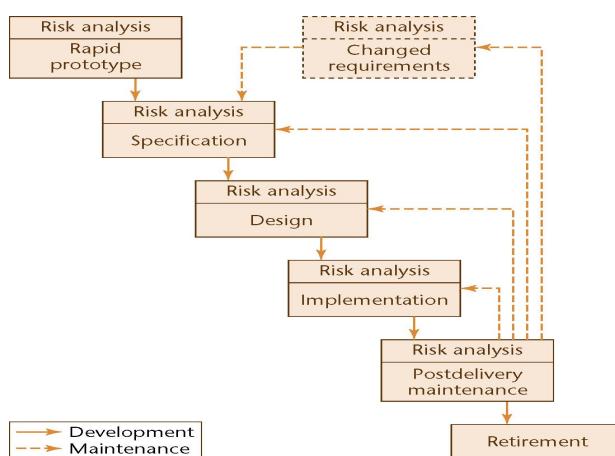
#### *Pha chuyển tiếp*

- Mục đích của pha chuyển tiếp là đảm bảo yêu cầu của khách hàng được đáp ứng
  - Lỗi trong sản phẩm phần mềm được sửa
  - Tất cả các sổ tay được hoàn thiện
  - Có gắng tìm ra những rủi ro mà trước đó chưa được nhận dạng
- Pha này hướng tới những phản hồi từ phía mà phát hành beta được cài đặt
- Sản phẩm chuyển giao của pha chuyển tiếp bao gồm:
  - Tất cả các tài liệu được tạo ra (các phiên bản cuối cùng)
  - Các sổ tay hoàn thiện

### 3.7 MÔ HÌNH XOĂN ỐC

Là mô hình bản mẫu nhanh kết hợp với phân tích rủi ro đi kèm ở mỗi pha

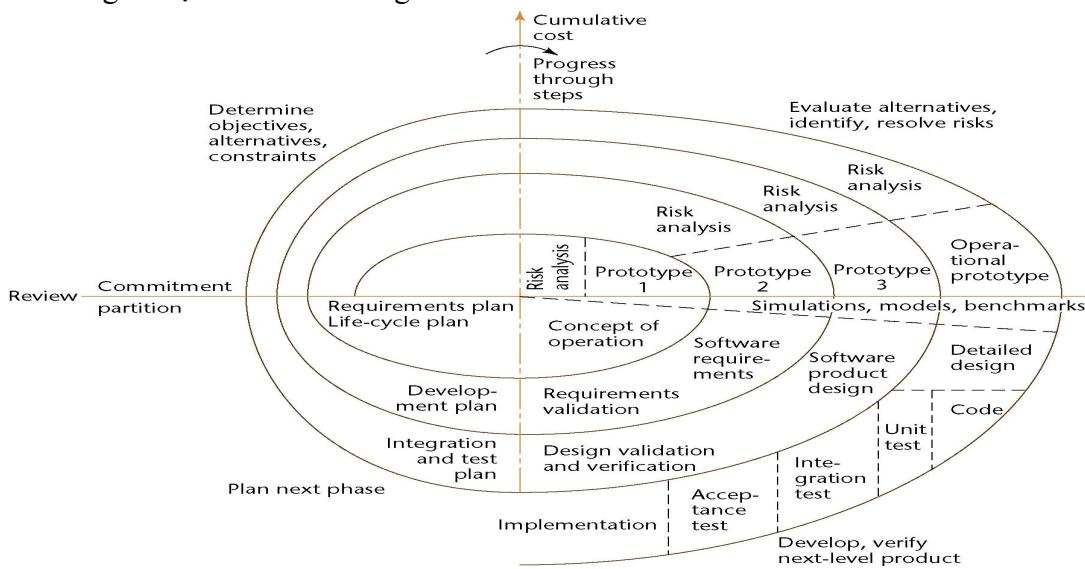
Đặc điểm chính của mô hình xoắn ốc: Nếu các rủi ro không được làm giảm bớt, thì dự án bị kết thúc ngay lập tức.



Mô hình xoắn ốc đầy đủ:

- Ở trước mỗi pha là các giải pháp và phân tích rủi ro
- Theo sau mỗi pha là: Đánh giá và lập kế hoạch cho pha tiếp theo
- Chiều bán kính: Chi phí tích lũy cho đến hiện tại

- Chiều góc: sự đi lên theo vòng ốc



Điểm mạnh của mô hình xoắn ốc:

- Dễ dàng thay đổi mức độ kiểm thử
- Không phân biệt giữa phát triển và bảo trì

Điểm yếu của mô hình xoắn ốc:

- Chỉ phù hợp với những phần mềm lớn
- Chỉ phù hợp với những phần mềm nội bộ

### 3.8 MÔ HÌNH MÃ NGUỒN MỞ

- Hai pha không hình thức
- Đầu tiên, một các nhân xây dựng một phiên bản đầu tiên
  - Đưa phiên bản này lên Internet (ví dụ: SourceForge.net)
- Sau đó, nếu có sự quan tâm về dự án này
  - Phiên bản đầu tiên sẽ được tải về một cách rộng rãi
  - Người dùng trở thành người đồng phát triển
  - Sản phẩm được mở rộng
- Điểm chính: Các cá nhân nhàn chung làm việc tình nguyện đối với dự án mã nguồn mở trong thời gian rảnh rỗi của họ
- Các hoạt động trong pha phi hình thức thứ hai
  - Báo cáo và sửa lỗi
    - Bảo trì sửa lỗi
    - Thêm các chức năng mới
    - Bảo trì hoàn thiện chức năng
    - Điều chỉnh phần mềm ở môi trường mới
    - Bảo trì thích hợp
    - Pha phi hình thức thứ hai chỉ bao gồm bảo trì sau khi đã chuyển giao sản phẩm

- Từ “người đồng phát triển” trong slide trước có thể thay thế bằng từ “đồng bảo trì”
- Mô hình vòng đời bảo trì sau khi chuyển giao sản phẩm



- Sản phẩm mã nguồn đóng được bảo trì và kiểm thử bởi nhân viên
  - Người dùng có thể đưa ra bản tường trình sự thất bại của phần mềm (failure) nhưng không bao giờ đưa ra được lỗi mã nguồn (fault) (vì mã nguồn không sẵn có)
- Phần mềm mã nguồn mở nhìn chung được bảo trì bởi người tình nguyện viên không lương
  - Người dùng được khuyến khích đưa ra bản tường trình về sự thất bại của phần mềm (failure) cũng như lỗi mã nguồn (fault)
- Nhóm chính
  - Số lượng nhỏ những người bảo trì tận tụy với sở thích, thời gian và những kỹ năng cần thiết để đưa ra những báo cáo lỗi mã nguồn đã được cố định (“fixes”）
  - Họ chịu trách nhiệm quản lý dữ án
  - Họ có quyền cài đặt lại những lỗi đã cố định
- Nhóm ngoại vi
  - Users who choose to submit defect reports from time to time
- Các phiên bản mới của phần mềm mã nguồn đóng được phát hành thường xuyên một năm một lần
  - Sau khi đã được kiểm thử cẩn thận bởi nhóm quản lý chất lượng phần mềm
- Các phát hành của nhóm chính của một phiên bản mới của sản phẩm mã nguồn mở được đưa ra ngay khi sẵn sàng
  - Có thể là một tháng hoặc thậm chí là một ngày so với phiên bản trước đó
  - Nhóm chính thực hiện kiểm thử tối thiểu
  - Kiểm thử mở rộng được thực hiện bởi các thành viên của nhóm ngoại vi trong suốt thời gian sử dụng phần mềm
  - “Phát hành sớm và thường xuyên”
- Phiên bản làm việc đầu tiên được đưa ra khi sử dụng:
  - Mô hình bản mẫu nhanh;
  - Mô hình xây và sửa; và
  - Mô hình mã nguồn mở

- Sau đó:
  - Mô hình bản mẫu nhanh
    - Phiên bản đầu tiên bị bỏ qua
  - Mô hình xây và sửa và mô hình vòng đời mã nguồn mở
    - Phiên bản ban đầu trở thành phiên bản đích
- Do đó, trong dự án mã nguồn mở, nhình chung không có đặc tả và không có thiết kế?
- Một số dự án mã nguồn mở đã thành công như thế nào khi không có đặc tả hoặc thiết kế?
- Sản phẩm mã nguồn mở đã thu hút được một số chuyên gia phần mềm tốt nhất trên thế giới
  - Họ có thể xây dựng các chức năng hiệu quả mà không cần đặc tả hoặc thiết kế
- Tuy nhiên, cuối cùng thì sự phát triển sản phẩm mã nguồn mở cũng dừng lại khi mà nó không có sự bảo trì nữa
- Mô hình vòng đời mã nguồn mở bị hạn chế trong khả năng ứng dụng của nó
- Nó có thể thành công cực độ đối với những dự án cơ sở hạ tầng như
  - Hệ điều hành (Linux, OpenBSD, Mach, Darwin)
  - Trình duyệt Web (Firefox, Netscape)
  - Trình biên dịch (gcc)
  - Máy chủ Web (Apache)
  - Hệ thống quản lý dữ liệu (MySQL)
- Không thể phát triển mã nguồn mở một sản phẩm phần mềm mà chỉ được sử dụng trong tổ chức thương mại
  - Các thành viên của cả nhóm chính và nhóm ngoại vi đều luôn là những người dùng của phần mềm đang được phát triển
- Mô hình vòng đời mã nguồn mở không thể áp dụng được trừ khi sản phẩm đích được xem xét bởi một số lượng lớn người dùng vì sản phẩm đó có ích cho họ.
- Khoảng một nửa dự án mã nguồn mở trên Web không thu hút được các nhóm phát triển vào làm việc cho dự án
- Even where work has started, the overwhelming preponderance will never be completed
- Nhưng khi mô hình mã nguồn mở đã làm việc thì đối khi nó cũng đem lại thành công đáng kinh ngạc
  - Những sản phẩm mã nguồn mở được kể ở slide trước đã được sử dụng bởi hàng triệu người dùng

### 3.9 CÂU HỎI ÔN TẬP

1. Thế nào là "from scratch"?
2. Thế nào là một episode?
3. Thế nào là một artifact?
4. Luật Brooks trong CNPM nói gì?

5. Nếu không áp dụng các mô hình vòng đời phần mềm thì có phát triển được phần mềm không? Tại sao?
6. Tại sao người ta phải dùng nhiều mô hình vòng đời khác nhau để phát triển phần mềm?
7. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu thác nước?
8. Mô hình vòng đời phần mềm kiểu thác nước thì phù hợp với những dự án có đặc điểm gì?
9. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu bản mẫu nhanh?
10. Mô hình vòng đời phần mềm kiểu bản mẫu nhanh thì phù hợp với những dự án có đặc điểm gì?
11. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu lặp và tăng trưởng?
12. Mô hình vòng đời phần mềm kiểu lặp và tăng trưởng thì phù hợp với những dự án có đặc điểm gì?
13. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu xoắn ốc?
14. Mô hình vòng đời phần mềm kiểu xoắn ốc thì phù hợp với những dự án có đặc điểm gì?
15. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa?
16. Mô hình vòng đời phần mềm kiểu ổn định và đồng bộ hóa thì phù hợp với những dự án có đặc điểm gì?
17. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu mã nguồn mở?
18. Mô hình vòng đời phần mềm kiểu mã nguồn mở thì phù hợp với những dự án có đặc điểm gì?
19. Nếu ưu điểm, nhược điểm của mô hình vòng đời phần mềm kiểu tiến trình linh hoạt?
20. Mô hình vòng đời phần mềm kiểu tiến trình linh hoạt thì phù hợp với những dự án có đặc điểm gì?
21. Tại sao trong mô hình tiến trình linh hoạt, không cần có pha đặc tả?
22. Trong mô hình tiến trình liên hoạt, luôn có đại diện của khách hàng trong nhóm phát triển thì có ưu điểm gì?
23. Nếu ưu điểm, nhược điểm của kỹ thuật pair programming?
24. Nếu ưu điểm, nhược điểm của kỹ thuật time boxing?
25. Nếu ưu điểm, nhược điểm của kỹ thuật stand up meeting?

## CHƯƠNG 4: KIỂM THỬ

### 4.1 VẤN ĐỀ CHẤT LƯỢNG PHẦN MỀM

- Phần mềm thỏa mãn ở một mức độ nào đó các đặc tả của nó
- Mỗi chuyên gia phần mềm có trách nhiệm đảm bảo công việc của họ là chính xác
  - Chất lượng phải được xây dựng từ ban đầu

#### 4.1.1 Đảm bảo chất lượng phần mềm (SQA)

- Các thành viên của nhóm SQA phải đảm bảo những người phát triển thực hiện công việc đạt chất lượng cao
  - Ở cuối mỗi luồng công việc
  - Khi sản phẩm được hoàn thiện
- Thêm vào đó, đảm bảo chất lượng phải được áp dụng
  - Trong mỗi tiến trình
    - **Ví dụ: các chuẩn**

#### 4.1.2. Độc lập trong quản lý

- Phải độc lập về quản lý giữa
  - Nhóm phát triển
  - Nhóm SQA
- Mỗi nhóm không được phép vượt quá quyền hạn sang các nhóm khác
- Quản lý lâu năm hơn phải quyết định liệu
  - Chuyển giao sản phẩm đúng thời hạn nhưng có lỗi xảy ra
  - Hoặc kiểm thử hơn nữa và chuyển giao sản phẩm muộn hơn
- Những quyết định phải xem xét đến sự quan tâm của khách hàng và tổ chức phát triển

### 4.2 KIỂM CHỨNG PHẦN MỀM

- Có hai kiểu : Kiểm thử không có sự thực thi và kiểm thử có dựa trên sự thực thi

- “V & V”
  - *Xác minh (Verification)*
    - Xác định nếu luồng công việc được hoàn thiện chính xác Determine if the workflow was completed correctly
  - *Thảm định (Validation)*
    - Xác định nếu toàn bộ sản phẩm đáp ứng các yêu cầu đưa ra (Determine if the product as a whole satisfies its requirements)
  - Thuật ngữ “xác minh” cũng được sử dụng cho kiểm thử không có sự thực thi.

### 4.3 CÁC PHƯƠNG PHÁP KIỂM CHỨNG

#### 4.3.1. Kiểm thử không có sự thực thi

- Nguyên lý cơ bản
  - Chúng ta không nên tự mình rà soát công việc của mình
  - Có sự hiệp lực trong nhóm

##### 4.3.1.1 Rà soát lướt qua (Walkthrough)

- Đội rà soát lướt qua bao gồm từ 4 đến 6 thành viên
- Nó bao gồm đại diện của
  - Đội chịu trách nhiệm cho luồng công việc hiện thời
  - Đội chịu trách nhiệm cho luồng công việc tiếp thep
  - Nhóm SQA
- Rà soát lướt qua được nói trước bởi sự chuẩn bị
  - Danh sách các mục
    - Các mục không hiếu
    - Các mục mà xuất hiện ở dạng lỗi

Quản lý rà soát lướt qua:

- *Đội* rà soát lướt qua được đại diện SQA làm chủ trì
- Trong rà soát lướt qua chỉ phát hiện lỗi, không sửa

- Việc sửa lỗi được thực hiện bởi một ủy ban và có khả năng dẫn đến chất lượng thấp
- Chi phí sửa lỗi của ủy ban là quá cao
- Không phải tất cả các hạng mục được đánh dấu thực tế đều sai
- Rà soát lướt qua không nên kéo dài quá 2 tiếng
- Cũng như không có thời gian sửa lỗi

#### 4.3.1 .2 Kiểm tra kỹ lưỡng (Inspection)

- Quá trình kiểm tra kỹ lưỡng gồm 5 bước hình thức
  - Tổng quan
  - Chuẩn bị, với mục đích thống kê các loại lỗi
  - Kiểm tra kỹ lưỡng
  - Sửa lỗi
  - Theo dõi
- Đội kiểm tra gồm 4 thành viên
  - Người điều tiết (Moderator)
  - Một thành viên trong đội thực đang thi luồng công việc hiện hành
  - Một thành viên của đội sẽ thực hiện luồng công việc tiếp theo
  - Một thành viên của nhóm SQA
- Những vai trò đặc biệt được đảm nhiệm bởi
  - Người điều tiết (Moderator)
  - Người đọc (Reader)
  - Người ghi chép (Recorder)

Thông kê lỗi:

- Các lỗi được ghi chép lại cẩn thận
  - Ví dụ:

- Chính hoặc phụ
- Các lỗi được ghi chép theo các kiểu lỗi
  - Ví dụ các lỗi thiết kế:
    - Không phải tất cả các mục đã được bàn bạc
    - Các lý lẽ hình thức và thực tế không tương ứng với nhau
- Đối với mỗi luồng công việc xác định, chúng ta so sánh tỷ lệ lỗi hiện thời với tỷ lệ lỗi của những sản phẩm trước
- Nếu thấy số lượng lỗi không cân xứng với tài liệu
  - Thiết kế lại từ ban đầu là cách tốt nhất
- Thực hiện chuyển những thông kê lỗi tới luồng công việc tiếp theo
  - Chúng ta có thể không phát hiện được tất cả các lỗi của mỗi kiểu cụ thể trong quá trình kiểm tra kỹ lưỡng hiện tại

#### Thống kê quá trình kiểm tra kỹ lưỡng

- Quá trình kiểm tra kỹ lưỡng IBM đã chỉ ra:
  - 82% lỗi được phát hiện (1976)
  - 70% lỗi được phát hiện (1978)
  - 93% lỗi được phát hiện (1986)
- Chuyển hệ thống (Switching system)
  - Giảm 90% chi phí trong việc phát hiện lỗi (1986)
- JPL
  - 4 lỗi chính, 14 lỗi phụ trong 2 giờ (1990)
  - Tiết kiệm \$25,000 trong mỗi quá trình kiểm tra kỹ lưỡng
  - Số lượng lỗi đã giảm theo hàm số mũ trong mỗi pha (1992)
- Cảnh báo
- Thống kê lỗi chưa bao giờ được sử dụng để đánh giá hiệu năng

- “Killing the goose that lays the golden eggs”

Công cụ cho kiểm tra kỹ lưỡng :

- Tỷ lệ kiểm tra kỹ lưỡng (ví dụ: số trang thiết kế được kiểm tra trên 1 giờ)
- Mật độ lỗi (ví dụ: số lỗi trên nghìn dòng lệnh (KLOC) được kiểm tra)
- Tỷ lệ phát hiện lỗi ( ví dụ: số lỗi được phát hiện trên 1 giờ)
- Hiệu năng phát hiện lỗi ( ví dụ: số lượng những lỗi chính, lỗi phụ được phát hiện trong 1 giờ)
- Có phải tăng 50% tỷ lệ phát hiện lỗi thì đồng nghĩa với
  - Chất lượng giảm? Hoặc
  - Quá trình kiểm tra kỹ lưỡng hiệu quả hơn?

#### 4.3.1.3 So sánh giữa rà soát lướt qua và kiểm tra kỹ lưỡng

- Lướt qua
  - Hai bước, tiến trình không hình thứcprocess
    - Chuẩn bị
    - Phân tích
- Kiểm tra kỹ lưỡng
  - 5 bước, tiến trình hình thức
    - Tổng quan
    - Chuẩn bị
    - Kiểm tra kỹ lưỡng
    - Sửa lỗi
    - Theo dõi

#### 4.3.1.4 Những điểm mạnh và điểm yếu của các loại rà soát

- Rà soát có thể rất hiệu
  - Các lỗi được phát hiện sớm ở trong quá trình phát triển phần mềm
- Rà soát ít hiệu quả nếu quá trình không đầy đủ

- Những phần mềm phạm vi lớn bao gồm những phần nhỏ và độc lập với nhau
- Tài liệu của những luồng công việc trước phải được hoàn thiện và luôn có sẵn

### 4.3.2 Kiểm thử có dựa trên sự thực thi

- Các tổ chức sử dụng tới 50% ngân sách dự án để kiểm thử
  - Nhưng phần mềm được chuyển giao thường xuyên xảy ra lỗi
- Dijkstra (1972)
  - “Kiểm chương trình có thể là cách rất hiệu quả để chỉ ra các lỗi nhưng không có khả năng để chỉ ra sự vắng mặt của chúng”. (“Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence”)

## 4.4 NHỮNG VẤN ĐỀ TRONG KIỂM THỬ

### 4.4.1 Cái gì nên được kiểm thử?

Chúng ta cần kiểm thử tính chính xác, tiện ích, tính đáng tin, tính mạnh mẽ và hiệu năng

#### 4.4.1.1 Tính chính xác

- Một phần mềm chạy chính xác nếu nó thỏa mãn đặc tả của nó
- Tính chính xác của các đặc tả
  - Đặc tả không chính xác đối với một yêu cầu sắp xếp:

*Input specification:*       $p : \text{array of } n \text{ integers}, n > 0.$

*Output specification:*       $q : \text{array of } n \text{ integers such that}$   
 $q[0] \leq q[1] \leq \dots \leq q[n - 1]$

- Hàm trickSort đã thỏa mãn đặc tả này:

```
void trickSort (int p[ ], int q[ ])
{
    int i;
    for (i = 0; i < n; i++)
        q[i] = 0;
}
```

- Đặc tả chính xác:

*Input specification:*  $p$  : array of  $n$  integers,  $n > 0$ .

*Output specification:*  $q$  : array of  $n$  integers such that  
 $q[0] \leq q[1] \leq \dots \leq q[n - 1]$

The elements of array  $q$  are a permutation of the elements of array  $p$ , which are unchanged.

- o Về mặt kỹ thuật, tính chính xác là không cần thiết (chẳng hạn như trình biên dịch C++) và không đầy đủ (ví dụ như : *trickSort*)

#### 4.4.1.2 Tiện ích

- Sản phẩm phần mềm đáp ứng ở một mức nào đó yêu cầu của người dùng
  - o Ví dụ:
    - Tính dễ sử dụng
    - Các chức năng hữu ích
    - Có hiệu quả trong chi phí

#### 4.4.1.3 Tính đáng tin

- Đo tầm suât và tính quan trọng của các thất bại (failure)
  - o Thời gian trung bình giữa những lần chức năng không thực hiện
  - o Thời gian trung bình để sửa
  - o Thời gian và chi phí để sửa chữa kết quả của một thất bại trong hệ thống phần mềm

#### 4.4.1.4 Tính mạnh mẽ

- Một chức năng của
  - o Một dãy các điều kiện hoạt
  - o Khả năng các kết quả không được chấp nhận với đầu vào hợp lệ
  - o Ảnh hưởng của đầu vào không hợp lệ

#### 4.4.1.5 Hiệu năng

- Những ràng buộc về thời gian và không gian được đáp ứng ở một mức độ nào đó
- Phần mềm thời gian thực được đặc trưng bởi những ràng buộc thời gian thực
- Nếu dữ liệu bị mất bởi vì hệ thống quá chậm

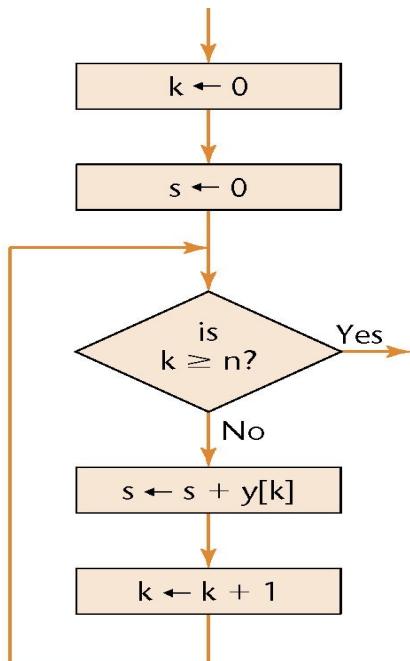
- o Không có cách để phục hồi dữ liệu đó

#### 4.4.2 Kiểm thử và sự kiểm chứng tính chính xác

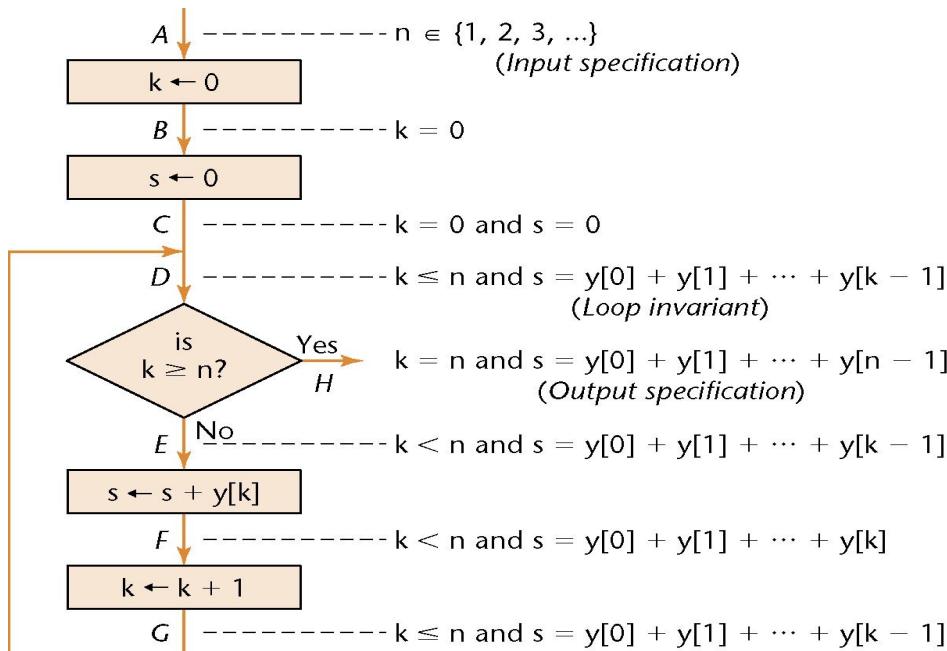
- Sự kiểm chứng tính chính xác là một phương pháp khác đối với kiểm thử có dựa trên sự thực thi
- Ví dụ về sự kiểm chứng tính chính xác
  - o Đoạn code đã chứng minh sự chính xác

```
int k, s;
int y[n];
k = 0;
s = 0;
while (k < n)
{
    s = s + y[k];
    k = k + 1;
}
```

- o Biểu đồ luồng tương ứng với đoạn code trên



- o Thêm vào : đặc tả đầu vào, đặc tả đầu ra, số lượng vòng lặp và sự xác nhận



#### 4.4.3 Sự kiểm chứng tính chính xác và kỹ nghệ phần mềm

- Ba chuyện tưởng tượng trong chứng minh tính chính xác
  - Kỹ sư phần mềm không đủ kiến thức toán học để kiểm chứng
    - Hầu hết các ngành khoa học máy tính hoặc biết hoặc có thể học toán học đã yêu cầu cho kiểm chứng
  - Việc chứng minh yêu cầu chi phí rất cao trong thực tế
    - Khả năng kinh tế được xác định từ phân tích lợi nhuận và chi phí
  - Việc chứng minh quá khó khăn
    - Nhiều sản phẩm phần mềm không bình thường đã được chứng minh thành công
    - Các công cụ giống như chứng minh định lý có thể sẽ hữu ích
- Những khó khăn khi chứng minh tính chính xác
  - Chúng ta có thể tin vào cách chứng minh định lý không?

```

void theoremProver ( )
{
    print "This product is correct";
}
    
```

- Chúng ta tìm các đặc tả đầu vào, đầu ra, số lượng vòng lặp như thế nào?

- Chuyện gì sẽ xảy ra nếu các đặc tả là sai?
- Chúng ta có thể chưa bao giờ chắc chắn rằng các đặc tả hoặc hệ thống xác minh là chính xác
- Sự kiểm chứng tính chính xác là một công cụ kỹ nghệ phần mềm quan trọng, thích hợp với:
  - Khi cuộc sống con người đang lâm nguy (When human lives are at stake)
  - Khi đã được chỉ ra bởi phân tích lợi nhuận – chi phí (When indicated by cost-benefit analysis)
  - Khi rủi ro của việc không phân tích quá lớn (When the risk of not proving is too great)
- Sự kiểm chứng không hình thức có thể cải thiện chất lượng của phần mềm
  - Sử dụng những câu lệnh assert

#### 4.4.4 Ai thực hiện kiểm thử phần mềm

- Lập trình là xây dựng
- Kiểm thử là phá huỷ
  - Một kiểm thử thành công tìm ra một lỗi
- Vì thế, những người lập trình không nên kiểm thử chính tài liệu mã của họ viết
- Giải pháp:
  - Người lập trình thực hiện kiểm thử không hình thức
  - Sau đó nhóm SQA thực hiện kiểm thử một cách hệ thống
  - Người lập trình gỡ lỗi mò đun đó
- Tất cả các trường hợp kiểm thử phải
  - Được lập kế hoạch bằng tay trước đó, bao gồm đầu ra mong muốn và
  - Được giữ về sau

#### 4.4.5 Khi nào kiểm thử dừng

- Chỉ khi sản phẩm phần mềm không thể thay đổi (Only when the product has been irrevocably discarded)

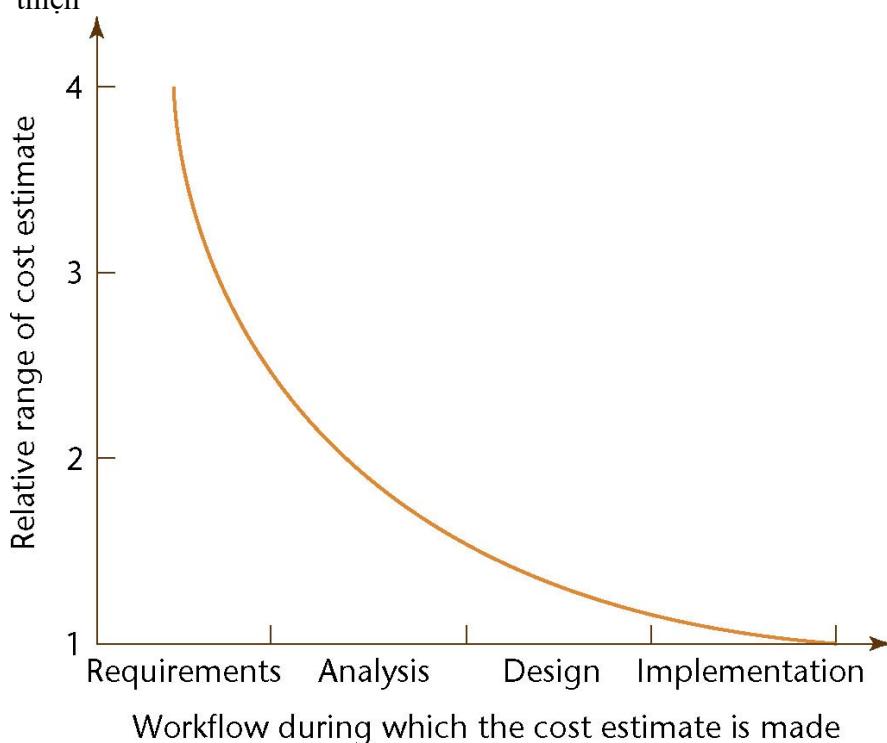
### 4.5 CÂU HỎI ÔN TẬP

1. alpha release là gì? Khác gì với beta release?
2. beta release là gì? Khác gì với alpha release?
3. Luật Dijkstra trong CNPM nói gì?
4. Thẩm định và xác minh là gì (V&V)? Tầm quan trọng của chúng?
5. Có những loại V&V nào? Mô tả nội dung mỗi loại.
6. inspection là gì? Khác gì với walkthrough?
7. walkthrough là gì? Khác gì với inspection?
8. Tại sao trong nhóm walkthrough và inspection, luôn phải có đại diện của workflow tiếp theo?
9. Người ta nói « nhóm SQA tạo ra chất lượng cho phần mềm » đúng hay sai? Tại sao?
10. Phần mềm còn lỗi hay không khi thực hiện hết các ca kiểm thử được thiết kế? Khi nào dừng kiểm thử?
11. Trình bày phương pháp kiểm thử hộp trắng: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.
12. Trình bày phương pháp kiểm thử hộp đen: cơ sở phương pháp; các yêu cầu cần kiểm tra, các kỹ thuật được sử dụng.
13. Kiểm thử đơn vị đối tượng là gì? Ai thực hiện. Các phương pháp và kỹ thuật nào được sử dụng? Kiểm tra những loại lỗi nào?
14. Các chiến lược nào sử dụng trong kiểm thử tích hợp? Ưu điểm và hạn chế mỗi loại?
15. Giải thích khái niệm stub và driver? Chúng được sử dụng ở đâu và vì sao?
16. Kiểm thử hệ thống nhằm kiểm tra cái gì? Ai thực hiện? Các phương pháp?
17. Trình bày các kiểm thử được thực hiện trong kiểm thử hệ thống?
18. Kiểm thử chấp nhận là gì? Trong đó có những kiểm thử nào được thực hiện? Phân biệt.

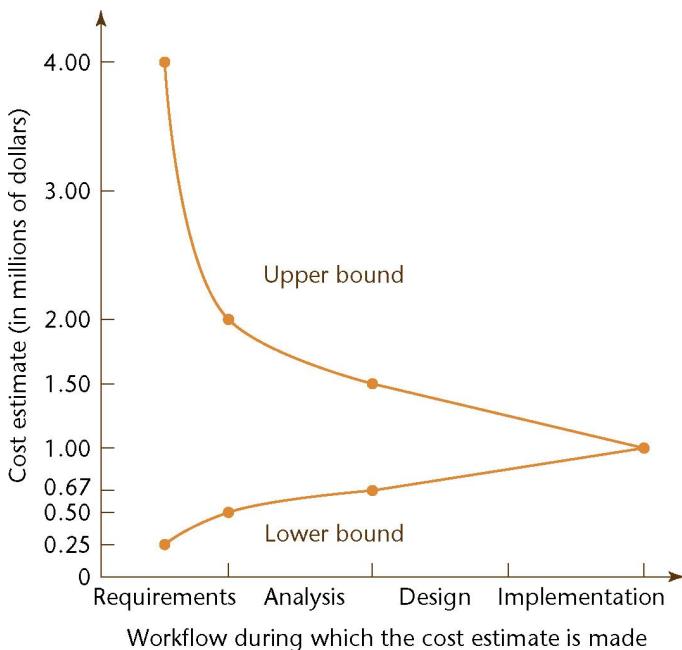
## CHƯƠNG 5: LẬP KẾ HOẠCH VÀ ƯỚC LUỢNG

### 5.1 VẤN ĐỀ LẬP KẾ HOẠCH VÀ ƯỚC LUỢNG DỰ ÁN PHẦN MỀM

- Trước khi bắt đầu xây dựng phần mềm, việc lập kế hoạch cho toàn bộ quá trình phát triển một cách chi tiết là cần thiết
- Việc lập kế hoạch được thực hiện liên tục trong suốt quá trình phát triển và bảo trì sau khi chuyển giao
  - Việc lập kế hoạch ban đầu là không đủ
  - Việc lập kế hoạch phải được thực hiện trong suốt dự án
  - Việc lập kế hoạch chi tiết có thể diễn ra sớm nhất có thể sau khi các đặc tả hoàn thiện



- Dộ chính xác của ước lượng tăng khi tiến trình phần mềm thực hiện được nhiều
- Ví dụ
  - Chi phí ước lượng 1 triệu đô la trong suốt luồng công việc xác định yêu cầu
    - Chi phí thực tế có thể nằm trong khoảng (\$0.25M, \$4M)
  - Chi phí ước lượng 1 triệu đô la ở cuối luồng công việc xác định yêu cầu
    - Chi phí thực tế có thể nằm trong khoảng (\$0.5M, \$2M)
  - Chi phí ước lượng của 1 triệu đô la ở cuối luồng công việc phân tích (thời gian thích hợp sớm nhất)
    - Chi phí thực có thể nằm trong khoảng (\$0.67M, \$1.5M)



- Mô hình là cũ (1976)
  - Kỹ thuật ước lượng đã cải tiến
  - Nhưng hình dạng của đường cong này là giống với ban đầu

## 5.2 UỚC LƯỢNG THỜI GIAN VÀ CHI PHÍ

- Uớc lượng thời gian chính xác là cần thiết
- Uớc lượng chi phí chính xác là cần thiết
  - Chi phí bên trong, bên ngoài
- Có quá nhiều thay đổi trong việc ước lượng chính xác chi phí hoặc thời gian

Nhân lực

- Sackman (1968) đo sự khác biệt lên đến 28-1 giữa các cặp lập trình viên(Sackman (1968) measured differences of up to 28 to 1 between pairs of programmers )
  - Ông so sánh sự kết hợp của các cặp lập trình viên êm mặt
    - Kích cỡ phần mềm
    - Thời gian thực thi sản phẩm
    - Thời gian phát triển
    - Thời gian viết mã
    - Thời gian gỡ lỗi
- Những thành viên nhân viên cốt yếu có thể từ bỏ trong suốt dự án

### 5.2.1 Thước đo kích cỡ của sản phẩm phần mềm

$\alpha$ - Số lượng dòng mã (LOC, KDSI, KLOC)

- Một thước đo khác
  - Số dòng mã nguồn (KDSI)
- Mã nguồn chỉ là một phần nhỏ của nỗ lực phát triển toàn bộ phần mềm (Source code is only a small part of the total software effort )
- Những ngôn ngữ khác nhau dẫn đến chiều dài mã lệnh khác nhau
- LOC không được dùng để xác định ngôn ngữ phi thủ tục (như LISP)

- Các điểm dòng lệnh không rõ ràng
  - Số dòng lệnh có thể thực thi?
  - Những định nghĩa dữ liệu?
  - Những lời chú thích?
  - Các câu lệnh JCL?
  - Dòng lệnh đã được thay đổi/xóa?
- Không phải mọi thứ được viết ra đều chuyên giao cho khách hàng
- Bộ sinh bản ghi, màn ảnh, GUI có thể sinh hàng nghìn dòng lệnh trong mỗi phút
- LOC chỉ được biết chính xác khi phần mềm được hoàn thiện
- Do đó, ước lượng dựa trên LOC nguy hiểm gấp đôi
  - Để bắt đầu tiên trình ước lượng, LOC trong phần mềm đã hoàn thiện phải được ước lượng
  - Sau đó, ước lượng LOC được sử dụng để ước lượng chi phí của phần mềm

### β– FFP

- Được sử dụng đối với ước lượng chi phí của một phần mềm xử lý dữ liệu cỡ trung bình
- Ba thành phần cấu trúc cơ bản của phần mềm xử lý dữ liệu
  - Các tệp
  - Các luồng
  - Các tiến trình
- Với số lượng tệp ( $Fi$ ), số luồng ( $Fl$ ), và số tiến trình ( $Pr$ )
  - Kích thước (s), chi phí (c) được xác định bởi

$$S = Fi + Fl + Pr$$

$$C = b \times S$$

- Hằng số  $b$  (hiệu năng hoặc năng suất) thay đổi từ tổ chức này đến tổ chức khác
- Tính hiệu lực và tính tin cậy của thước đo FFP được chứng minh bằng cách sử dụng cùng một mục đích
  - Tuy nhiên, thước đo này không bao giờ được mở rộng đối với cơ sở dữ liệu

### γ– Điểm chức năng (Function Points)

- Dựa trên số lượng đầu vào ( $Inp$ ), đầu ra ( $Out$ ), các câu hỏi ( $Inq$ ), các tệp chính ( $Maf$ ), các giao diện ( $Inf$ )
- Đối với bất kỳ sản phẩm nào, số điểm chức năng được xác định bằng
  - $FP = 4 \times Inp + 5 \times Out + 4 \times Inq + 10 \times Maf + 7 \times Inf$
- Đây là một trường hợp quá đơn giản của một tiến trình 3 bước
- Bước 1. phân loại mỗi thành phần của phần mềm ( $Inp, Out, Inq, Maf, Inf$ ) thuộc loại đơn giản, trung bình, hoặc phức tạp
  - Gán số lượng thích hợp các điểm chức năng
  - The sum gives UFP (unadjusted function points)

Component	Level of Complexity		
	Simple	Average	Complex
Input item	3	4	6
Output item	4	5	7
Inquiry	3	4	6
Master file	7	10	15
Interface	5	7	10

- Bước 2. Tính toán nhân tố độ phức tạp về mặt kỹ thuật (technical complexity factor -  $TCF$ )
  - Gián giá trị từ 0 (không có mặt) tới 5 (ảnh hưởng mạnh mẽ từ đầu đến cuối) đối với mỗi nhân tố thuộc 14 nhân tố như tỷ giá giao dịch, tính khả chuyển

1. Data communication
2. Distributed data processing
3. Performance criteria
4. Heavily utilized hardware
5. High transaction rates
6. Online data entry
7. End-user efficiency
8. Online updating
9. Complex computations
10. Reusability
11. Ease of installation
12. Ease of operation
13. Portability
14. Maintainability

- Thêm 14 con số
  - Đưa ra mức độ ảnh hưởng tổng thể ( $DI$ )
 
$$TCF = 0.65 + 0.01 \times DI$$
- Nhân tố độ phức tạp về mặt kỹ thuật nằm trong khoảng từ 0.65 tới 1.35
- Bước 3. Số lượng điểm chức năng được tính bằng
 
$$FP = UFP \times TCF$$
- Phân tích về điểm chức năng
- Các điểm chức năng thường tốt hơn KDSI – nhưng có một vài vấn đề xảy ra

	Assembler Version	Ada Version
Source code size	70 KDSI	25 KDSI
Development costs	\$1,043,000	\$590,000
KDSI per person-month	0.335	0.211
Cost per source statement	\$14.90	\$23.60
Function points per person-month	1.65	2.92
Cost per function point	\$3,023	\$1,170

- Giống như FFP, bảo trì có thể được đo một cách chính xác
- Có thể thay đổi
  - Số lượng các tệp, luồng và các tiến trình
  - Số lượng đầu vào, đầu ra, câu hỏi, các tệp chính và các giao diện
- Theo lý thuyết, có thể thay đổi tất cả các dòng mã với việc thay đổi số lượng các dòng mã  
(In theory, it is possible to change every line of code with changing the number of lines of code)

#### Các điểm chúc năng Mk II

- Thước đo này đã được đưa ra để tính toán UFP một cách chính xác hơn
- Chúng ta chia nhỏ phần mềm thành các giao tác thành phần, mỗi giao tác thành phần gồm đầu vào, tiến trình và đầu ra
- Các điểm chúc năng Mark II được sử dụng rộng rãi trên thế giới

### δ– COCOMO

#### 5.2.2 Các kỹ thuật ước lượng chi phí

α– Những đánh giá của chuyên gia nhờ sự tương tự (Expert judgment by analogy)

- Các chuyên gia so sánh sản phẩm đích với những sản phẩm đã hoàn thiện
  - Sự ước chừng có thể dẫn tới ước lượng chi phí sai
  - Các chuyên gia có thể thu thập lại những thiếu sót của các phần mềm đã hoàn thiện
  - Các chuyên gia con người có nhiều xu hướng (Human experts have biases )
  - Tuy nhiên, kết quả của sự ước lượng bởi một nhóm lớn các chuyên gia có thể chính xác
- Kỹ thuật Delphi đôi khi được sử dụng để đạt được sự đồng thuận nhất trí

β– Phương pháp dưới lên

- Chia sản phẩm phần mềm thành những thành phần nhỏ hơn
  - Các thành phần nhỏ có thể không dễ ước lượng hơn
  - Tuy nhiên, có chi phí mức tiến trình
- Khi sử dụng mô hình hướng đối tượng
  - Sự độc lập của các lớp có ở đây (The independence of the classes assists here)
  - Tuy nhiên, những tương tác giữa các lớp làm rắc rối tiến trình ước lượng

γ– Các mô hình ước lượng chi phí thuật toán (Algorithmic cost estimation models )

- Một thước đo được sử dụng như đầu vào đối với mô hình để tính toán chi phí và thời gian
  - Mô hình thuật toán không thiên vị, do đó tốt hơn hẳn ý kiến chuyên gia

- Tuy nhiên, ước lượng chỉ tốt bằng những giả định tiềm ẩn (However, estimates are only as good as the underlying assumptions )
- Ví dụ
  - SLIM Model
  - Price S Model
  - COnstructive COst MOdel (COCOMO)

### 5.2.3 COCOMO trung gian

- COCOMO bao gồm 3 mô hình
  - Mô hình ước lượng vĩ mô đối với toàn bộ sản phẩm phần mềm
  - COCOMO trung gian
  - Mô hình ước lượng vi mô xem xét chi tiết phần mềm
- Chúng ta nghiên cứu COCOMO trung gian
- Bước 1. Ước lượng chiều dài của phần mềm trong KDSI
- Bước 2. Ước lượng chế độ phát triển phần mềm (có tổ chức (organic), nửa tách rời (semidetached), nhúng(embedded))
- Ví dụ:
  - Phần mềm không phức tạp (chế độ organic)
    - (Công sức danh nghĩa) Nominal effort =  $3.2 \cdot (KDSI)^{1.05}$  person-months
- Bước 3. Tính toán công sức danh nghĩa
- Ví dụ:
  - Phần mềm có tổ chức (Organic product)
  - 12,000 câu lệnh được chuyển giao (12 KDSI) (đã ước lượng)
  - (Công sức danh nghĩa) Nominal effort =  $3.2 \cdot (12)^{1.05} = 43$  person-months
- Bước 4. Nhân giá trị danh nghĩa với 15 lần chi phí phát triển phần mềm (Step 4. Multiply the nominal value by 15 software development cost multipliers )

Cost Drivers	Rating					
	Very Low	Low	Nominal	High	Very High	Extra High
<b>Product Attributes</b>						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Database size		0.94	1.00	1.08	1.16	
Product complexity	0.70	0.85	1.00	1.15	1.30	1.65
<b>Computer Attributes</b>						
Execution time constraint			1.00	1.11	1.30	1.66
Main storage constraint			1.00	1.06	1.21	1.56
Virtual machine volatility*		0.87	1.00	1.15	1.30	
Computer turnaround time		0.87	1.00	1.07	1.15	
<b>Personnel Attributes</b>						
Analyst capabilities	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Programmer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience*	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
<b>Project Attributes</b>						
Use of modern programming practices	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

\*For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system) it calls on to accomplish its task.

- Ví dụ:
  - Phần mềm xử lý giao tiếp dựa trên bộ vi xử lý cho mạng chuyển tiền điện tử với độ tin cậy, hiệu năng, lịch phát triển và các yêu cầu giao diện cao  
(Microprocessor-based communications processing software for electronic funds transfer network with high reliability, performance, development schedule, and interface requirements )
- Bước 1. Ước lượng chiều dài của phần mềm sản phẩm
  - 10,000 câu lệnh được chuyển giao (10 KDSI)
- Bước 2. Ước lượng chế độ phát triển
  - Chế độ phức tạp (“nhúng”-“embedded”)
- Bước 3. Tính công sức danh nghĩa
  - (Công sức danh nghĩa)Nominal effort =  $2.8 \cdot (10)^{1.20} = 44$  person-months
- Bước 4. Nhân giá trị danh nghĩa với 15 lần chi phí phát triển phần mềm
  - Product of effort multipliers = 1.35 (**Software development effort multipliers**)

Cost Drivers	Situation	Rating	Effort Multiplier
Required software reliability	Serious financial consequences of software fault	High	1.15
Data base size	20,000 bytes	Low	0.94
Product complexity	Communications processing	Very high	1.30
Execution time constraint	Will use 70% of available time	High	1.11
Main storage constraint	45K of 64K store (70%)	High	1.06
Virtual machine volatility	Based on commercial microprocessor hardware	Nominal	1.00
Computer turnaround time	2 hour average turnaround time	Nominal	1.00
Analyst capabilities	Good senior analysts	High	0.86
Applications experience	3 years	Nominal	1.00
Programmer capability	Good senior programmers	High	0.86
Virtual machine experience	6 months	Low	1.10
Programming language experience	12 months	Nominal	1.00
Use of modern programming practices	Most techniques in use over 1 year	High	0.91
Use of software tools	At basic minicomputer tool level	Low	1.10
Required development schedule	9 months	Nominal	1.00

- Do đó, Ước lượng công sức cho dự án  $1.35 \times 44 = 59$  person-months. Ước lượng công sức cho dự án (59 person-months) được sử dụng là đầu vào cho công thức bổ sung đối với
  - Chi phí đô la
  - Lịch biểu phát triển
  - Phân phối pha và hoạt động
  - Chi phí máy tính
  - Chi phí bảo trì hàng năm
  - Các mục liên quan
- COCOMO trung gian đã được xác nhận với một mẫu lớn (Intermediate COCOMO has been validated with respect to a broad sample )
- Giá trị thực nằm trong khoảng 20% giá trị dự đoán và khoảng 68% thời gian
  - COCOMO trung gian là phương thức ước lượng chính xác nhất về thời gian của nó
- Vấn đề chính
  - Nếu ước lượng số lượng dòng mã của sản phẩm đích là sai, thì mọi thứ đều sai

#### 5.2.4 COCOMO II

- 1995 đã mở rộng COCOMO 1981 với sự kết hợp
  - Hướng đối tượng
  - Mô hình vòng đời hiện đại
  - Bản mẫu nhanh
  - Ngôn ngữ thế hệ thứ tư
  - Phần mềm COTS
- COCOMO II phức tạp hơn nhiều so với phiên bản đầu
- Có 3 mô hình khác nhau

- Mô hình kết hợp ứng dụng đối với các pha ban đầu (**Application composition model for the early phases**)
  - Dựa trên các điểm đặc trưng (tương tự với điểm chức năng)
- Mô hình thiết kế sớm
  - Dựa trên các điểm chức năng
- Mô hình hậu kiến trúc (**Post-architecture model**)
  - Dựa trên các điểm chức năng hoặc KDSI
- Mô hình công sức COCOMO cơ bản là
  - $\text{effort} = a \times (\text{size})^b$
  - COCOMO trung gian
    - Ba giá trị đối với (a,b)
  - COCOMO II
    - *b thay đổi dựa vào giá trị của các biến xác định*
- COCOMO II hỗ trợ sử dụng lại
- COCOMO II has 17 multiplicative cost drivers (was 15)
  - Seven are new
- It is too soon for results regarding
  - The accuracy of COCOMO II
  - The extent of improvement (if any) over Intermediate COCOMO

### 5.2.5 Theo dõi ước lượng thời gian và chi phí

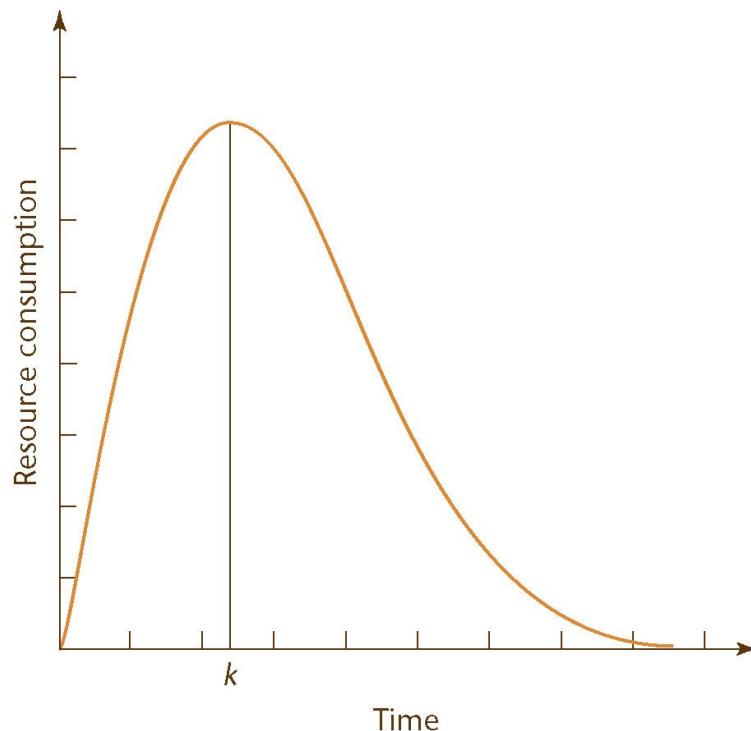
- Sử dụng bất cứ phương thức ước lượng nào thì việc theo dõi cũng là quan trọng
- Công việc được làm
  - Những tài nguyên để thực hiện công việc đó
- Số tiền để chi trả cho công việc đó

#### Các tài nguyên

- Các tài nguyên cần thiết cho phát triển phần mềm:
  - Con người
  - Phản ứng
  - Phần mềm hỗ trợ

#### Việc sử dụng các các loại tài nguyên với thời gian

- Đường cong Rayleigh miêu tả một cách chính xác việc giả định về tài nguyên
- Toute bộ kế hoạch phát triển phần mềm phải là một hàm thời gian



#### Các loại công việc

- Chức năng dự án
  - Công việc được thực thi xuyên suốt dự án
  - Ví dụ:
    - Quản lý dự án
    - Điều khiển chất lượng
- Hoạt động (activity)
  - Công việc liên quan tới một pha cụ thể
  - Đơn vị chính của công việc,
  - Với ngày tháng bắt đầu và kết thúc chính xác,
  - Giả định về thời gian và
  - Các sản phẩm công việc như ngân sách, thiết kế, lập lịch, mã nguồn, hoặc số tay người dùng
- Nhiệm vụ (task)
  - Các hoạt động bao gồm tập các nhiệm vụ (đơn vị nhỏ nhất của công việc có thể quản lý được)

#### Sự hoàn thành của các sản phẩm công việc

- *Mốc quan trọng (Milestone)*: là ngày mà sản phẩm công việc được hoàn thiện
- Nó phải chuyển qua và thực hiện rà soát bởi
  - Các thành viên trong đội
  - Quản lý
  - Khách hàng
- Khi sản phẩm công việc được rà soát và được đồng ý, thì nó trở thành một phiên bản cơ sở

#### Gói công việc

- Sản phẩm công việc, cùng với

- Biên chế các yêu cầu
- Thời gian
- Tài nguyên
- Gán trách nhiệm cho các cá nhân
- Tiêu chấp nhận sản phẩm công việc
- Ngân sách chi tiết là hàm thời gian, chỉ định
  - Các chức năng dự án
  - Các hoạt động

### 5.3 CÁC THÀNH PHẦN CỦA VIỆC LẬP KẾ HOẠCH PHÁT TRIỂN PHẦN MỀM

#### 5.3.1 Khung kế hoạch quản lý dự án phần mềm(SPMP)

- Có 3 cách để xây dựng SPMP
- Một cách tốt nhất là chuẩn IEEE 1058.1
  - Chuẩn được chấp nhận rộng rãi
  - Nó được thiết kế để sử dụng với tất cả các loại sản phẩm phần mềm
  - Nó hỗ trợ cải thiện tiến trình
    - Many sections reflect CMM key process areas
  - Là lý tưởng đối với quy trình hợp nhất
    - Có những phần để điều khiển các yêu cầu và quản lý rủi ro
- Một trong những phần không được áp dụng trong các phần mềm cỡ nhỏ
  - Ví dụ: kế hoạch quản lý nhà thầu phụ

#### 5.3.2 Kế hoạch quản lý dự án phần mềm IEEE

1 Overview	5.3 Control plan
1.1 Project summary	5.3.1 Requirements control plan
1.1.1 Purpose, scope, and objectives	5.3.2 Schedule control plan
1.1.2 Assumptions and constraints	5.3.3 Budget control plan
1.1.3 Project deliverables	5.3.4 Quality control plan
1.1.4 Schedule and budget summary	5.3.5 Reporting plan
1.2 Evolution of the project management plan	5.3.6 Metrics collection plan
2 Reference materials	5.4 Risk management plan
3 Definitions and acronyms	5.5 Project close-out plan
4 Project organization	6 Technical process plans
4.1 External interfaces	6.1 Process model
4.2 Internal structure	6.2 Methods, tools, and techniques
4.3 Roles and responsibilities	6.3 Infrastructure plan
5 Managerial process plans	6.4 Product acceptance plan
5.1 Start-up plan	7 Supporting process plans
5.1.1 Estimation plan	7.1 Configuration management plan
5.1.2 Staffing plan	7.2 Testing plan
5.1.3 Resource acquisition plan	7.3 Documentation plan
5.1.4 Project staff training plan	7.4 Quality assurance plan
5.2 Work plan	7.5 Reviews and audits plan
5.2.1 Work activities	7.6 Problem resolution plan
5.2.2 Schedule allocation	7.7 Subcontractor management plan
5.2.3 Resource allocation	7.8 Process improvement plan
5.2.4 Budget allocation	8 Additional plans

### 5.3.3 Việc lập kế hoạch kiểm thử

- SPMP phải chỉ ra rõ ràng những gì kiểm thử phải làm
- Việc theo dõi kiểm tra là cần thiết
- Tất cả các trường hợp kiểm thử hộp đen có thể được phác thảo ra sớm nhất có thể sau khi các đặc tả hoàn thiện

### 5.3.4 Yêu cầu đào tạo

- “Chúng ta không cần lo lắng về việc đào tạo đến tận khi phần mềm hoàn thiện, và sau đó chúng ta có thể đào tạo người dùng”
- Việc đào tạo nhìn chung được yêu cầu bởi các thành viên của nhóm phát triển, bắt đầu với đào tạo trong việc lập kế hoạch phần mềm
- Một phương thức phát triển phần mềm mới đòi hỏi phải đào tạo mọi thành viên trong nhóm phát triển
- Việc giới thiệu các công cụ phần cứng hoặc phần mềm của bất cứ loại nào đều đòi hỏi phải có đào tạo
- Những người lập trình có thể yêu cầu đào tạo về các hệ điều hành và/hoặc ngôn ngữ cài đặt
- Việc đào tạo chuẩn bị tài liệu có thể được yêu cầu
- Các thao tác máy tính đòi hỏi phải được đào tạo

### 5.3.5 Các chuẩn tài liệu

- Số lượng tài liệu được sinh ra bởi một phần mềm?
  - Phần mềm thương mại bên trong IBM (50 KDSI)
    - 28 trang tài liệu trên KDSI
  - Phần mềm thương mại cùng kích cỡ
    - 66 trang tài liệu trên KDSI
  - IMS/360 Version 2.3 (about 166 KDSI)
    - 157 trang tài liệu trên KDSI
  - [TRW] với 100 giờ sử dụng cho hoạt động viết mã, 150-200 giờ được sử dụng cho các hoạt động liên quan tới việc viết tài liệu
- Lập kế hoạch
- Điều khiển
- Tài chính
- Kỹ thuật
- Mã nguồn
- Những lời chú thích với mã nguồn

#### Ưu điểm của các chuẩn tài liệu

- Giảm thiểu lầm lẫn giữa các thành viên trong đội
- Trợ giúp SQA
- Chỉ những nhân viên mới phải học các chuẩn
- Các chuẩn trợ giúp những người lập trình bảo trì
- Việc chuẩn hóa là quan trọng đối với các số tay người dùng
- Là một phần của tiến trình lập kế hoạch
  - Các chuẩn phải được thiết lập đối với tất cả các tài liệu
- Sản phẩm là tài liệu

### 5.3.6 Công cụ CASE cho việc lập kế hoạch và ước lượng

- Rất cần thiết để có
  - Một bộ xử lý từ, và
  - Một bảng tính
- Hiện nay có công cụ COCOMO and COCOMO II
- Công cụ quản lý trợ giúp việc lập kế hoạch và giám sát
  - MacProject
  - Microsoft Project

## 5.4 KIỂM THỦ VIỆC LẬP KẾ HOẠCH

- Chúng ta phải kiểm tra toàn bộ SPMP
  - Đặc biệt chú ý tới ước lượng thời gian và chi phí

## 5.5 LẬP KẾ HOẠCH CHO CÁC DỰ ÁN PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- Phần mềm hướng đối tượng bao gồm các phần độc lập với nhau
- Do đó, việc lập kế hoạch phần nào dễ dàng hơn
- Việc lập kế hoạch toàn bộ nhiều hơn tổng việc lập kế hoạch của các
- Chúng ta có thể sử dụng COCOMO II
- Tuy nhiên, sử dụng lại bao gồm những sai sót trong ước lượng chi phí và thời gian
  - Sử dụng lại những thành phần sẵn có trong suốt quá trình phát triển
  - Sản phẩm của các thành phần để sử dụng lại trong tương lai
- Những công việc này theo các hướng đối nghịch nhau
- Dữ liệu mới hơn: việc lưu trữ vượt quá chi phí

## 5.6 CÂU HỎI ÔN TẬP

1. Thế nào là coin of uncertainty?
2. Thế nào là norminal effort?
3. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng LOC?
4. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng FFP?
5. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng Function Point?
6. Nêu ưu điểm, nhược điểm của phương pháp ước lượng phần mềm bằng COCOMO?



## CHƯƠNG 6: PHA XÁC ĐỊNH YÊU CẦU

### 6.1 XÁC ĐỊNH YÊU CẦU CỦA KHÁCH HÀNG

- Hiểu sai
  - Chúng ta phải xác định những gì khách hàng muốn
- “Tôi biết bạn tin rằng bạn đã hiểu những gì bạn nghĩ là tôi đã nói, nhưng tôi không chắc bạn nhận ra rằng những gì bạn nghe không phải là điều mà tôi muốn nói!” (“I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant!”)
- Chúng ta phải xác định những gì khách hàng cần
- Rất khó để người phân tích một hệ thống để hình dung ra một sản phẩm phần mềm và các chức năng của nó
  - Vấn đề này khó hỏi khách hàng
- Một người phân tích hệ thống có kinh nghiệm cần làm rõ những thông tin thích hợp cho khách hàng
- Khách hàng là nguồn duy nhất của thông tin này
- Giải pháp:
  - Thu thập những thông tin ban đầu từ khách hàng
  - Sử dụng những thông tin ban đầu giống như đầu vào của quy trình hợp nhất
  - Theo sát các bước của quy trình hợp nhất để xác định các nhu cầu thực của khách hàng

### 6.2 TỔNG QUAN VỀ LUỒNG CÔNG VIỆC XÁC ĐỊNH YÊU CẦU

Mục đích của luồng công việc xác định yêu cầu

- Để trả lời câu hỏi:
  - Sản phẩm phần mềm phải có khả năng làm được những gì?

Nội dung về luồng công việc xác định yêu cầu

- Đầu tiên, hiểu được lĩnh vực ứng dụng
  - Môi trường cụ thể mà sản phẩm phần mềm đích hoạt động
- Thứ hai, xây dựng một mô hình nghiệp vụ

- Mô hình các tiến trình nghiệp vụ của khách hàng
- Thứ ba, sử dụng mô hình nghiệp vụ để xác định các yêu cầu khách hàng
- Lặp lại các bước trên

Các định nghĩa

- Tìm ra các yêu cầu của khách hàng
  - Thu thập các yêu cầu
  - Các phương thức bao gồm phỏng vấn và điều tra
- Làm mịn và mở rộng những yêu cầu ban đầu
  - *Phân tích yêu cầu*

### 6.2.1 Hiểu lĩnh vực ứng dụng

- Mỗi thành viên của đội phát triển phải trở nên quen thuộc với lĩnh vực ứng dụng
  - Thuật ngữ chính xác là cần thiết
- Xây dựng thuật ngữ
  - Một danh sách các từ kỹ thuật được sử dụng trong lĩnh vực ứng dụng và ý nghĩa của nó

### 6.2.2 Mô hình nghiệp vụ

- Một mô hình nghiệp vụ là sự miêu tả các tiến trình nghiệp vụ của một tổ chức
- Mô hình nghiệp vụ đưa ra cách hiểu về toàn bộ nghiệp vụ của khách hàng
  - Tri thức này là cần thiết để đưa ra lời khuyên cho khách hàng về mặt tính toán
- Các nhà phân tích hệ thống cần thu thập một cách hiểu chi tiết về các loại tiến trình nghiệp vụ khác nhau.
  - Các kỹ thuật khác nhau được sử dụng, ban đầu là phỏng vấn

#### 6.2.2.1 Phỏng vấn

- Đội xác định yêu cầu cần gặp gỡ khách hàng và người dùng để thu thập được những thông tin liên quan
- Có hai loại câu hỏi:
  - *Câu hỏi kết thúc đóng (Close-ended question)* yêu cầu một câu trả lời cụ thể

- Câu hỏi kết thúc mở (Open-ended questions) khuyến khích người được phỏng vấn nói thẳng ý kiến của mình
- Có hai kiểu phỏng vấn
  - Trong cuộc phỏng vấn có cấu trúc, các câu hỏi đã được lập kế hoạch cụ thể từ trước và thường là những câu hỏi kết thúc đóng
  - Trong cuộc phỏng vấn không cấu trúc, các câu hỏi được đưa ra để phản ứng lại những câu trả lời đã nhận được, thường xuyên là câu hỏi với kết thúc mở
- Việc phỏng vấn là không dễ dàng
  - Một cuộc phỏng vấn mà không có cấu trúc sẽ không sinh ra thông tin liên quan
  - Người phỏng vấn phải quen thuộc với lĩnh vực ứng dụng
  - Người phỏng vấn phải sẵn sàng tiếp thu cái mới ở mọi lúc (The interviewer must remain open-minded at all times)
- Sau khi phỏng vấn, người phỏng vấn phải chuẩn bị một bản tường trình đã được viết ra
  - Nên đưa một bản sao của bản tường trình cho người được phỏng vấn

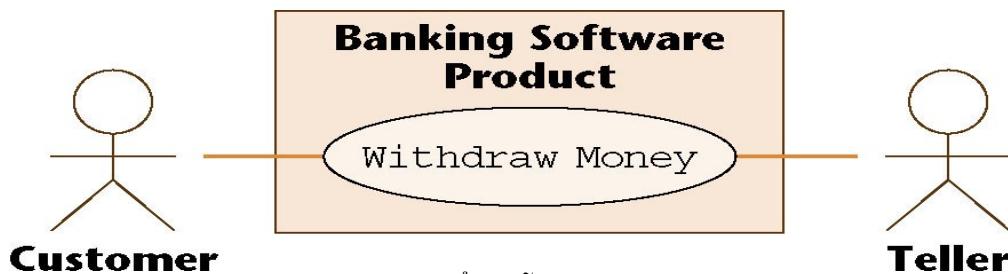
#### 6.2.2.2 Các kỹ thuật khác

- Phỏng vấn là kỹ thuật chính
- Một bản thăm dò ý kiến rất hữu ích khi lấy ý kiến của hàng trăm người.
- Kiểm tra các định dạng nghiệp vụ mà chỉ ra cách khách hàng thực hiện những công việc nghiệp vụ (Examination of business forms shows how the client currently does business )
- Quan sát trực tiếp những người công nhân thực hiện những nhiệm vụ của họ có thể là một cách rất hữu ích
  - Máy quay là một phiên bản hiện đại của kỹ thuật này
  - Nhưng, cần rất nhiều thời gian để phân tích các băng video
  - Những người công nhân có thể xem máy quay vì sự xâm phạm tùy tiện đời sống riêng tư

#### 6.2.3 Các use case

- Một use case mô hình tương tác giữa sản phẩm phần mềm với người dùng sản phẩm phần mềm đó (tác nhân - actors)

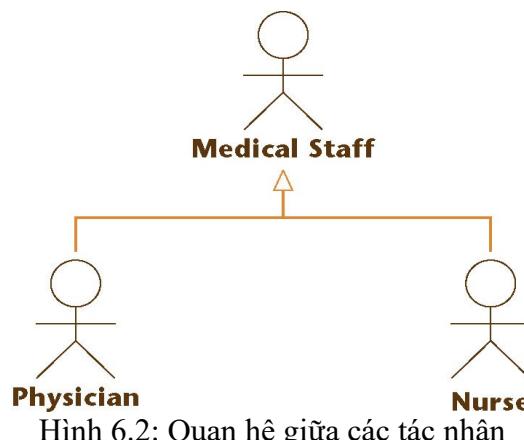
- Ví dụ:



Hình 6.1: Biểu diễn một use case

- Một tác nhân là một thành viên của thế giới bên ngoài sản phẩm phần mềm
- Thường rất dễ dàng nhận dạng ra tác nhân
  - Một tác nhân thường là một người dùng của hệ thống sản phẩm phần mềm
- Nhìn chung, một tác nhân đóng vai trò đối với hệ thống sản phẩm phần mềm. Vai trò này gồm:
  - Là một người dùng; hoặc
  - Là một khởi đầu; hoặc
  - Là một người nào đó đóng vai trò quan trọng trong use case
- Một người dùng của hệ thống có thể giữ nhiều hơn một vai trò
- Ví dụ: Một người khách hàng (**Customer**) của ngân hàng có thể là
  - Một người vay tiền hoặc
  - Một người cho mượn
- Ngược lại, một tác nhân có thể tham gia vào nhiều use case
- Ví dụ: một người vay tiền (**Borrower**) có thể là một tác nhân trong
  - Use case Borrow Money ;
  - Use case Pay Interest on Loan
  - Use case Repay Loan Principal
- Tác nhân người vay tiền (**Borrower**) có thể đại diện cho hàng nghìn khách hàng của ngân hàng
- Một tác nhân không cần thiết phải là một con người

- Ví dụ: hệ thống thông tin thương mại điện tử phải tương tác với hệ thống thông tin công ty thẻ tín dụng
  - Hệ thống thông tin công ty thẻ tín dụng là một tác nhân từ quan điểm của hệ thống thương mại điện tử
  - Hệ thống thương mại điện tử là một tác nhân của hệ thống thông tin công ty thẻ tín dụng
- Vấn đề dễ xảy ra khi xác định các tác nhân
  - Nạp chồng tác nhân
- Ví dụ: Hệ thống phần mềm bệnh viện
  - Một use case có tác nhân **Y tá** (Nurse)
  - Một use case khác có tác nhân **Nhân viên Y khoa** ( Medical Staff)
  - Tốt hơn:
    - Các tác nhân:: Bác Sỹ và Y tá (**Physician and Nurse**)
- Về mặt giải pháp:
  - Tác nhân **Nhân viên Y khoa** (Medical Staff) với hai sự chuyên môn hóa: Bác sỹ và Y tá (**Physician and Nurse**)



Hình 6.2: Quan hệ giữa các tác nhân

#### 6.2.4 Các yêu cầu ban đầu

- Những yêu cầu ban đầu dựa trên mô hình nghiệp vụ đầu tiên
- Sau đó chúng được làm mịn

- Các yêu cầu là động – có sự thay đổi thường xuyên
  - Duy trì một danh sách những yêu cầu quan trọng, cùng với các use case của các yêu cầu đã được phê chuẩn bởi khách hàng
- Có hai loại yêu cầu
- Yêu cầu chức năng chỉ rõ hành động mà sản phẩm phần mềm phải có khả năng thực hiện
  - Thường được biểu diễn về mặt các đầu vào và đầu ra
- Các yêu cầu phi chức năng chỉ rõ những đặc trưng của hệ thống sản phẩm phần mềm, như
  - Những ràng buộc về môi trường
  - Thời gian đáp ứng
  - Tính đáng tin
- Những yêu cầu chức năng được xử lý như là một phần của luồng công việc xác định yêu cầu và phân tích
- Một số yêu cầu phi chức năng phải phải chờ đến tận khi đến luồng công việc thiết kế
  - Thông tin chi tiết cho những yêu cầu phi chức năng không có sẵn cho đến tận khi luồng công việc phân tích và xác định yêu cầu hoàn thành

### 6.3 PHA XÁC ĐỊNH YÊU CẦU CỔ ĐIỂN

- Không phải thực hiện pha xác định yêu cầu cổ điển trong “xác định yêu cầu hướng đối tượng”
  - Luồng công việc xác định yêu cầu không phải làm gì khi sản phẩm phần mềm được xây dựng
- Tuy nhiên, phương pháp này được biểu diễn trong chương này là
  - Hướng mô hình và do đó
  - Hướng đối tượng
- Phương pháp cổ điển để xác định yêu cầu
  - Làm rõ các yêu cầu
  - Phân tích yêu cầu
  - Xây dựng bản mẫu nhanh

- Khách hàng và người dùng trong tương lai thử nghiệm với bản mẫu nhanh

#### 6.4 BẢN MẪU NHANH

- Xây dựng nhanh (“rapid”)
  - Sự hoàn thiện có thể được bỏ qua
- Chỉ đưa ra những chức năng chính
- Nhấn mạnh chỉ những gì mà khách hàng xem
  - Kiểm tra lỗi, cập nhật tệp có thể được bỏ qua
- Mục đích:
  - Để cung cấp tới khách hàng cách hiểu của sản phẩm phần mềm
- Bản mẫu nhanh được xây dựng để thay đổi
  - Ngôn ngữ cho bản mẫu nhanh bao gồm 4GLs và ngôn ngữ đã được thông dịch

#### 6.5 NHÂN TỐ CON NGƯỜI

- Khách hàng và những người dùng có ý định sử dụng hệ thống phải tương tác với giao diện người dùng
- Giao diện máy tính – con người (Human-computer interface HCI)
  - Bảng chọn, không dòng lệnh (Menu, not command line)
  - “Trỏ và nhấp”(“Point and click” )
  - Cửa sổ, biểu tượng, bảng chọn kéo xuống (Windows, icons, pull-down menus)
- Nhân tố con người phải được xem xét
  - Tránh bảng đơn dài dòng
  - Cho phép mức thay đổi mức độ thành thạo của giao diện
  - Tính đồng đều của hình thức là quan trọng ( Uniformity of appearance is important)
  - Tâm lý tiến bộ so với cảm giác chung (Advanced psychology vs. common sense?)
- Bản mẫu nhanh của giao diện người máy của mỗi sản phẩm phần mềm là bắt buộc

## 6.6 SỬ DỤNG LẠI BẢN MẪU NHANH

- Việc sử dụng lại bản mẫu nhanh là về bản chất là mô hình xây và sửa
- Những thay đổi được đưa ra để xây dựng phần mềm
  - Đắt (Expensive)
- Bảo trì khó vì không có tài liệu đặc tả và tài liệu thiết kế
- Những ràng buộc về thời gian thực khó đáp ứng
- Một cách để đảm bảo rằng bảng mẫu nhanh được bỏ qua
  - Cài đặt bản mẫu nhanh bằng một ngôn ngữ khác so với ngôn ngữ đã lựa chọn cho sản phẩm đích(Implement it in a different language from that of the target product)
- Mã được sinh ra có thể được sử dụng lại
- Chúng ta có thể giữ lại một cách an toàn (các phần của bản mẫu nhanh ) bản mẫu nhanh nếu
  - Điều này được chuẩn bị trước
  - Những phần của bản mẫu nhanh đã qua kiểm tra kỹ lưỡng của nhóm SQA
  - Tuy nhiên, đây không phải là một bản mẫu nhanh cổ điển ( “classical” rapid prototyping)

## 6.7 CÁC CÔNG CỤ CASE CHO XÁC ĐỊNH YÊU CẦU

- Chúng ta cần các công cụ đồ họa cho các biểu đồ UML
  - Dễ dàng để thay đổi các biểu đồ UML
  - Tài liệu được lưu trong các công cụ và luôn có sẵn
- Các công cụ như vậy đôi khi rất khó sử dụng
- The diagrams may need considerable “tweaking”
- Nhìn chung, điểm mạnh có nhiều ảnh hưởng tốt hơn điểm yếu (Overall, the strengths outweigh the weaknesses)
- Các môi trường CASE đồ họa được mở rộng để trợ giúp UML
  - System Architect

- Software through Pictures
- Các môi trường CASE hướng đối tượng bao gồm
  - IBM Rational Rose
  - Together
  - ArgoUML (open source)

## 6.8 CÁC THƯỚC ĐO CHO XÁC ĐỊNH YÊU CẦU

- Volatility and speed of convergence are measures of how rapidly the client's needs are determined
- Số lượng thay đổi được đưa ra trong suốt chuỗi con các pha
- Những thay đổi được đề xướng bởi những người phát triển
  - Quá nhiều thay đổi có thể đồng nghĩa với việc quy trình không hoàn thiện
  - Những thay đổi được đề xướng bởi khách hàng
  - Thay đổi sản phẩm phần mềm cuối cùng

## 6.9 NHỮNG THỦ THÁCH CHO PHA XÁC ĐỊNH YÊU CẦU

- Nhân viên của tổ chức khách hàng thường cảm giác bị đe dọa bởi máy tính
- Những thành viên đội xác định yêu cầu phải có khả năng thương lượng
  - Yêu cầu của khách hàng có thể phải thu hẹp phạm vi
- Nhân viên chính của tổ chức khách hàng không thể có thời gian cho những cuộc thảo luận cốt yếu và sâu sắc
- Linh hoạt và khách quan là cốt yếu

## 6.10 CASE STUDY CHO PHA XÁC ĐỊNH YÊU CẦU

### 6.10.1 Bài toán

Khách hàng đến đặt hàng chúng ta xây dựng một phần mềm quản lý khách sạn. Do đó, trước khi đến với phần mô tả chi tiết hệ thống, chúng ta cần tìm hiểu về lĩnh vực quản lý khách sạn.

*Danh sách các từ khóa liên quan đến lĩnh vực đặt phòng khách sạn*

khách sạn	đặt phòng	nhân viên	Quản lý thông tin khách sạn
phòng	đặt phòng tại quầy	nhân viên lễ tân	sửa thông tin khách sạn
phòng đơn	đặt qua điện thoại	nhân viên bán hàng	quản lý thông tin phòng
phòng đôi	nhận phòng	nhân viên quản lý	thêm thông tin phòng
phòng kép (twink)	trả phòng	nhân viên quản trị	sửa thông tin phòng
phòng hướng vườn	hủy đặt phòng	nhân viên dọn phòng	xóa thông tin phòng
phòng hướng biển	hủy đặt phòng tại quầy	khách hàng	quản lý thông tin nhân viên
phòng hướng sân	hủy qua điện thoại	khách lẻ	thêm thông tin nhân viên
giường đơn	hóa đơn	khách đoàn	sửa thông tin nhân viên
giường đôi	hóa đơn thanh toán	khách vãng lai	đăng nhập
dịch vụ	hóa đơn đặt cọc	khách đặt trước	đăng xuất
dịch vụ ăn sáng	đặt cọc	đối tác	đổi mật khẩu
dịch vụ giặt là	khuyến mãi		tìm phòng trống
dịch vụ đưa đón	khuyến mãi theo dịch vụ		tìm thông tin khách hàng
dịch vụ chăm sóc sức khỏe	khuyến mãi theo hóa đơn		tìm thông tin đặt phòng
mini bar	phí hủy đặt phòng		xem thống kê phòng
đồ ăn	phí phạt hỏng/mất đồ		thống kê phòng theo doanh thu
nước uống	thanh toán		thống kê phòng theo tỉ
tỉ lệ lắp đầy phòng			
tỉ lệ lắp đầy khách sạn			

tỷ lệ lấp đầy

thống kê doanh thu  
theo thời gian

thống kê dịch vụ theo  
doanh thu

**Giải thích nghĩa của các thuật ngữ trong ngữ cảnh của ứng dụng đặt phòng khách sạn:**

TT	Tên Tiếng Việt	Tên Tiếng Anh	Ngữ nghĩa
<b>Nhóm các thuật ngữ liên quan đến khách sạn</b>			
1	Khách sạn	Hotel	
2	Phòng	Room	
3	Phòng đơn	Single	
4	Phòng đôi	Double	
5	Phòng kép	Twink	
6	Hướng vườn	Garden view	
7	Hướng biển	Sea view	
8	Hướng sân		
9	Giường đơn		
10	Giường đôi		
11	Dịch vụ		
12	Dịch vụ ăn sáng		
13	Dịch vụ giặt là		
14	Dịch vụ đưa đón		
15	Dịch vụ chăm sóc sức khỏe		
16	Quầy bar tại phòng		
17	đồ ăn		
18	đồ uống		
19	tỉ lệ lấp đầy phòng		
20	tỉ lệ lấp đầy khách sạn		
<b>Nhóm thuật ngữ liên quan hoạt động của khách hàng</b>			
21	Đặt phòng		

22	Đặt phòng tại quầy		
23	Đặt qua điện thoại		
24	Nhận phòng		
25	Trả phòng		
26	Hủy phòng		
27	Hủy tại quầy		
28	Hủy qua điện thoại		
29	Hóa đơn		
30	Hóa đơn thanh toán		
31	Đặt cọc		
32	Hóa đơn đặt cọc		
33	Khuyến mãi		
34	Khuyến mãi theo dịch vụ		
35	Khuyến mãi theo hóa đơn		
36	Phí hủy đặt phòng		
37	Hóa đơn phạt		
38	Phí phạt hỏng/mất đồ		
39	Thanh toán		
<b>Nhóm thuật ngữ liên quan đến người dùng</b>			
40	Nhân viên		
41	Nhân viên lễ tân		
42	Nhân viên quản lý		
43	Nhân viên bán hàng		
44	Nhân viên quản trị		
45	Nhân viên don phòng		
46	Khách hàng		
47	Khách lẻ		
48	Khách đoàn		
49	Khách vãng lai		

50	Khách đặt trước		
51	Đối tác		
<b>Các thuật ngữ liên quan đến hoạt động của nhân viên</b>			
52	Quản lý thông tin khách sạn		
53	Sửa thông tin khách sạn		
54	Quản lý thông tin phòng		
55	Thêm thông tin phòng		
56	Sửa thông tin phòng		
57	Xóa thông tin phòng		
58	Quản lý thông tin nhân viên		
59	Thêm thông tin nhân viên		
60	Sửa thông tin nhân viên		
61	Xóa thông tin nhân viên		
62	đăng nhập		
63	đăng xuất		
64	Tìm phòng trống		
65	Tìm khách hàng		
66	Tìm thông tin đặt phòng		
67	Xem thống kê phòng		
68	Thống kê phòng theo doanh thu		
69	Thống kê phòng theo tỉ lệ lấp đầy		
70	Thống kê doanh		

	thu theo thời gian		
71	Thống kê dịch vụ theo doanh thu		

### 6.10.2 Mô tả hệ thống (bằng ngôn ngữ tự nhiên)

Đây có thể coi như là một bản mô tả yêu cầu của khách hàng bằng ngôn ngữ tự nhiên. Phần mô tả chung như sau:

#### **Phạm vi phần mềm:**

- Phần mềm dạng ứng dụng cho máy tính cá nhân, dùng nội bộ trong một khách sạn, chỉ có nhân viên khách sạn được sử dụng trực tiếp. Phần mềm có thể cài đặt trên nhiều máy tính cá nhân của nhân viên. Nhưng cơ sở dữ liệu là tập trung tại một máy chủ duy nhất.

#### **Người dùng và chức năng của người dùng:**

- Chỉ các nhân viên sau của khách sạn được phép dùng phần mềm: nhân viên lễ tân, nhân viên bán hàng, quản trị hệ thống, và nhân viên quản lý khách sạn được sử dụng.
- Nhân viên lễ tân có thể thực hiện các chức năng:
  - Đặt phòng theo yêu cầu trực tiếp của khách tại quầy,
  - Hủy đặt phòng theo yêu cầu của khách tại quầy,
  - Checkin cho khách đã đặt phòng hoặc đặt phòng trực tiếp,
  - Checkout cho khách và in hóa đơn thanh toán cho khách.
- Nhân viên bán hàng có thể thực hiện các chức năng:
  - Đặt phòng theo yêu cầu của khách qua điện thoại,
  - Hủy đặt phòng theo yêu cầu của khách qua điện thoại.
- Nhân viên quản lý có thể thực hiện các chức năng:
  - Quản lý thông tin khách sạn: thêm/sửa,
  - Quản lý thông tin phòng: thêm/sửa/xóa,
  - Xem các loại báo cáo thống kê: xem các báo cáo doanh thu theo thời gian/theo phòng/theo loại phòng, xem báo cáo tỉ lệ phòng trống theo thời gian/theo

phòng/theo loại phòng, xem báo cáo khách hàng đặt nhiều theo thời gian/theo nguồn gốc khách hàng.

- Nhân viên quản trị hệ thống có thể thực hiện:
  - Quản lý tài khoản người dùng theo yêu cầu của nhân viên: thêm/sửa/xóa.

#### **Thông tin các đối tượng cần xử lý:**

- Thông tin về khách sạn bao gồm : tên, địa chỉ, số sao, mô tả (bao gồm mô tả bằng văn bản và bằng hình ảnh).
- Thông tin về phòng khách sạn: tên, kiểu (single/double/twink), giá, mô tả.
- Thông tin về khách hàng: tên, địa chỉ, số CCCD/passport, điện thoại, email, ghi chú về phục vụ đặc biệt như cho người khuyết tật, ăn chay...
- Thông tin các dịch vụ đi kèm: tên, đơn vị tính, đơn giá, mô tả.
- Thông tin nhân viên trong khách sạn: tên, tên đăng nhập, mật khẩu, chức vụ, ghi chú.
- Thông tin hóa đơn thanh toán: bao gồm thông tin nhân viên thanh toán: tên, chức vụ. Thông tin khách hàng: tên, địa chỉ idcard, số điện thoại. thông tin các phòng đặt/ở, mỗi phòng trên một dòng với các thông tin: tên, kiểu, giá, ngày đến, ngày đi, thành tiền. Mỗi phòng có danh sách các dịch vụ đi kèm đã sử dụng, mỗi dịch vụ có các thông tin: tên, đơn vị tính, đơn giá, số lần dùng, thành tiền. Dòng cuối cùng là tổng tiền hóa đơn, tổng tiền đã thanh toán trước, tổng tiền phải thanh toán còn lại.
- Thông tin thống kê phòng theo doanh thu: tên phòng, kiểu phòng, mô tả, tổng số ngày có khách, tổng doanh thu.
- Thông tin thống kê khách hàng theo doanh thu: tên, địa chỉ, điện thoại, tổng số ngày đã ở, tổng tiền thu được.
- Thông tin thống kê các dịch vụ theo doanh thu: tên, tổng đơn vị tiêu thụ, tổng doanh thu.
- Thông tin thống kê doanh thu theo thời gian: tên tháng/năm/quý/tuần, tổng lượng khách, tổng doanh thu.

#### **Quan hệ giữa các đối tượng cần xử lý:**

- Trong khách sạn có nhiều phòng, mỗi phòng chỉ thuộc vào một khách sạn.
- Mỗi phòng có thể được đặt/ở bởi nhiều khách hàng khác nhau tại những thời điểm khác nhau.

- Mỗi khách hàng có thể đặt/ở nhiều phòng khác nhau tại những thời điểm khác nhau.
- Đối với khách lẻ, một khách hàng, tại một thời điểm, chỉ ở một phòng xác định, xác định ngày đến, ngày đi, đơn giá nhất định.
- Đối với khách đoàn, mỗi thời điểm, một khách có thể đứng tên và ở trên danh nghĩa nhiều phòng cùng lúc (đại diện cho cả đoàn).
- Khách hàng chỉ có thể đặt phòng nếu phòng đó còn trống trong suốt thời gian khách hàng muốn đặt.
- Khách hàng có thể thanh toán nhiều lần cho đến ngày trả phòng.
- Khách hàng có thể hủy đặt phòng (miễn phí) nếu hủy trước ngày đến. Nếu khách hàng hủy sau ngày đặt thì khách hàng bị lưu vào danh sách đen và có thể bị từ chối đặt phòng trong các lần tiếp theo.
- Một nhân viên bán hàng/lễ tân có thể đặt phòng nhiều lần cho khách.
- Một nhân viên lễ tân có thể thanh toán nhiều lần cho khách.

#### *Mô tả nghiệp vụ chi tiết của các chức năng:*

Nội dung phần này chỉ mô tả một số chức năng mà không mô tả hết toàn bộ chức năng của hệ thống:

- *Chức năng sửa thông tin phòng:* Quản lý đăng nhập vào hệ thống -> giao diện quản lý chung hiện ra có chức năng quản lý thông tin khách sạn, quản lý thông tin phòng, xem báo cáo thống kê -> chọn chức năng quản lý thông tin phòng -> giao diện hiện ra có 3 lựa chọn là thêm, sửa, xóa thông tin phòng -> click chọn sửa thông tin phòng -> giao diện tìm phòng theo tên hiện ra với 1 ô nhập tên phòng và nút tìm kiếm -> gõ từ khóa vào và click tìm -> danh sách các phòng có tên chứa từ khóa vừa nhập hiện ra, mỗi phòng đầy đủ thông tin: id, tên, kiểu, giá, mô tả -> click 1 phòng muốn sửa -> giao diện sửa hiện ra với các ô điền sẵn thuộc tính hiện tại của phòng (trừ id không sửa được) -> sửa lại một vài thuộc tính và click save -> hệ thống báo thành công và quay về trang chủ người quản lý.
- *Chức năng đặt phòng (qua điện thoại):* Khách hàng gọi điện đến khách sạn yêu cầu đặt phòng -> lễ tân sẽ chuyển máy cho nhân viên bán hàng để tư vấn đặt phòng cho khách -> nhân viên bán hàng bắt máy. Nhân viên hỏi khách hàng muốn ở từ ngày nào đến ngày nào -> chọn chức năng đặt phòng sau khi đã đăng nhập -> giao diện tìm phòng trống hiện ra, có 2 ô nhập ngày bắt đầu, ngày kết thúc -> nhân viên nhập theo nguyện vọng của khách hàng và click tìm -> danh sách các phòng trống trong khoảng thời gian đã chọn hiện lên dưới dạng bảng, mỗi dòng có id, tên, kiểu, giá, mô tả -> nhân viên thông báo các loại phòng còn trống cho khách hàng và yêu cầu khách hàng chọn phòng mong muốn -> khách hàng nói nguyện vọng và nhân viên click 1 phòng theo đúng nguyện vọng của

khách (đúng giá, loại phòng) -> giao diện thông tin khách hàng hiện lên, có ô nhập để tìm -> nhân viên hỏi xin khách hàng tên, địa chỉ, số CCCD/pasport, số điện thoại, sau đó nhập tên vào để tìm kiếm -> giao diện các khách hàng đã có tên chúa từ khóa vừa nhập hiện lên dạng bảng, mỗi dòng đầy đủ thông tin: id, tên, địa chỉ, số CCCD/passport, điện thoại, ghi chú -> nhân viên click vào dòng đúng với khách hàng vừa báo (nếu không có khách hàng này thì nhân viên click chức năng thêm khách hàng để thêm mới) -> hệ thống chuyển sang phần xác nhận thông tin đặt phòng: thông tin khách hàng, thông tin phòng, thông tin ngày đến, ngày đi -> nhân viên xác nhận lại với khách hàng và yêu cầu xác nhận -> khách hàng xác nhận, nhân viên click lưu đặt phòng -> hệ thống báo thành công và quay về trang chủ nhân viên bán hàng -> nhân viên báo đặt phòng thành công với khách hàng.

- **Chức năng nhận phòng:** khách hàng đến quầy lễ tân khách sạn yêu cầu nhận phòng đã đặt -> nhân viên lễ tân xin CCCD/passport của khách hàng để kiểm tra thông tin -> chọn chức năng giao phòng cho khách sau khi đăng nhập -> giao diện tìm thông tin đặt phòng hiện lên, có các ô nhập tên khách, số CCCD/passport, ngày vào, ngày ra -> lễ tân nhập tên khách và click tìm-> giao diện danh sách các khách hàng có tên chúa từ khóa vừa nhập và đã đặt phòng nhưng chưa nhận hiện lên, mỗi dòng của bảng có đầy đủ thông tin: thông tin khách, thông tin phòng, ngày vào, ngày ra -> lễ tân hỏi lại thông tin đặt phòng của khách hàng cho đúng và click vào dòng đúng thông tin phòng đặt -> hệ thống cập nhật lại thông tin khách đã nhận phòng và quay về trang chủ của lễ tân -> lễ tân lấy chìa khóa phòng và giao cho khách. Trong khi nhận phòng, khách có thể đặt tiền trước một phần hoặc toàn bộ theo nguyện vọng của mình. Khi đó nhân viên bổ sung thông tin trả trước và xuất hóa đơn trả trước cho khách hàng chúa thông tin khách hàng, thông tin phòng, thông tin đặt phòng, số tiền đã thanh toán.
- **Chức năng trả phòng (và thanh toán):** Khách hàng cầm chìa khóa phòng ra quầy lễ tân yêu cầu trả phòng và thanh toán -> lễ tân nhận chìa khóa và mời khách ngồi chờ để kiểm tra thông tin và in hóa đơn -> lễ tân gọi nhân viên dọn phòng yêu cầu kiểm tra phòng do khách muốn trả, đồng thời click vào chức năng trả phòng sau khi đăng nhập -> giao diện hiện lên có thông tin phòng đặt, thông tin ngày đến, ngày đi, thông tin các lần khách tạm thanh toán, nếu có, danh sách các dịch vụ khách đã sử dụng trong thời gian ở khách sạn, nếu có, chức năng bổ sung dịch vụ sử dụng, tổng tiền còn phải thanh toán -> nếu nhân viên dọn phòng báo lại khách có làm hỏng đồ hoặc sử dụng đồ ăn, nước uống trong mini bar thì nhân viên lễ tân click chức năng bổ sung dịch vụ để thêm lần lượt vào -> sau đó lễ tân sẽ báo lại thông tin cho khách hàng kiểm tra lần cuối trước khi xuất hóa đơn: thông tin khách, thông tin phòng, ngày đến, ngày đi, tổng tiền phòng, các dịch vụ đã sử dụng, tổng tiền dịch vụ, tổng tiền đã thanh toán trước, tổng tiền còn lại phải thanh toán -> khách hàng xác nhận lại và lễ tân click lưu hóa đơn -> hệ thống lưu hóa đơn, đồng thời in hóa đơn ra cho khách hàng -> lễ tân nhận tiền thanh toán và giao hóa đơn cho khách hàng.
- **Chức năng xem thống kê phòng theo doanh thu:** Quản lý đăng nhập vào hệ thống -> giao diện quản lý chung hiện ra có chức năng quản lý thông tin khách sạn, quản lý thông tin

phòng, xem báo cáo thống kê -> chọn chức năng xem báo cáo thống kê -> giao diện chọn loại thống kê hiện ra, có các đối tượng thống kê gồm thống kê khách sạn, thống kê phòng, thống kê kiểu phòng, thống kê dịch vụ, thống kê khách hàng, thống kê doanh thu. Các hình thức thống kê gồm theo doanh thu, theo thời gian (nếu chọn đối tượng thống kê là doanh thu), theo tỉ lệ lấp đầy (nếu chọn đối tượng thống kê là phòng, khách sạn) -> chọn thống kê phòng theo doanh thu -> giao diện hiện lên yêu cầu nhập thời điểm bắt đầu và thời điểm kết thúc thống kê -> quản lý nhập vào và click thống kê -> kết quả hiện lên dạng bảng, mỗi dòng tương ứng với một phòng: id, tên, kiểu, tổng số ngày có khách, tổng tiền thu được, sắp xếp theo chiều giảm dần tổng số tiền thu được -> quản lý click vào 1 phòng để xem chi tiết -> giao diện hiện lên thông tin phòng (id, tên, kiểu) và danh sách các lần phòng đã được đặt trong khoảng thời gian thống kê, mỗi dòng có các thông tin: tên khách hàng, ngày đến, ngày đi, đơn giá, tổng tiền. Sắp xếp theo chiều thời gian ngày thanh toán. -> quản lý xem xong click quay lại để về trang chủ của quản lý.

### 6.10.3 Xây dựng sơ đồ use case tổng quan

Mục đích của bước này là xây dựng một bản mô tả yêu cầu của khách hàng bằng ngôn ngữ kỹ thuật (UML).

Xác định các actor có thể có của hệ thống:

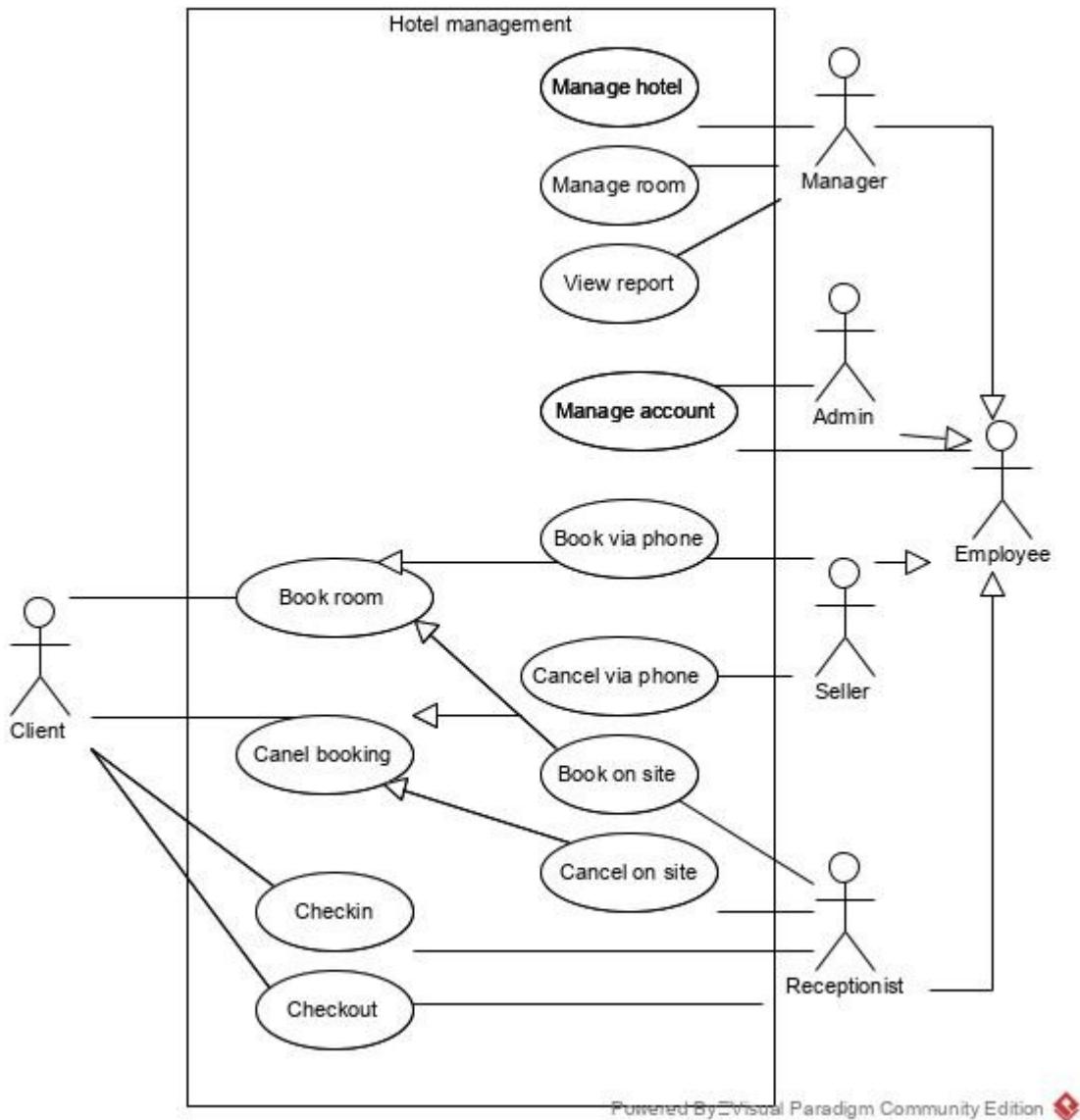
- Actor là người dùng trực tiếp: người quản lý khách sạn (manager), nhân viên quản trị hệ thống (admin), nhân viên bán hàng (saller), nhân viên lễ tân kiêm luôn thủ quỹ để nhận thanh toán (receptionist). Các actor này đều cùng một dạng là nhân viên khách sạn (Employee), nên có quan hệ kế thừa từ actor này.
- Actor là người dùng gián tiếp: Khách hàng (client), mặc dù không trực tiếp sử dụng và thao tác trên phần mềm, nhưng một số chức năng phải có mặt khách hàng mới thực hiện được như: đặt chỗ, checkin, checkout, thanh toán.

Các chức năng liên quan đến các actor:

- Người quản lý khách sạn (Manager): quản lý thông tin phòng (room manage) và khách sạn (hotel manage), tạo và xem các loại báo cáo (view report)
- Người quản trị hệ thống (Admin): quản lý thông tin người dùng (account manage)
- Nhân viên bán hàng (Saller): giao dịch với khách hàng (Client) qua điện thoại để đặt chỗ (Book a room) hoặc hủy đặt chỗ (Cancel a booking)
- Nhân viên tiếp tân (Receptionist): giao dịch trực tiếp với khách hàng (Client) tại quầy để đặt chỗ (Book on site), hủy đặt chỗ (Cancel on site), nhận Checkin, Checkout và thanh toán cho khách hàng.

- Khách hàng (Client): có thể đặt phòng/hủy phòng (Book via phone/Cancel via phone) trực tiếp tại quầy với nhân viên lễ tân hoặc đặt/hủy qua điện thoại với nhân viên bán hàng. Checkin, Checkout và thanh toán tại quầy với nhân viên lễ tân.
- Đặt phòng tại quầy với lễ tân (Book on site) cũng giống đặt phòng qua điện thoại với nhân viên bán hàng (Book via phone), do đó hai UC này có thể kế thừa từ UC đặt phòng (Book a room).
- Tương tự, hủy đặt phòng tại quầy với lễ tân (Cancel on site) cũng giống hủy đặt phòng qua điện thoại với nhân viên bán hàng (Cancel via phone), nên hai UC này có thể kế thừa từ UC hủy đặt phòng (Cancel a booking)

Như vậy nhóm dự án thu được sơ đồ use case tổng quan như sau:



Hình 6.3: Sơ đồ use case tổng quan của hệ thống

Các use case được mô tả như sau:

- Manage hotel: UC này cho phép nhân viên quản lý cập nhật thông tin về khách sạn.
- Manage room: UC này cho phép nhân viên quản lý có thể thêm, sửa, xóa thông tin phòng.
- View report: UC này cho phép nhân viên quản lý vào xem các loại báo cáo thống kê về tình hình hoạt động kinh doanh của khách sạn.
- Manage account: UC này cho phép nhân viên quản trị hệ thống có thể thêm, sửa, xóa tài khoản người sử dụng khi có yêu cầu từ nhân viên tương ứng.

- Book a room: UC này cho phép nhân viên lễ tân hoặc nhân viên bán hàng đặt phòng cho khách hàng theo yêu cầu.
- Book on site: UC này cho phép nhân viên lễ tân đặt phòng cho khách ngay tại quầy.
- Book via phone: UC này cho phép nhân viên bán hàng đặt phòng cho khách qua điện thoại.
- Cancel a booking: UC này cho phép nhân viên lễ tân hoặc nhân viên bán hàng hủy đặt phòng cho khách hàng theo yêu cầu.
- Cancel on site: UC này cho phép nhân viên lễ tân hủy đặt phòng cho khách tại quầy.
- Cancel via phone: UC này cho phép nhân viên bán hàng hủy đặt phòng cho khách qua điện thoại.
- Checkin: UC này cho phép nhân viên lễ tân cho khách nhận phòng tại quầy.
- Checkout: UC này cho phép nhân viên lễ tân nhận trả phòng và thanh toán tiền phòng của khách hàng.

#### 6.10.4 Phân rã chi tiết các use case

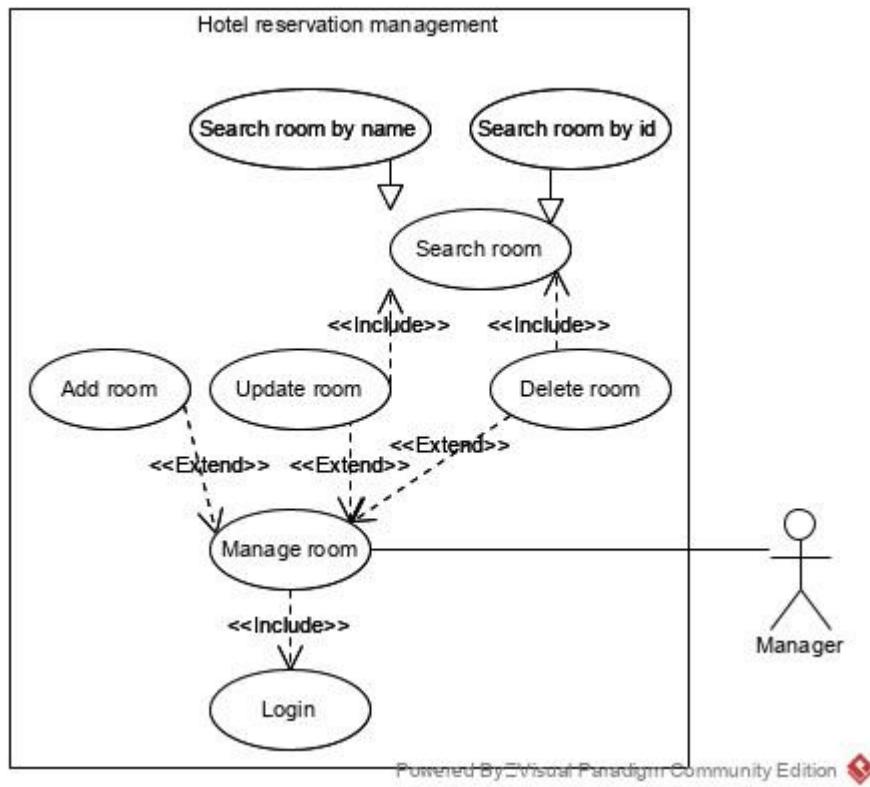
Mục đích của bước này là mô tả chi tiết các use case đã xác định được trong sơ đồ tổng quan.

##### a. Use case quản lý phòng

Quản lý thông tin phòng có thể thêm, sửa, xóa phòng. Sửa và xóa phòng thì phải tìm phòng để sửa/xóa. Tìm phòng ở đây có thể tìm phòng theo tên hoặc theo ID.

Mô tả các use case:

- Add room: UC này cho phép người quản lý thêm thông tin một phòng mới vào hệ thống
- Edit room: UC này cho phép người quản lý chỉnh sửa thông tin một phòng (giá, tên, mô tả)
- Delete room: UC này cho phép người quản lý xóa thông tin một phòng.
- Search room: UC này cho phép tìm thông tin phòng để phục vụ UC sửa, hoặc xóa phòng. Việc tìm kiếm có thể thực hiện tìm theo tên hoặc id của phòng.



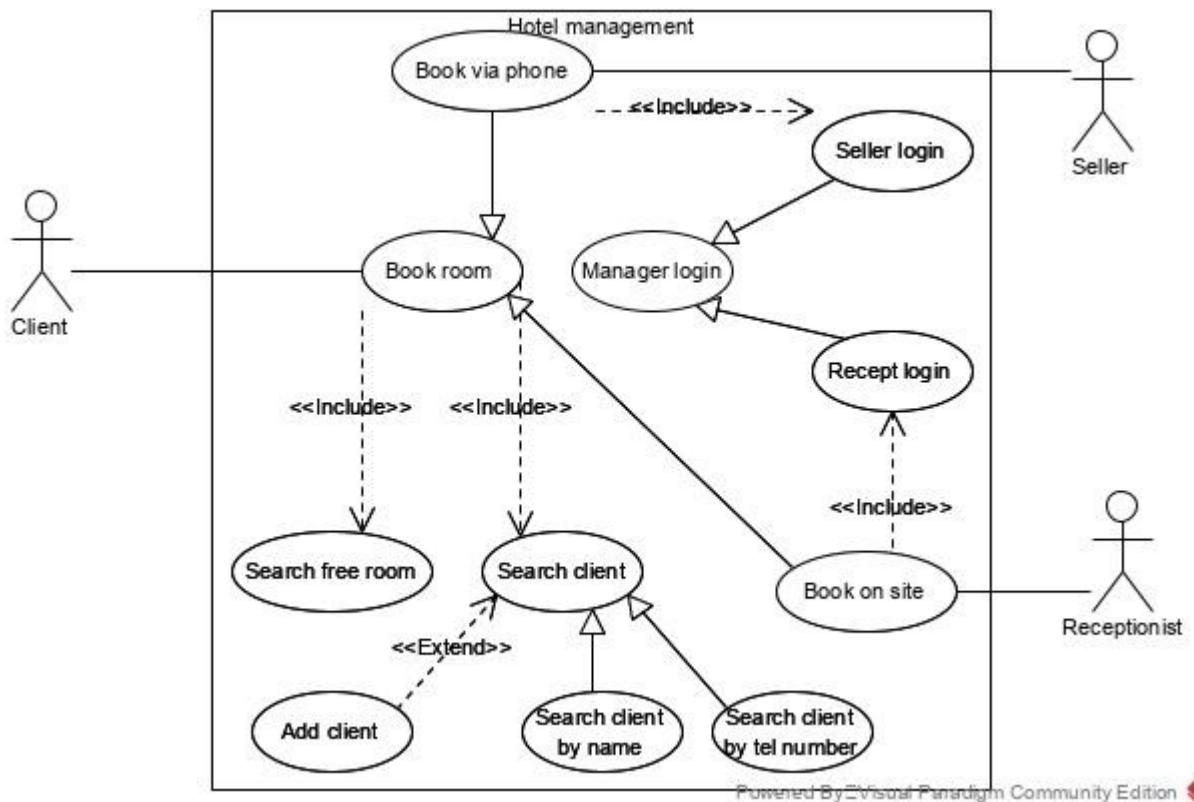
Hình 6.4: Chi tiết use case quản lý phòng

### b. Use case đặt phòng

Muốn thực hiện đặt phòng, nhân viên lễ tân hoặc nhân viên bán hàng phải đăng nhập, tìm phòng trống, tìm khách hàng xem đã có trong hệ thống chưa, nếu chưa có thì phải thêm khách hàng mới.

Mô tả các use case:

- Search free room: UC này cho phép nhân viên tìm phòng còn trống để phục vụ việc đặt phòng.
- Search client: UC này cho phép nhân viên tìm thông tin khách hàng trong hệ thống để phục vụ việc đặt phòng. Việc tìm kiếm có thể thực hiện tìm kiếm theo tên hoặc số điện thoại.
- Add client: UC này cho phép nhân viên thêm thông tin khách hàng vào hệ thống khi đặt phòng cho khách, mà thông tin của khách hàng chưa có trong hệ thống.



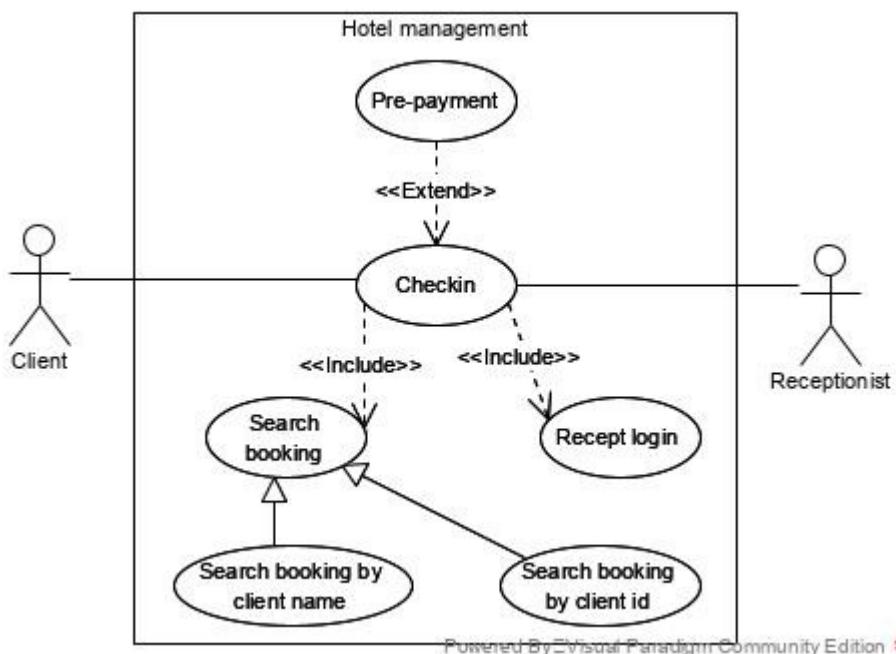
Hình 6.5: Chi tiết use case đặt phòng

### c. Use case nhận phòng

Để cho phép khách nhận phòng, sau khi đăng nhập, nhân viên lễ tân phải tìm thông tin đặt phòng để cập nhật trạng thông tin khách đã nhận phòng. Việc tìm thông tin đặt phòng có thể thực hiện theo tên khách hàng, số thẻ căn cước công dân/hộ chiếu của khách hàng. Trong khi nhận phòng, khách hàng có thể thanh toán trước nếu muốn, khi đó nhân viên thực hiện thanh toán và in hóa đơn cho khách hàng.

Mô tả các use case:

- Search booking: UC này cho phép nhân viên lễ tân tìm thông tin đặt phòng để thực hiện chức năng cho khách nhận phòng. Việc tìm thông tin đặt phòng có thể thực hiện theo tên khách hàng, số thẻ căn cước công dân/hộ chiếu của khách hàng.
- Pre-payment: UC này cho phép nhân viên lễ tân thực hiện nhận thanh toán trước (toute bộ hoặc một phần) từ khách hàng ngay khi khách đến nhận phòng.



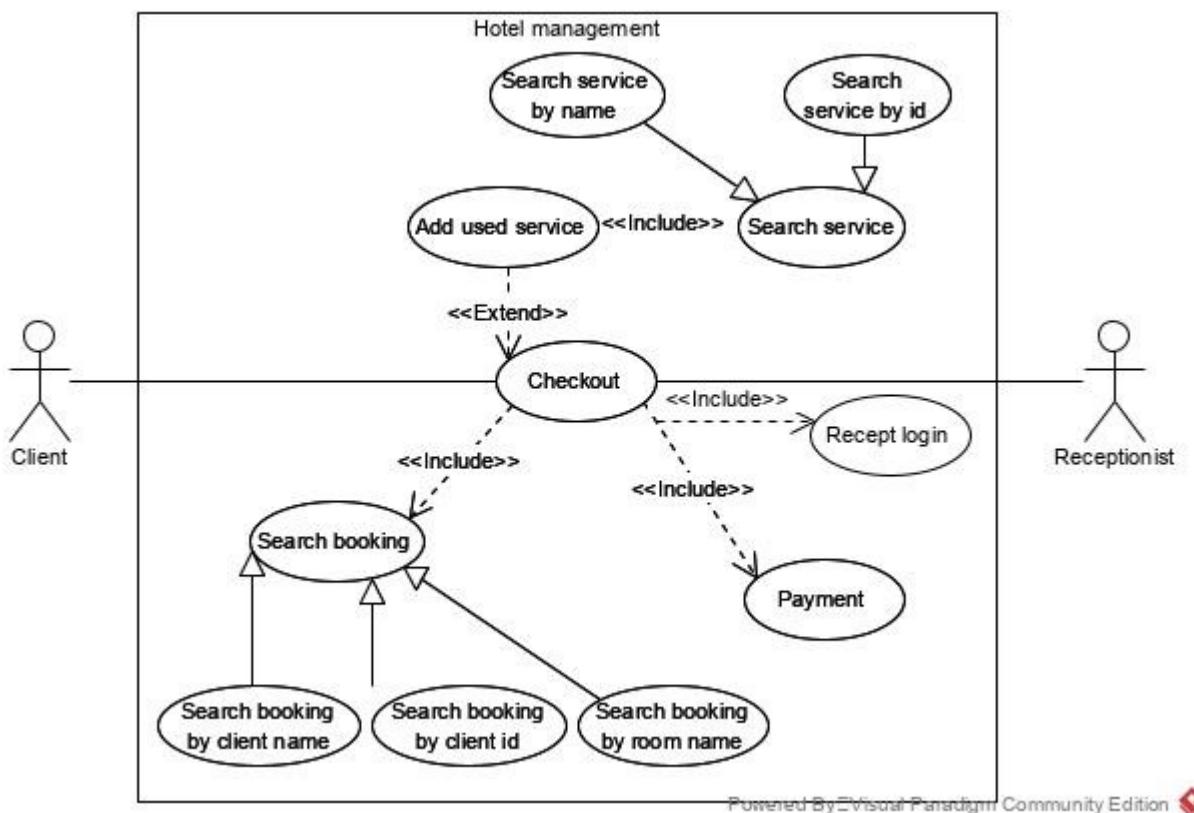
Hình 6.6: Chi tiết use case nhận phòng

#### d. Use case trả phòng

Khi có yêu cầu trả phòng từ khách hàng, nhân viên lễ tân sau khi đăng nhập phải tìm thông tin đặt phòng. Việc này có thể tìm theo tên khách hàng, số thẻ căn cước/hộ chiếu của khách hàng, hoặc theo tên phòng. Lễ tân có thể phải thực hiện bổ sung thông tin các dịch vụ khách đã dùng trong thời gian nghỉ tại khách sạn, nếu có. Với mỗi dịch vụ thì phải tiến hành tìm kiếm để chọn, nhập số lượng để lên hóa đơn. Cuối cùng là lễ tân thực hiện thanh toán cho khách hàng.

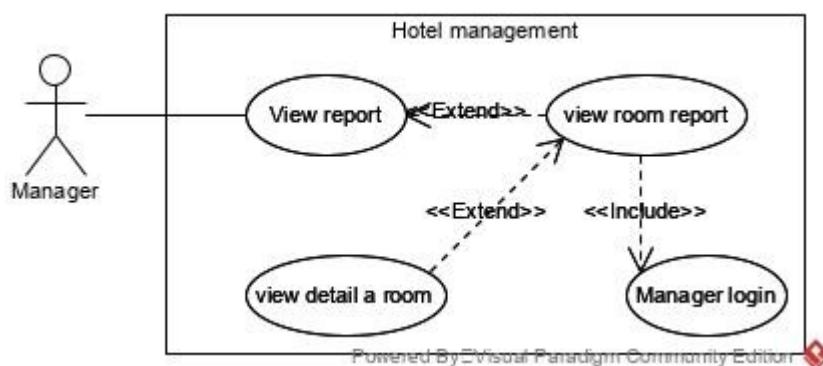
Mô tả các use case:

- Search booing: UC này cho phép nhân viên lễ tân tìm thông tin đặt phòng để thực hiện chức năng cho khách trả phòng. Việc tìm thông tin đặt phòng có thể thực hiện theo tên khách hàng, số thẻ căn cước công dân/hộ chiếu của khách hàng, hoặc tên phòng do khách trả chìa khóa.
- Add used service: UC này cho phép lễ tân thêm dịch vụ do khách hàng sử dụng trong thời gian nghỉ tại khách sạn
- Search service: UC này cho phép lễ tân tìm thông tin dịch vụ để thêm vào hóa đơn thanh toán cho khách. Việc tìm kiếm có thể tiến hành dựa trên tên dịch vụ hoặc mã dịch vụ.
- Payment: UC này cho phép lễ tân tạo hóa đơn, xuất hóa đơn và nhận thanh toán từ khách hàng.



Hình 6.7: Use case trả phòng

#### e. Use case Xem thống kê phòng theo doanh thu



Hình 6.8: Use case xem thống kê

- view room report: UC này cho phép nhân viên quản lý xem thống kê phòng theo doanh thu.
- view detail a room: UC này cho phép nhân viên quản lý xem chi tiết các hóa đơn của 1 phòng.

## CHƯƠNG 7: PHƯƠNG PHÁP PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

### 7.1 LUỒNG CÔNG VIỆC PHÂN TÍCH

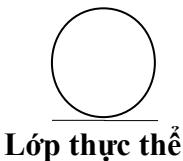
Mục đích: có 2 mục đích chính

- Để hiểu sâu hơn về các yêu cầu
- Mô tả các yêu cầu theo một cách thức nhất định để tạo điều kiện thuận lợi cho việc thiết kế và cài đặt sau đó có khả năng bảo trì được.

Ba kiểu lớp chính:

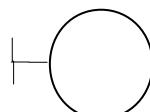
- Các lớp thực thể: mô hình thông tin lưu trữ lâu dài, chẳng hạn như: lớp account và lớp investment.

Ký hiệu UML của lớp thực thể:



- Các lớp biên: mô hình những tương tác giữa hệ thống phần mềm với môi trường. Lớp biên nhìn chung gắn liền với đầu vào hoặc đầu ra hoặc giao tiếp với các tác nhân. Ví dụ: lớp Investments Report và lớp Mortgages Report.

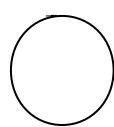
Ký hiệu UML của lớp biên:



Lớp biên

- Các lớp điều khiển: mô hình những tính toán và những thuật toán phức tạp. Ví dụ: lớp Estimate Funds for Week.

Ký hiệu UML của lớp điều khiển:



Lớp điều khiển

### 7.2 VIỆC TRÍCH RÚT CÁC LỚP THỰC THỂ

Thực hiện theo ba bước sau một cách lặp và tăng dần:

- Việc mô hình hóa chức năng (hay còn gọi là mô hình hóa Use-Case): Xác định các kết quả khác nhau được đưa ra bởi hệ thống phần mềm. Biểu diễn các thông tin đó dưới dạng các kịch bản của tất cả các Use-Case (mỗi kịch bản là một thể hiện của Use Case).
- Mô hình hóa lớp: Xác định các lớp thực thể và các thuộc tính của các lớp. Sau đó, xác định các mối quan hệ qua lại và các tương tác giữa các lớp. Biểu diễn thông tin này bằng biểu đồ lớp.
  - Mô hình hóa động: Xác định các hành động được thực hiện bởi hoặc đối với mỗi lớp thực thể hoặc các lớp con. Biểu diễn thông tin này dưới dạng các biểu đồ trạng thái.

Trong thực tế, ba bước trên không được thực hiện một cách tuần tự. Khi nào có một sự thay đổi ở một biểu đồ thì tương ứng sẽ có sự sửa đổi ở hai biểu đồ kia. Do đó, ba bước của phân tích hướng đối tượng được thực hiện song song một cách có hiệu quả.

### 7.3 PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG CHO BÀI TOÁN THANG MÁY

Một hệ thống được cài đặt để điều khiển n thang máy trong một tòa nhà với m tầng. Việc di chuyển thang máy giữa các tầng phải tuân theo những ràng buộc sau:

- Một là, mỗi thang máy có m nút, mỗi nút tương ứng với một tầng. Các nút này sáng lên khi được bấm và khi đó thang máy sẽ di chuyển tới tầng tương ứng mới số ghi trên nút. Và khi thang máy tới tầng đó thì nút đó sẽ hết sáng và trở lại bình thường.
- Hai là, ngoại trừ tầng đầu tiên và tầng trên cùng, các tầng khác đều có hai nút, một nút để yêu cầu thang máy đi lên và một nút yêu cầu thang máy đi xuống. Những nút này sẽ sáng lên khi được bấm. Và nút đó trở lại bình thường khi thang máy tới tầng tương ứng, sau đó thang máy sẽ di chuyển theo hướng được yêu cầu sau đó.
- Ba là, nếu thang máy không nhận được yêu cầu đi lên hoặc đi xuống, thì nó vẫn ở nguyên tầng đó và đóng cửa.

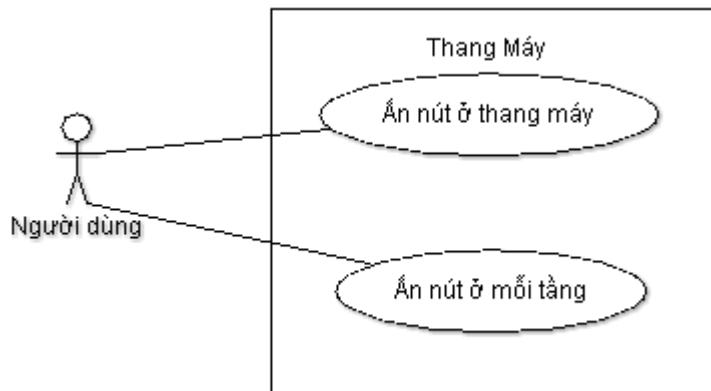
### 7.4 MÔ HÌNH HÓA CHỨC NĂNG

Một Use-Case miêu tả về những chức năng mà hệ thống phần mềm xây dựng. Một Use-Case đưa ra một miêu tả chung về chức năng toàn thể. Còn các kịch bản là những thể hiện cụ thể của các Use-Case. Các kịch bản cần được xem xét cẩn thận để có một cái nhìn toàn diện đối với hệ thống đích được xây dựng.

Use-Case miêu tả tương tác giữa hệ thống và các tác nhân (những người dùng bên ngoài).

Đối với bài toán thang máy, chỉ có hai Use-Case là : Án nút ở thang máy và Án nút ở mỗi tầng.

Sử dụng UML để biểu diễn Use-Case cho bài toán thang máy như hình 7.1



Hình 7.1 Các use case của bài toán thang máy

Những tương tác có thể giữa người dùng và các lớp đó là: một người dùng nhấn nút thang máy và ra lệnh cho thang máy di chuyển tới một tầng nào đó hoặc một người dùng nhấn nút để yêu cầu thang máy dừng lại ở một tầng cụ thể. Với mỗi một chức năng nói chung ta có thể đưa ra một số lượng lớn các kịch bản khác nhau, mỗi một kịch bản biểu diễn một tập các tương tác .

Hình 7.2 mô tả kịch bản chuẩn, nó bao gồm một tập các tương tác giữa người dùng và các thang máy tương ứng với cách mà thang máy được sử dụng.

Hình 7.2 được xây dựng sau khi đã quan sát tỉ mỉ những tương tác giữa người dùng với thang máy (chính xác hơn là với các nút của thang máy và các nút của các tầng). 15 sự kiện đã được đánh số miêu tả chi tiết gồm: hai tương tác giữa người dùng A và các nút của hệ thống thang máy (sự kiện 1 và sự kiện 7) và các thao tác của thang máy (sự kiện 2 đến 6 và 8 đến 15). Hai sự kiện người dùng A bước vào thang máy và người dùng A ra khỏi thang máy không được đánh số sự kiện. Những mục như vậy được xem như là những bình luận thêm, người dùng A không tương tác với các thành phần của hệ thống thang máy khi đã bước vào thang máy hoặc rời khỏi thang máy.

1. Người dùng A nhấn nút đi lên của tầng ba để yêu cầu thang máy và người dùng A muốn đi lên tầng 7.
2. Nút đi lên của tầng ba sáng lên.
3. Thang máy đến tầng 3. Trong thang máy đang có người dùng B, người dùng B đã vào thang máy từ tầng 1 và yêu cầu lên tầng 9.
4. Nút đi lên của tầng 3 trở lại trạng thái bình thường.
5. Cửa thang máy mở ra.
6. Máy bấm giờ bắt đầu. Người dùng A bước vào thang máy.
7. Người dùng A nhấn nút 7 của thang máy.
8. Nút 7 của thang máy sáng lên.
9. Cửa thang máy đóng lại sau một thời gian vượt quá thời gian quy định của máy bấm giờ.
10. Thang máy lên tới tầng 7.
11. Nút 7 của thang máy trở lại trạng thái bình thường.
12. Cửa thang máy mở và cho phép người dùng A ra khỏi thang máy.
13. Máy bấm giờ bắt đầu. Người dùng A bước ra khỏi thang máy.
14. Cửa thang máy đóng lại sau một thời gian quy định.
15. Thang máy di tiếp tục lên tầng 9 theo yêu cầu trước đó của người B.

Hình 7.2 Kịch bản chuẩn cho bài toán thang máy

Trái lại, hình 7.3 là một kịch bản ngoại lệ. Nó miêu tả những gì xảy ra khi người dùng nhấn nút đi lên ở tầng 3 nhưng thực sự muốn đi xuống tầng 1. Kịch bản này được xây dựng bởi việc quan sát các hành động của nhiều người trong thang máy.

1. Người dùng A nhấn nút đi lên của tầng ba để yêu cầu thang máy và người dùng A muốn đi xuống tầng 1.
2. Nút đi lên của tầng ba sáng lên.
3. Thang máy đến tầng 3. Trong thang máy đang có người dùng B, người dùng B đã vào thang máy từ tầng 1 và yêu cầu lên tầng 9.
4. Nút đi lên của tầng 3 trở lại trạng thái bình thường.
5. Cửa thang máy mở ra.
6. Máy bấm giờ bắt đầu. Người dùng A bước vào thang máy.
7. Người dùng A nhấn nút 1 của thang máy.
8. Nút 1 của thang máy sáng lên.
9. Cửa thang máy đóng lại sau một thời gian vượt quá thời gian quy định của máy bấm giờ.
10. Thang máy lên tới tầng 9.
11. Nút 9 của thang máy trở lại trạng thái bình thường.
12. Cửa thang máy mở và cho phép người dùng B ra khỏi thang máy.
13. Máy bấm giờ bắt đầu. Người dùng B bước ra khỏi thang máy.
14. Cửa thang máy đóng lại sau một thời gian quy định.
15. Thang máy đi tiếp tục xuống tầng 1 theo yêu cầu của người dùng A.

Hình 7.3: Kịch bản ngoại lệ của bài toán thang máy

## 7.5 MÔ HÌNH HÓA LỚP THỰC THẾ

Trong bước này, các lớp và các thuộc tính được trích rút và được biểu diễn bằng biểu đồ UML. Các thuộc tính của mỗi lớp thực thể được xác định ở bước này nhưng các phương thức thì không. Các phươn thức sẽ được gán cho các lớp ở pha thiết kế hướng đối tượng.

Có 3 cách thức để trích rút các lớp và các thuộc tính:

- Một là, suy luận ra các lớp từ các Use-Case và các kịch bản của các Use-Case. Phương pháp này có nhược điểm là thường có nhiều kịch bản do đó có quá nhiều lớp ứng cử để tìm ra lớp thực thể.
- Hai là, sử dụng CRC Cards (nếu chúng ta có tri thức về miền nghiệp vụ).
- Ba là, sử dụng phương pháp trích rút danh từ.

### 7.5.1 Trích rút danh từ

Đối với những người phát triển mà chưa có kiến thức chuyên môn thành thạo về nghiệp vụ của hệ thống xây dựng thì cách tốt nhất để trích rút ra các lớp ứng cử là sử dụng phương pháp trích rút danh từ với 2 giai đoạn như sau:

- Giai đoạn 1: Định nghĩa vấn đề một cách ngắn gọn: Miêu tả hệ thống phần mềm xây dựng một cách ngắn gọn, súc tích dưới dạng một đoạn văn duy nhất. Trong bài toán thang máy ta có đoạn văn như sau:  
“ Các nút trong thang máy và ở mỗi tầng điều khiển sự di chuyển của n thang máy trong tòa nhà m tầng. Các nút sáng lên khi có người bấm nút đó để yêu cầu thang máy di chuyển tới một tầng nào đó; nút đó sẽ trở lại trạng thái bình thường khi yêu cầu đã được

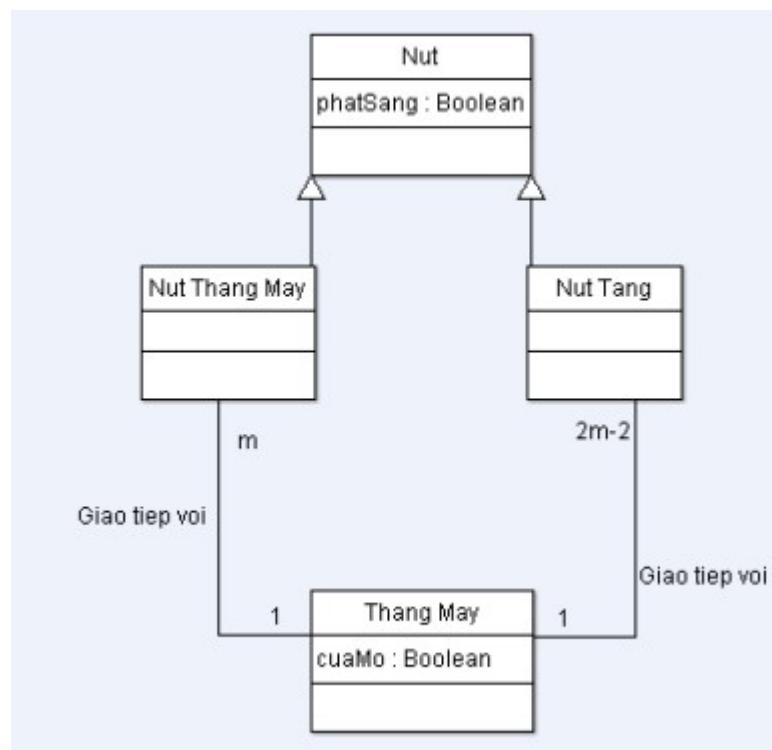
đáp ứng. Khi thang máy không nhận được yêu cầu nào thì nó vẫn ở tầng hiện tại và cửa vẫn đóng.”

- Giai đoạn 2: Xác định các danh từ: Xác định các danh từ theo chiến lược không hình thức. Trong bài toán thang máy có:

“Các nút trong thang máy và ở mỗi tầng điều khiển sự di chuyển của n thang máy trong tòa nhà m tầng. Các nút sáng lên khi có người bấm nút đó để yêu cầu thang máy di chuyển tới một tầng nào đó; nút đó sẽ trở lại trạng thái bình thường khi yêu cầu đã được đáp ứng. Khi thang máy không nhận được yêu cầu nào thì nó vẫn ở tầng hiện tại và cửa vẫn đóng”.

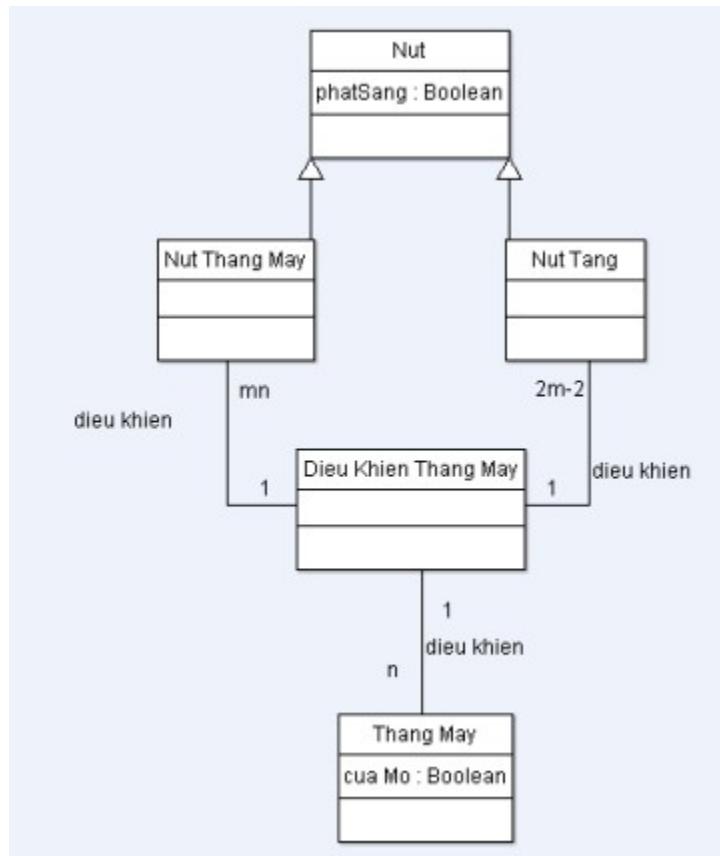
Sử dụng các danh từ trên như là các lớp ứng cử. Ta có các danh từ: Nút, thang máy, tầng, sự di chuyển, tòa nhà, yêu cầu, cửa. Trong đó, các danh từ: tầng, tòa nhà, cửa nằm bên ngoài biên của bài toán nên bị loại trừ. Còn danh từ sự di chuyển là danh từ trùu tượng nên bị loại trừ (chúng có thể trở thành các thuộc tính). Do đó, các lớp ứng cử là: thang máy và lớp nút và các lớp con: lớp nút thang máy và lớp nút tầng.

Biểu đồ lớp kết quả được biểu diễn bằng UML như hình 7.4



Hình 7.4 Bước lắp thứ nhất của biểu đồ lớp

Vấn đề xảy ra ở đây là trong thực tế các nút của thang máy không giao tiếp trực tiếp với thang máy. Khi đó thường yêu cầu có một vài loại điều khiển thang máy, và quyết định thang máy nào sẽ đáp ứng yêu cầu cụ thể. Tuy nhiên, trong phát biểu bài toán không đề cập đến lớp điều khiển, vì thế không có lớp điều khiển trong suốt quá trình trích rút danh từ. Mặt khác, kỹ thuật trích danh từ mà chúng ta sử dụng ở đây để tìm ra các lớp ứng cử được xem như điểm khởi đầu nhưng cũng không nên dựa hoàn toàn vào đó. Chúng ta cần thêm lớp điều khiển thang máy vào biểu đồ lớp hình 7.4. Đồng thời cũng thêm mối quan hệ 1-n giữa lớp điều khiển thang máy và lớp thang máy. Yêu tố này sẽ tạo điều kiện cho việc thiết kế và cài đặt dễ dàng hơn



Hình 7.5 Bước lặp thứ hai của biểu đồ lớp

### 7.5.2 CRC Cards

Trong nhiều năm về trước, CRC (Class-responsibility-collaboration) Card được sử dụng trong suốt pha phân tích [Wirfs-Brock, Wilkerson và Wiener, 1990]. Đối với mỗi lớp, đội phát triển phần mềm điền vào thẻ biểu diễn tên của lớp (name of Class), các chức năng của lớp (**Responsibility**) và danh sách các lớp khác có liên quan đến lớp đó để cùng nhau thực hiện các chức năng của lớp đó (**Collaboration**). Hiện nay, CRC card được thực hiện một cách tự động bằng cách sử dụng thành phần công cụ CASE.

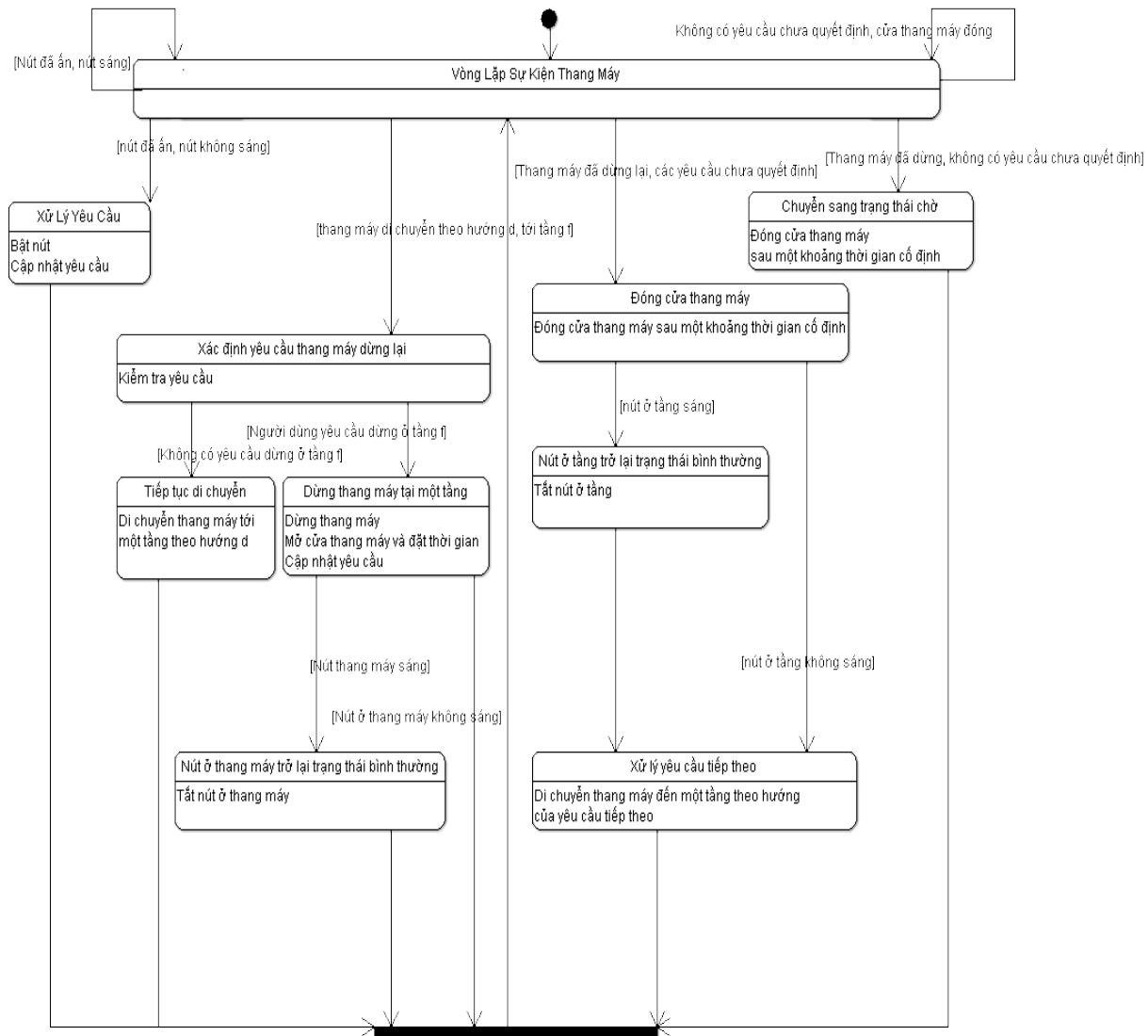
Điểm mạnh của CRC card là khi CRC card được thực hiện bởi các thành viên trong đội thì tương tác qua lại giữa các thành viên trong đội dễ dàng phát hiện ra những thiếu sót hoặc những mục không đúng trong CRC card.

Điểm yếu của CRC card là nếu chúng ta sử dụng CRC card để nhận dạng các lớp thực thể, thì yêu cầu có sự thành thạo về chuyên môn nghiệp vụ.

### 8.6 MÔ HÌNH HÓA ĐỘNG

Mục đích của việc mô hình hóa động là đưa ra biểu đồ tuần tự. Biểu đồ tuần tự mô tả hệ thống phần mềm cuối cùng gần giống với máy hữu hạn trạng thái đối với mỗi lớp. Hình 8.6 biểu diễn biểu đồ trạng thái của lớp Elevator Controller.

Khi biểu diễn biểu đồ trạng thái bằng UML gồm các 3 phần chính trạng thái, sự kiện và vị từ được phân bố trên toàn biểu đồ trạng thái. Ví dụ trạng thái **Chuyển sang trạng thái chờ** trong hình 8.6 được bắt đầu nếu trạng thái hiện tại là **Vòng lặp sự kiện thang máy** và vị từ **[Thang máy đã dừng, không có yêu cầu nào chưa quyết định]** là đúng. Khi trạng thái **Chuyển sang trạng thái chờ** đã bắt đầu thì hành động **Đóng cửa thang máy sau một khoảng thời gian cố định** được thực hiện.



Hình 7.6 Biểu diễn biểu đồ trạng thái của lớp Elevator Controller

Để thấy sự tương đương giữa biểu đồ trạng thái trong hình 7.6, phải xem xét trong các loại kịch bản khác nhau. Chẳng hạn, xem xét phần đầu tiên của kịch bản của hình 7.2. Đầu tiên, người dùng A nhấn nút đi lên ở tầng 3. Nếu nút ở tầng ba không sáng lên thì sau đó phải xử lý như hình 11.15 và trạng thái **Xử lý yêu cầu** của hình 7.6 xử lý tình huống nút được bật sáng lên. Trong trường hợp của biểu đồ trạng thái thì trạng thái tiếp theo sẽ là **Vòng lặp sự kiện thang máy**. Sau đó, thang máy tiến gần lại tầng 3. Trước tiên ta xem xét cách tiếp cận STD, trong hình 11.16, thang máy bắt đầu với trạng thái S(U, 3) có nghĩa là đi lên và dừng ở tầng 3 (giả định đơn giản hóa ở đây là chỉ có duy nhất một thang máy nên đối số e trong hình 11.16 được bỏ ở đây). Từ hình 11.15, khi mà thang máy đến tầng 3 thì nút đi lên ở tầng 3 bị tắt. (Theo hình 11.16) hiện

thời cửa thang máy đóng và thang máy bắt đầu di chuyển lên tầng 4. Quay trở lại hình 8.6 chúng ta xem xét chuyện gì sẽ xảy ra khi mà thang máy lên tới gần tầng 3. Bởi vì thang máy di chuyển nên trạng thái tiếp theo sẽ là **Xác định yêu cầu thang máy dừng lại**. Các yêu cầu được kiểm tra và bởi vì, người dùng A đã yêu cầu thang máy dừng ở đó, nên trạng thái tiếp theo là **Dừng thang máy tại một tầng**. Thang máy dừng ở tầng 3 và cửa thang máy mở. Nút thang máy ở tầng 3 không được nhấn nên trạng thái tiếp theo là **Vòng lặp sự kiện thang máy**. Người dùng A bước vào và nhấn nút thang máy lên tầng 7. Do đó, trạng thái tiếp theo lại là **Xử lý yêu cầu**, tiếp đó là trạng thái **Vòng lặp sự kiện thang máy**. Thang máy dừng lại và hai yêu cầu đang bị treo, vì thế trạng thái tiếp theo là **Đóng cửa thang máy** cửa đóng lại sau khoảng thời gian cố định. Nút ở tầng 3 được nhấn ở tầng 3 bởi người dùng A, vì thế trạng thái tiếp theo là **Nút ở tầng trở lại trạng thái bình thường** và nút đi lên ở tầng 3 trở về trạng thái bình thường. Trạng thái tiếp theo là **Xử lý yêu cầu tiếp theo** và thang máy bắt đầu di chuyển lên tầng 4. Rõ ràng những thành phần liên quan của biểu đồ tương ứng là tương đương với kịch bản này.

Trong thực tế, biểu đồ trạng thái được xây dựng từ việc mô hình hóa các sự kiện của kịch bản cụ thể. Chẳng hạn, xem xét sự kiện đầu tiên của kịch bản trong hình 8.2, người dùng A nhấn nút đi lên ở tầng ba. Sự kiện này được tổng quan hóa thành là một nút bất kỳ được nhấn. Có hai khả năng có thể xảy ra hoặc là nút sẵn sàng bật sáng (trong trường hợp không có gì xảy ra) hoặc là nút không được bật sáng (trong trường hợp hành động phải nắm bắt được để xử lý yêu cầu của người dùng). Để mô hình sự kiện này, trạng thái **Vòng lặp sự kiện thang máy** được đưa ra trong hình 8.6. Trường hợp nút sẵn sàng phát sáng được mô hình bởi vòng lặp không làm gì với vị từ **[nút đã án, nút sáng]** ở góc trái phía trên của hình 7.6. Trong trường hợp khác, nút không phát sáng, được mô hình bởi mũi tên được gán nhán là vị từ **[nút đã án, nút không sáng]** dẫn tới trạng thái **Xử lý yêu cầu**. Từ sự kiện 2 của kịch bản rõ ràng hành động **Bật nút** là cần thiết trong mỗi trạng thái **Xử lý yêu cầu**. Hơn nữa, mục đích của hành động của người dùng là việc nhấn một nút bất kỳ để yêu cầu thang máy, vì thế hành động **Cập nhật yêu cầu** cũng phải được thực thi ở trạng thái **Xử lý yêu cầu**.

Bây giờ chúng ta sẽ xem xét sự kiện 3 của kịch bản, thang máy tới tầng 3. Sự kiện này được mô hình hóa là thang máy bắt kỳ đang di chuyển giữa các tầng. Sự di của thang máy được mô hình bằng vị từ **[Thang máy di chuyển theo hướng d, tới tầng f]** và trạng thái tiếp theo là **Xác định yêu cầu thang máy dừng lại**. Và ở đây lại có hai khả năng xảy ra hoặc là có một yêu cầu dừng tại tầng f hoặc là không có yêu cầu nào như vậy. Trong trường hợp yêu cầu dừng tại từng f tương ứng với vị từ **[người dùng đã yêu cầu dừng thang máy ở tầng f]** và trạng thái tiếp theo là **Dừng thang máy tại một tầng** và có các hành động tương ứng với kịch bản ở hình 8.2 là **Dừng thang máy** (từ sự kiện 3), **Mở cửa và bắt đầu đêm giờ** (từ sự kiện 5, 6) và **Cập nhật yêu cầu**. Cuối cùng, mô hình hóa sự kiện 4 trong kịch bản đó là nút thang may sẽ trở lại trạng thái bình thường nếu nó sáng lên và được mô hình hóa bằng trạng thái **Nút thang máy trở lại trạng thái bình thường** (là trạng thái cuối cùng bên trái của hình 7.6), cùng với 2 vị từ ở phía trên của hộp trạng thái này.

Mô hình hóa sự kiện 9 của kịch bản trong hình 7.2 sinh ra trạng thái **Đóng cửa thang máy**; mô hình hóa sự kiện 10 tương ứng là trạng thái **Xử lý yêu cầu tiếp theo**. Tuy nhiên, rất cần thiết có trạng thái **Chuyển sang trạng thái chờ** và vị từ **[Không có yêu cầu chưa quyết định, cửa thang máy đóng]** được suy luận ra từ việc mô hình hóa một sự kiện của kịch bản khác, đó là kịch bản người dùng ra khỏi thang máy và không còn nút nào sáng.

## 7.6 KIỂM THỬ TRONG PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

Sau khi 3 mô hình của tiến trình phân tích hướng đối tượng được xem là hoàn thành thì bước tiếp theo là kiểm tra lại pha phân tích hướng đối tượng. CRC Cards là kỹ thuật kiểm thử tốt.

Các CRC cards đối với mỗi lớp **Nút**, **Nút Thang Máy**, **Nút Tầng**, **Thang Máy** và **Điều Khiển Thang Máy** được điền đầy đủ các thông tin. Ví dụ hình 8.7 được suy luận từ biểu đồ lớp 8.5 và biểu đồ trạng thái của hình 7.6. **Trách nhiệm** của lớp **Điều Khiển Thang Máy** chứa một loạt các hành động trong biểu đồ trạng thái của **Điều Khiển Thang Máy** (hình 7.6). Các **lớp cộng tác** của lớp **Điều Khiển Thang Máy** được xác định bằng cách xem xét biểu đồ lớp hình 7.5 và chú ý rằng các lớp **Nút Thang Máy**, **Nút Tầng** và **Thang Máy** tương tác với lớp **Điều Khiển Thang Máy**.

<b>Lớp Điều Khiển Thang Máy</b>	
Trách nhiệm	1. Bật nút thang máy 2. Tắt nút thang máy 3. Bật nút ở tầng 4. Tắt nút ở tầng 5. Di chuyển thang máy lên một tầng nào đó 6. Di chuyển thang máy xuống một tầng nào đó 7. Mở cửa thang máy và bắt đầu đếm giờ 8. Đóng cửa thang máy sau một khoảng thời gian nhất định 9. Kiểm tra yêu cầu 10. Cập nhật yêu cầu
Cộng tác	1. <b>Lớp Nút Thang Máy</b> 2. <b>Lớp Nút Tầng</b> 3. <b>Lớp Thang Máy</b>

Hình 7.7 Vòng lặp thứ nhất cho CRC Card đối với lớp Elevator Controller

CRC Card này nổi bật với hai vấn đề chính trong vòng lặp đầu tiên của phân tích hướng đối tượng. Trước tiên hãy xem xét trách nhiệm 1.*Bật nút thang máy*. Yêu cầu này nói chung nằm ngoài phạm vi của mô hình hướng đối tượng. Từ quan điểm của thiết kế hướng trách nhiệm (1.6), chính các đối tượng của lớp **Nút Thang Máy** chịu trách nhiệm bật và tắt. Từ quan điểm ẩn dấu thông tin (7.6) **Điều Khiển Thang Máy** cần biết về khả năng của **Nút Thang Máy** là có thể bật một nút. Khi đó trách nhiệm đó phải được sửa lại là : gửi một thông điệp tới nút **Nút Thang Máy** để bật một nút. Tương tự ta cần phải thay đổi đối với các trách nhiệm 2 đến 6 trong hình 7.7. Sự sửa đổi này được phản ánh trong hình 7.8, vòng lặp thứ 2 của CRC Card đối với lớp **Điều Khiển Thang Máy**.

Vấn đề thứ hai là một lớp đã bị bỏ qua. Xem xét trách nhiệm 7.*Mở cửa thang máy và bắt đầu đếm giờ*. Khái niệm chính ở đây chính là trạng thái. Các thuộc tính của một lớp đôi khi được gọi là biến trạng thái. Trong hầu hết pha cài đặt hướng đối tượng, trạng thái của hệ thống phần mềm được định nghĩa bởi giá trị của các thuộc tính của các loại đối tượng thành phần khác nhau. Biểu đồ trạng thái có nhiều đặc trưng chung so với máy hữu hạn trạng thái. Vì thế khái niệm trạng thái đóng vai trò quan trọng trong mô hình hướng đối tượng. Khái niệm này được sử dụng để xác định liệu một thành phần có nên được mô hình như một lớp không? Nếu thành phần có trạng thái sẽ bị thay đổi trong suốt quá trình thực thi của sự cài đặt thì nó có thể được mô hình như một lớp. Rõ ràng, cửa của thang máy có trạng thái (đóng và mở) và do đó nên mô hình **Cửa Thang Máy** là một lớp.

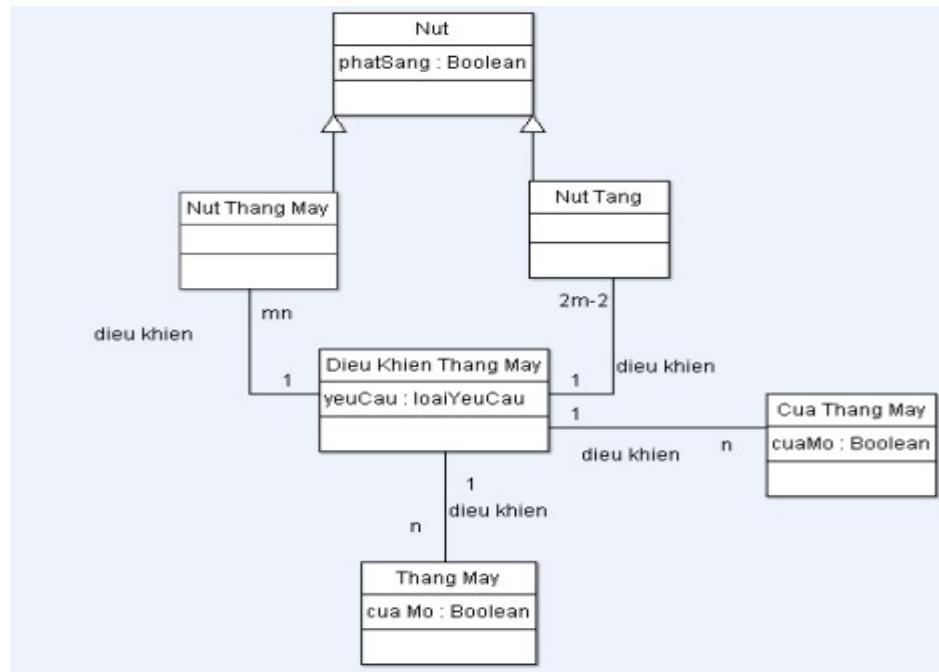
Có một lý do khác lý giải tại sao **Cửa Thang Máy** nên là một lớp. Mô hình hướng đối tượng cho phép trạng thái được ẩn dấu bên trong đối tượng và do đó trạng thái được bảo vệ khỏi những sự thay đổi không cho phép. Nếu có một đối tượng **Cửa Thang Máy**, thì có duy nhất một cách để đóng hoặc mở cửa của thanh máy bằng cách gửi một thông điệp tới đối tượng **Cửa Thang Máy**.

Việc đưa thêm lớp **Cửa Thang Máy** có nghĩa là trách nhiệm 7 và 8 trong hình 8.7 được thay đổi tương tự như trách nhiệm 1 cho đến 6. Có một thông điệp gửi tới lớp **Cửa Thang Máy** để tự đối tượng đó đóng và mở. Nhưng có thêm sự phức tạp, trách nhiệm 7 là *Mở cửa thang máy và bắt đầu đêm giờ* phải được phân tách thành 2 trách nhiệm riêng biệt. Một trách nhiệm tương ứng là gửi thông điệp tới **Cửa Thang Máy** để mở cửa thang máy. Vì máy bấm giờ là một phần của **Điều Khiển Thang Máy** do đó việc bắt đầu tính giờ là trách nhiệm của **Điều Khiển Thang Máy**. Vòng lặp thứ hai của CRC Card đối với lớp **Điều Khiển Thang Máy** được biểu diễn như hình 7.8. Trong hình 7.8 cũng thêm vào hai trách nhiệm là *Kiểm tra yêu cầu* và *Cập nhật yêu cầu* đối với lớp **Điều Khiển Thang Máy**.

<b>Lớp Điều Khiển Thang Máy</b>	
Trách nhiệm	
	<ol style="list-style-type: none"> <li>Gửi thông điệp tới <b>Lớp Nút Thang Máy</b> để bật nút</li> <li>Gửi thông điệp tới <b>Lớp Nút Thang Máy</b> để tắt nút</li> <li>Gửi thông điệp tới <b>Lớp Nút Tầng</b> để bật nút</li> <li>Gửi thông điệp tới <b>Lớp Nút Tầng</b> để tắt nút</li> <li>Gửi thông điệp tới <b>Lớp Thang Máy</b> để di chuyển lên một tầng nào đó.</li> <li>Gửi thông điệp tới <b>Lớp Thang Máy</b> để di chuyển xuống một tầng nào đó.</li> <li>Gửi thông điệp tới <b>Lớp Cửa Thang Máy</b> để mở cửa</li> <li>Bắt đầu đêm giờ</li> <li>Gửi thông điệp tới <b>Lớp Cửa Thang Máy</b> để đóng cửa sau một khoảng thời gian nhất định.</li> <li>Kiểm tra yêu cầu</li> <li>Cập nhật yêu cầu</li> </ol>
Cộng tác	<ol style="list-style-type: none"> <li><b>Lớp Nút Thang Máy</b> (lớp con)</li> <li><b>Lớp Nút Tầng</b> (lớp con)</li> <li><b>Lớp Cửa Thang Máy</b></li> <li><b>Lớp Thang Máy</b></li> </ol>

Hình 7.8 Vòng lặp thứ 2 của CRC Card của lớp Elevator Controller

Biểu đồ lớp được sửa lại như hình 7.9. Sau khi chỉnh sửa biểu đồ lớp, thì biểu đồ Use-Case và biểu đồ trạng thái cũng phải xem xét lại, nếu cần thiết sẽ làm mịn hơn nữa. Biểu đồ Use case vẫn đầy đủ. Tuy nhiên, các hành động trong biểu đồ trạng thái hình 7.6 phải được chỉnh sửa để phản ánh đầy đủ những trách nhiệm trong 7.8 (vòng lặp thứ hai của CRC Card). Tập các biểu đồ trạng thái phải được mở rộng vì có thêm một lớp mới. Kịch bản cần được cập nhật để thể hiện sự thay đổi; Hình 7.10 biểu diễn vòng lặp thứ hai của kịch bản hình 7.2. Mặc dù sau tất cả những thay đổi đã được đưa ra và kiểm tra nhưng trong suotό pha thiết kế hướng đối tượng vẫn phải quay trở lại phân tích hướng đối tượng và xem xét lại một hoặc nhiều biểu đồ.



Hình 7.9 Vòng lặp thứ ba của biểu đồ lớp

1. Người dùng A nhấn nút đi lên của tầng ba để yêu cầu thang máy và người dùng A muốn đi lên tầng 7.
2. Nút ở tầng 3 thông báo cho bộ điều khiển thang máy là nút ở tầng đã được nhấn vào.
3. Điều khiển thang máy gửi một thông điệp tới nút đi lên ở tầng 3 để nút đó tự bật sáng.
4. Điều khiển thang máy gửi một loạt các thông điệp tới thang máy để thang máy tự di chuyển lên tầng 3. Trong thang máy hiện có người dùng B đã vào thang máy từ tầng một và yêu cầu lên tầng 9.
5. Điều khiển thang máy gửi một thông điệp tới cửa thang máy để mở cửa.
6. Thang máy bắt đầu tính thời gian. Người dùng A bước vào thang máy.
7. Người dùng A nhấn nút thang máy lên tầng 7.
8. Nút thang máy thông báo tới điều khiển thang máy rằng là nút thang máy đã được nhấn vào.
9. Bộ điều khiển thang máy gửi một thông điệp tới nút số 7 của thang máy để nút đó được bật sáng.
10. Bộ điều khiển thang máy gửi một thông điệp tới cửa thang máy để đóng thang máy sau một khoảng thời gian cố định.
11. Bộ điều khiển gửi một thông điệp tới nút đi lên ở tầng 3 để nó trở về trạng thái bình thường.
12. Bộ điều khiển thang máy gửi một loạt thông điệp tới thang máy để nó di chuyển lên tới tầng 7.
13. Bộ điều khiển thang máy gửi một thông điệp tới nút 7 ở thang máy để nó trở về trạng thái bình thường (không sáng).
14. Bộ điều khiển thang máy gửi một thông điệp tới cửa thang máy yêu cầu mở cửa để cho phép người dùng A bước ra khỏi thang máy.
15. Bộ điều khiển thang máy bắt đầu đặt thời gian. Người dùng A bước ra khỏi thang máy.
16. Bộ điều khiển thang máy gửi một thông điệp tới cửa thang máy để đóng cửa sau một thời gian cố định.
17. Bộ điều khiển thang máy gửi một loạt thông điệp tới thang máy để nó di chuyển lên tầng 9.

## 7.7 CÁC CÔNG CỤ CASE CHO PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

Các biểu đồ đóng vai trò quan trọng trong phân tích hướng đối tượng. Các biểu đồ thường xuyên thay đổi do đó cần có các công cụ vẽ biểu đồ.

Các công cụ hỗ trợ cho UML:

- Các công cụ mang tính thương mại như: IBM Rational Rose và Together.
- Các công cụ Open-source: AgroUML

## 7.8 CASE STUDY CHO PHÂN TÍCH HƯỚNG ĐỐI TƯỢNG

### 7.8.1. Các kịch bản

Mục đích của bước này là viết các kịch bản (scenario) cho các use case đã xác định được trong pha lấy yêu cầu. Để tập trung vào phần nghiệp vụ chuyên môn, các kịch bản trong mục này sẽ bỏ qua phần đăng nhập của các nhân viên cho ngắn gọn.

#### a. Kịch bản sửa thông tin phòng

1. Nhân viên quản lý (QL) vào hệ thống để sửa thông tin phòng 305 của khách sạn.
2. Hệ thống hiện giao diện đăng nhập, có ô nhập tên đăng nhập, mật khẩu, và nút đăng nhập.
3. Nhân viên nhập thông tin tài khoản của mình và click đăng nhập
4. Hệ thống hiện giao diện chính của nhân viên quản lý, có 3 chức năng lựa chọn: quản lý thông tin khách sạn, quản lý thông tin phòng, xem thống kê.
5. Nhân viên chọn chức năng quản lý phòng.
6. Hệ thống hiện giao diện quản lý phòng, có 3 chức năng lựa chọn: thêm, sửa, xóa phòng.
7. Nhân viên chọn vào chức năng sửa phòng.
8. Hệ thống hiển thị giao diện tìm phòng để sửa: ô nhập mã (tên) phòng, nút tìm.
9. QL nhập tên phòng 305 và click vào nút Tìm phòng
10. Hệ thống hiển thị danh sách các phòng tìm thấy dưới dạng bảng:

<b>id</b>	<b>Tên</b>	<b>Giá</b>	<b>Kiểu</b>	<b>Mô tả</b>
1	305	1000	Double	Sea view
2	305bis	500	Single	Garden view

11. QL click vào dòng thứ nhất, tương ứng phòng 305.
12. Hệ thống hiện giao diện sửa thông tin phòng 305 với các thuộc tính chưa sẵn thông tin:
  - id: 1 (không sửa được)
  - name: 305
  - price: 1000
  - type: double
  - description: see view
  - nút cập nhật và reset.
13. QL sửa giá phòng thành 800 và click nút cập nhật.
14. Hệ thống báo thành công và quay về trang chủ của người quản lý.

Ngoại lệ:

4. Hệ thống báo không có phòng nào trong kết quả tìm kiếm

### b. Kịch bản đặt phòng

Đặt phòng qua điện thoại (bỏ qua các bước đăng nhập):

1. Nhân viên bán hàng A click vào chức năng đặt phòng trong menu quản lý đặt phòng. Nhân viên bán hàng A muốn đặt phòng cho khách hàng B, người đang gọi điện thoại yêu cầu đặt phòng.
2. Hệ thống hiển thị giao diện yêu cầu nhập khoảng thời gian muốn đặt của khách hàng: ngày bắt đầu, ngày kết thúc.
3. Nhân viên A hỏi lại khách hàng B ngày bắt đầu và kết thúc theo mong muốn
4. Khách hàng B trả lời muốn đặt phòng từ 30/04/2020 đến 01/05/2020.
5. Nhân viên A nhập ngày bắt đầu là 30/04/2020, ngày kết thúc là 01/05/2020 và click vào nút tìm phòng trống.
6. Hệ thống hiển thị danh sách các phòng trống trong khoảng thời gian đã nhập.

<b>id</b>	<b>Tên</b>	<b>Giá</b>	<b>Kiểu</b>	<b>Mô tả</b>
1	305	1000	Double	Sea view
2	201	500	Single	Garden view
3	202	1000	Twink	Garden view

7. Nhân viên A thông báo các phòng có thể đặt cho khách hàng B và yêu cầu khách hàng B chọn 1 phòng trong số đó.
8. Khách hàng B chọn phòng đôi nhìn ra biển
9. Nhân viên A click vào phòng 305 (dòng số 1)
10. Hệ thống hiện lên giao diện yêu cầu nhập thông tin khách hàng: tên, số CMT (passport), địa chỉ, email, điện thoại.
11. Nhân viên A hỏi lại khách hàng B các thông tin trên.
12. Khách hàng khai: Tên là B, địa chỉ Hà Nội, số thẻ id 123456, số điện thoại 77777777, và email là b77@gmail.com.
13. Nhân viên nhập B vào ô tên. Sau đó click vào tìm kiếm.
14. Hệ thống hiện danh sách các khách hàng có tên chứa chữ B trong hệ thống:

<b>id</b>	<b>Tên</b>	<b>Địa chỉ</b>	<b>Số thẻ ID</b>	<b>Số ĐT</b>	<b>Email</b>
1	B	Hà Nội	123456	77777777	b77@gmail.com
2	BC	Đà Nẵng	223344	88888888	bc88@Gmail.com
3	BB	TP. HCM	343434	55555555	null

15. Nhân viên nhận thấy khách hàng B có đúng các thông tin trùng với dòng thứ 1. Do đó, nhân viên click chọn vào dòng số 1
16. Hệ thống hiện giao diện xác nhận thông tin đặt phòng. Bao gồm thông tin phòng 305, phòng đôi, hướng biển, giá 1tr/đêm. Thông tin khách hàng B, đến từ Hà Nội... Thông tin đặt phòng từ 30/04 đến 01/05 năm 2020. Nút xác nhận và hủy bỏ.

- 17. Nhân viên xác nhận các thông tin với khách hàng trước khi lưu.
- 18. Khách hàng xác nhận lại OK, đúng hết.
- 19. Nhân viên click vào nút xác nhận (lưu)
- 20. Hệ thống thông báo đặt phòng thành công và quay về trang chủ nhân viên bán hàng.
- 21. Nhân viên A thông báo cho khách hàng B đã đặt phòng thành công.

Ngoại lệ:

- 6. Hệ thống báo không còn phòng trống trong khoảng ngày đã chọn
- 8. Khách hàng không chọn phòng nào trong số các phòng do nhân viên thông báo
- 12. Không có khách hàng nào hiện lên trong kết quả tìm kiếm. Hoặc không có khách hàng đúng thông tin với khách đang đặt.

#### c. Kịch bản nhận phòng

- 1. Chiều ngày 30/04/2020, nhân viên lễ tân A click vào chức năng nhận phòng trong menu quản lý khách nhận phòng. Nhân viên lễ tân A muốn cho khách hàng B nhận phòng theo yêu cầu của B.
- 2. Hệ thống hiển thị giao diện tìm thông tin đặt phòng theo khách hàng: 1 ô nhập id, 1 ô nhập tên và một nút tìm kiếm
- 3. Nhân viên A hỏi mượn khách hàng B thẻ CCCD/hộ chiếu
- 4. Khách hàng B đưa thẻ CCCD của mình cho nhân viên A
- 5. Nhân viên A nhập tên của khách hàng là B ô tìm kiếm và click vào nút tìm kiếm.
- 6. Hệ thống hiển thị thông tin đặt phòng của khách hàng có tên chứa chữ B, có ngày nhận phòng đúng hôm đó:

TT	Tên	Địa chỉ	Số thẻ ID	Số ĐT	ngày đến	ngày đi	tên phòng	kiểu
1	B	Hà Nội	123456	77777777	30/04/20	01/05/20	305	Double
2	BC	Đà Nẵng	223344	88888888	30/04/20	01/05/20	201	Single
3	BB	TP. HCM	343434	5555555	30/04/20	02/05/20	202	Twink

- 7. Nhân viên A yêu cầu khách hàng B và xác nhận thông tin phòng đặt: ngày đến và đi.
- 8. Khách hàng B xác nhận ngày đến là 30/04/2020, ngày đi là 01/05/2020 cho nhân viên A
- 9. Nhân viên A click vào dòng số 1 trên danh sách.
- 10. Hệ thống thông báo nhận phòng thành công.
- 11. Nhân viên A xác nhận và trao chìa khóa phòng cho khách hàng B

Ngoại lệ:

- 6. Bảng kết quả không có thông tin, hoặc không có thông tin của khách đang nhận phòng.

#### d. Kịch bản trả phòng

- Sáng ngày 01/05/2020, nhân viên lễ tân A click vào chức năng trả phòng trong menu quản lý khách nhận phòng. Nhân viên lễ tân A muốn làm thủ tục trả phòng do khách hàng B yêu cầu.
- Hệ thống hiển thị giao diện tìm thông tin đặt phòng: 1 ô nhập tên phòng và một nút tìm kiếm
- Nhân viên A hỏi lại khách hàng B số hiệu phòng muốn trả
- Khách hàng B trả lời số hiệu phòng cho nhân viên A là 305
- Nhân viên A nhập số hiệu phòng của khách hàng B ô tìm kiếm và click vào nút tìm kiếm.
- Hệ thống hiển thị thông tin đặt của phòng 305 có ngày checkout đúng hôm đó:

TT	Tên	Địa chỉ	Số thẻ ID	Số ĐT	ngày đến	ngày đi	tên phòng	kiểu
1	B	Hà Nội	123456	77777777	30/04/20	01/05/20	305	Double

- Nhân viên A click vào phòng 305 trên giao diện, và chọn trả phòng
- Hệ thống hiển thị thông tin hóa đơn cần thanh toán bao gồm các thông tin: mã hóa đơn, ngày tạo, tên và địa chỉ khách hàng, số hiệu phòng, kiểu phòng, đơn giá. Dòng tiếp theo ghi tổng số tiền của hóa đơn, một dòng ghi số tổng số tiền đã thanh toán trước đó, dòng cuối ghi tổng số tiền còn lại khách hàng phải thanh toán. Bên dưới có thêm một nút nhấn “bổ sung các dịch vụ”
- Nhân viên A thông báo số tiền phải trả cho khách hàng B
- Khách hàng B thanh toán cho nhân viên A số tiền yêu cầu.
- Nhân viên A click vào nút xác nhận thanh toán
- Hệ thống thông báo trả phòng thành công.
- Nhân viên A thông báo lại cho khách hàng B kết thúc thành công giao dịch.

Ngoại lệ:

- Không thấy kết quả tìm kiếm
- Nhân viên click vào bổ sung dịch vụ đã sử dụng.

#### e. Kịch bản xem báo cáo thống kê

- Nhân viên quản lí A click vào chức năng xem báo cáo thống kê sau khi đăng nhập. Nhân viên A muốn xem báo cáo doanh thu theo phòng từ đầu tháng.
- Hệ thống hiển thị giao diện chọn đối tượng thống kê: thống kê phòng, khách hàng, hay dịch vụ. Ô thứ hai chọn kiểu thống kê theo doanh thu hay theo thời gian.
- Nhân viên A chọn thống kê theo phòng, loại thống kê theo doanh thu.
- Giao diện thống kê phòng theo doanh thu hiện ra, có ô nhập ngày bắt đầu, ngày kết thúc thống kê
- Nhân viên A nhập ngày bắt đầu 01/05/2020, kết thúc là 30/05/2020, click thống kê.
- Hệ thống hiển thị thông tin thống kê các phòng:

TT	Tên	kiểu	tổng số ngày có khách	tổng doanh thu
1	305	double	15	15000
2	201	single	24	12000
3	202	twink	10	10000
4	203	double	5	5000

5	301	single	2	1000
6	302	twink	0	0

7. Nhân viên A click vào phòng 305 để xem chi tiết.

8. Hệ thống hiển thị danh sách các khách hàng đã ở phòng 305 trong tháng 05/2020:

TT	Tên khách	ngày đến	ngày đi	đơn giá	tổng ngày ở	tổng doanh thu
1	B	01/05/20	05/05/20	1000	4	4000
2	CC	09/05/20	10/05/20	1000	1	1000
3	zz	13/05/20	21/05/20	1000	8	8000
4	kk	24/05/20	26/05/20	1000	2	2000
<b>Tổng</b>					<b>15</b>	<b>15000</b>

9. Nhân viên A click quay lại

10. Hệ thống quay về trang chủ của người quản lí.

Ngoại lệ:

6. Các phòng đều có doanh thu 0

### 7.8.2 Trích các lớp thực thể

Mô tả hệ thống trong một đoạn văn như sau:

Hệ thống quản lí thông tin về phòng của khách sạn, thông tin về khách hàng đặt phòng. Hệ thống cho phép người quản lí có thể quản lí thông tin về phòng và khách sạn, xem các loại báo cáo thống kê về phòng theo doanh thu, thống kê khách hàng theo doanh thu, thống kê dịch vụ theo doanh thu, thống kê doanh thu theo thời gian (năm/tháng/quý). Hệ thống cũng cho phép khách hàng đặt phòng qua điện thoại thông qua nhân viên bán hàng, hoặc đặt phòng trực tiếp tại quầy thông qua nhân viên lễ tân. Hệ thống cũng cho phép nhân viên lễ tân thực hiện các hoạt động như nhận phòng, trả phòng, thanh toán khi có yêu cầu từ khách hàng. Mỗi khi thanh toán, hóa đơn sẽ được in ra cho khách hàng, bao gồm cả phí các dịch vụ mà khách hàng đã sử dụng khi nghỉ tại khách sạn.

Như vậy, ta có các danh từ và các phân tích như sau:

- Hệ thống: danh từ chung chung --> loại.
- Thông tin: danh từ chung chung --> loại.
- Phòng: là đối tượng xử lí của hệ thống --> là 1 lớp thực thể: Room
- Khách sạn: là đối tượng xử lí của hệ thống --> là 1 lớp thực thể: Hotel

- Khách hàng: là đối tượng xử lí của hệ thống --> là 1 lớp thực thể: Client
- Người quản lí: không phải là đối tượng xử lí trực tiếp của hệ thống, nhưng cũng bị quản lí cùng với nhân viên lễ tân và nhân viên bán hàng theo kiểu người dùng trực tiếp của phần mềm --> đê xuất là 1 lớp thực thể chung: User
- Điện thoại: không thuộc phạm vi xử lí của phần mềm --> loại
- Nhân viên bán hàng: không phải là đối tượng xử lí trực tiếp của hệ thống, nhưng cũng bị quản lí cùng với người quản lí và nhân viên lễ tân theo kiểu người dùng trực tiếp của phần mềm --> đê xuất là 1 lớp thực thể chung: User
- Quầy: không thuộc phạm vi xử lí của phần mềm --> loại
- Nhân viên lễ tân: không phải là đối tượng xử lí trực tiếp của hệ thống, nhưng cũng bị quản lí cùng với người quản lí và nhân viên bán hàng theo kiểu người dùng trực tiếp của phần mềm --> đê xuất là 1 lớp thực thể chung: User
- Hóa đơn: là đối tượng xử lí của hệ thống --> là 1 lớp thực thể: Bill
- Phí, lệ phí, tiền: trừu tượng, chung chung --> loại (có thể làm thuộc tính của hóa đơn)
- Dịch vụ: là đối tượng xử lí của hệ thống --> là 1 lớp thực thể: Service
- Các thông tin thống kê: thống kê phòng --> RoomStat; thống kê khách hàng --> ClientStat; thống kê dịch vụ --> ServiceStat; thống kê khách sạn --> HotelStat; thống kê doanh thu --> IncomeStat.

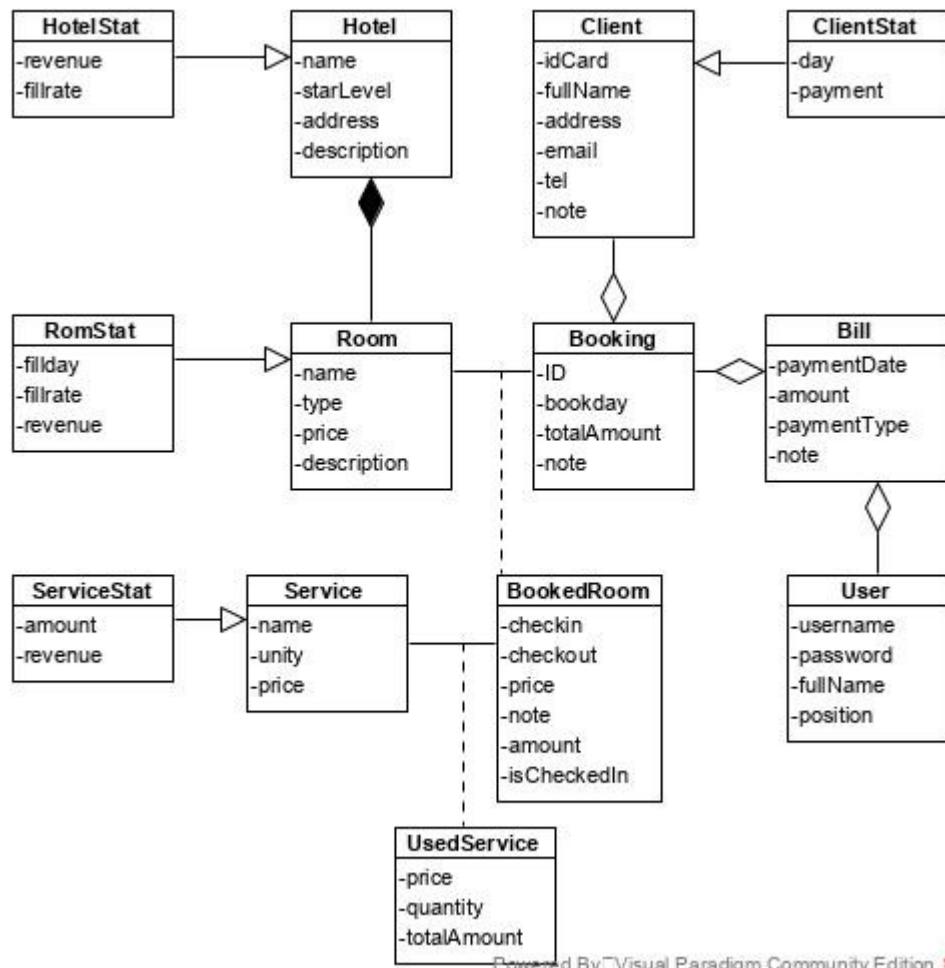
Vậy chúng ta thu được các lớp thực thể ban đầu là: Room, Hotel, Client, User, Bill, Service và các lớp thực thể thống kê: RoomStat, HotelStat, ClientStat, ServiceStat, IncomeStat.

Quan hệ giữa các lớp thực thể được xác định như sau:

- Một Hotel có nhiều Room, một Room chỉ thuộc vào một Hotel. Vậy quan hệ giữa Hotel và Room là 1-n.
- Một Client có thể đặt nhiều Room, một Room có thể bị đặt trước bởi nhiều Client ở nhiều thời điểm khác nhau: quan hệ giữa Client và Room là n-n. Do đó có thể bổ sung một lớp thực thể liên kết giữa hai đối tượng này là Booking (thông tin đặt phòng).
- Một Client có thể có nhiều Booking. Một phòng cũng có thể có nhiều Booking. Tuy nhiên, một lần Booking, khách hàng có thể đặt nhiều phòng (đại diện khách đoàn). Do đó, quan hệ giữa Booking và Room vẫn còn là n-n. Chúng ta cần tách thành một lớp trung gian ở giữa nữa là BookedRoom (thông tin phòng được đặt). Một Boooking và một Room

xác định duy nhất một BookedRoom. Liên kết này xác định thêm các thông tin: ngày đến, ngày đi, giá thực.

- Một Booking có thể được thanh toán nhiều lần khác nhau. Do đó có nhiều hóa đơn khác nhau. Vậy quan hệ giữa Booking và Bill là 1-n.
- Một nhân viên lễ tân có thể lập nhiều hóa đơn khác nhau cho các Booking khác nhau. Do đó quan hệ giữa User và Bill cũng là 1-n.
- Một Service có thể dùng bởi nhiều Client khác nhau, tại nhiều phòng đặt BookedRoom khác nhau. Nhưng một Service chỉ được thanh toán 1 lần (có thể với nhiều đơn vị) tại một BookedRoom. Một BookedRoom có thể dùng nhiều Service khác nhau: Quan hệ giữa BookedRoom và Service là n-n. Do đó đề xuất bổ sung lớp UsedService làm cầu nối 1-n giữa lớp BookedRoom và Service.



Hình 7.10: Biểu đồ lớp thực thể pha phân tích

- Đối với các lớp thông kê, do chúng phải dùng lại một số thuộc tính của lớp thực thể tương ứng nên có quan hệ kế thừa từ các lớp thực thể tương ứng ấy: HotelStat kế thừa từ Hotel; RoomStat kế thừa từ Room; ServiceStat kế thừa từ Service; ClientStat kế thừa từ Client.
- Riêng lớp IncomeStat không dùng lại thuộc tính của lớp nào nên không có quan hệ kế thừa với lớp nào. Nó chỉ có quan hệ với lớp Bill vì có thể liên quan đến chi phí trong các hóa đơn.

Như vậy, ta thu được sơ đồ các lớp thực thể của hệ thống như Hình 7.10.

### 7.8.3 Phân tích chi tiết từng modul

Nội dung phần này sẽ phân tích chi tiết hoạt động từng modul với hai bước: phân tích tĩnh và phân tích động. Phân tích tĩnh là phân tích các lớp biên và lớp đối tượng cần thiết trong chức năng (biểu đồ lớp). Phân tích động là phân tích các bước hoạt động của chức năng (biểu đồ tuần tự/cộng tác). Các chức năng được phân tích ở đây là: sửa thông tin phòng, đặt phòng, thống kê phòng theo doanh thu.

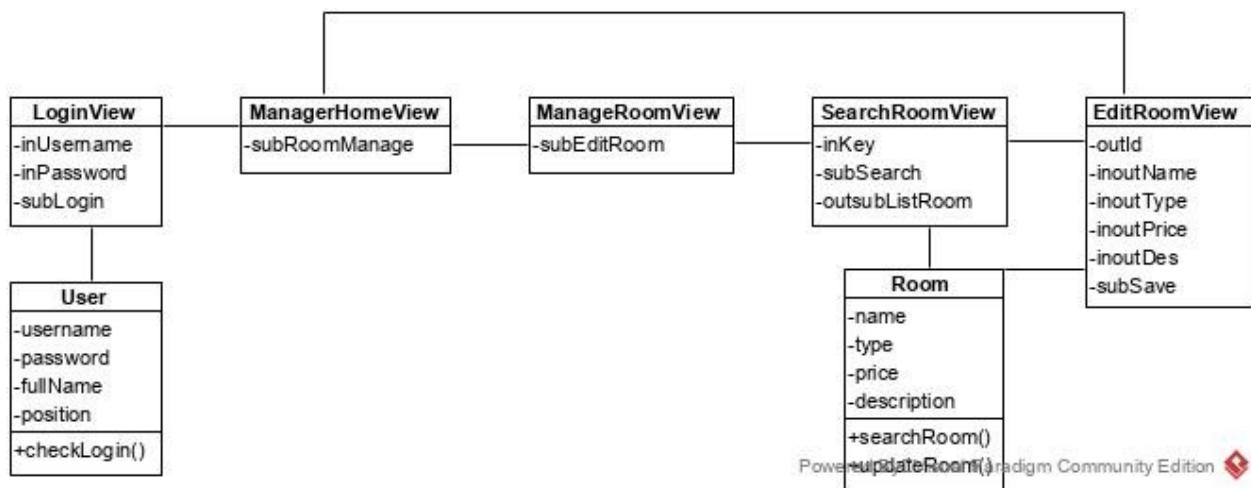
#### a. Chức năng sửa thông tin phòng

Phân tích chi tiết chức năng sửa thông tin phòng diễn ra như sau:

- Vào hệ thống -> giao diện login hiện lên -> để xuất lớp LoginView, có 2 ô nhập username, password và nút Login.
- Nhập username/password -> hệ thống phải kiểm tra thông tin đăng nhập -> cần chức năng checkLogin() -> chức năng này là hành động của đối tượng User.
- Login thành công, hệ thống hiện giao diện chính của quản lí -> để xuất lớp ManagerHomeView, có ít nhất nút chọn vào quản lí thông tin phòng.
- Click vào nút quản lí thông tin phòng -> giao diện quản lí phòng hiện lên -> để xuất lớp RoomManageView, có ít nhất nút sửa phòng.
- Click nút sửa phòng -> giao diện tìm phòng để sửa hiện lên -> để xuất lớp SearchRoomView, có ô nhập tên phòng để tìm, nút tìm, và bảng danh sách kết quả.
- Nhập từ khóa, click nút tìm -> hệ thống tìm phòng có tên vừa nhập -> cần chức năng searchRoom() -> chức năng này là hành động của đối tượng Room.
- Tìm xong, danh sách phòng tìm thấy sẽ hiện lên giao diện tìm phòng SearchRoomView.
- Click vào một phòng -> hiển thị giao diện chi tiết phòng -> để xuất lớp EditRoomView, có các ô edit: tên, kiểu, giá, mô tả, nút lưu.

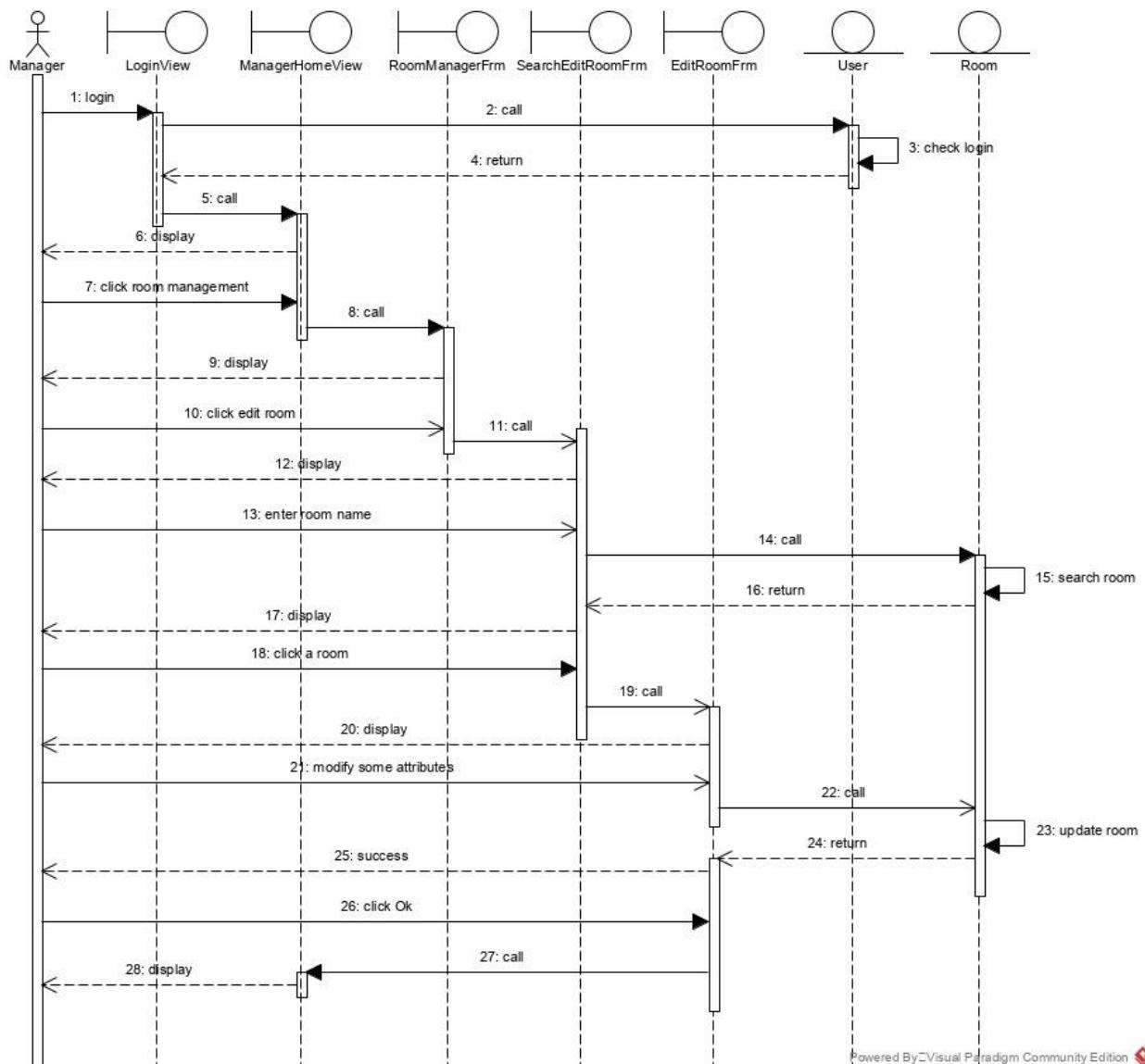
- Chính sửa thông tin xong nhấn nút lưu -> hệ thống lưu vào CSDL -> cần chức năng updateRoom() -> chức năng này là hành động của đối tượng Room.
- Cập nhật xong, hệ thống quay về giao diện chính của nhân viên quản lý ManagerHomeView.

Như vậy, kết quả thu được biểu đồ lớp cho chức năng sửa thông tin phòng như trong Hình 7.11. Với biểu đồ lớp trong Hình 7.11, kịch bản chi tiết cho chức năng sửa thông tin phòng diễn ra như sau (Hình 7.12):



Hình 7.11: Biểu đồ lớp phân tích chức năng sửa thông tin phòng

- Nhân viên quản lí nhập username/password vào giao diện đăng nhập và click nút Login.
- Lớp LoginView gọi đến lớp User để xử lí.
- Lớp User gọi hàm kiểm tra đăng nhập. Kết quả đăng nhập thành công.
- Lớp User gửi kết quả lại cho lớp LoginView.
- Lớp LoginView gọi sang lớp ManagerHomeView.
- Lớp ManagerHomeView hiển thị cho nhân viên quản lí.
- Nhân viên click vào chức năng quản lí thông tin phòng.
- Lớp ManagerHomeView gọi lớp ManageRoomView.
- Lớp ManageRoomView hiển thị cho nhân viên quản lí.



Hình 7.12: Biểu đồ tuần tự phân tích chức năng sửa thông tin phòng.

- Nhân viên click vào chức năng sửa thông tin phòng.
- Lớp ManageRoomView gọi sang lớp SearchRoomView.
- Lớp SearchRoomView hiển thị cho nhân viên quản lí.
- Nhân viên nhập từ khóa và click tìm.
- Lớp SearchRoomView gọi lớp Room xử lí.
- Lớp Room gọi phương thức tìm phòng theo tên.

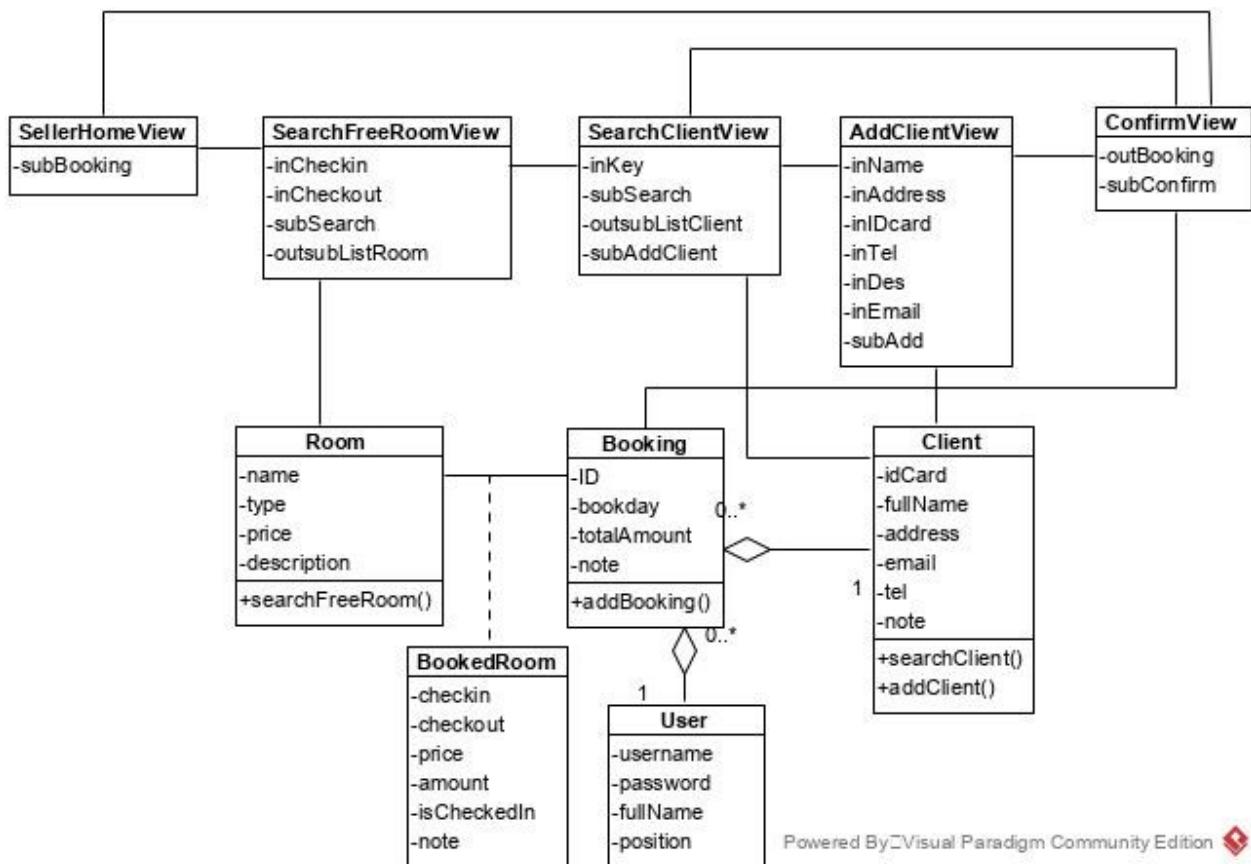
- Kết quả được lớp Room gửi lại cho lớp SearchRoomView.
- Lớp SearchRoomView hiển thị kết quả cho nhân viên quản lí.
- Nhân viên click một phòng để sửa.
- Lớp SearchRoomView gọi lớp EditRoomView.
- Lớp EditRoomView hiển thị cho nhân viên quản lí với các thông tin có sẵn của phòng.
- Nhân viên sửa một vài thuộc tính rồi click lưu.
- Lớp EditRoomView gọi lớp Room xử lí.
- Lớp Room gọi phương thức cập nhật thông tin phòng.
- Lớp Room trả lại cho lớp EditroomView
- Lớp EditRoomView thông báo thành công.
- Nhân viên click OK của thông báo.
- Lớp EditRoomView gọi lại lớp ManagerHomeView.
- Lớp ManagerHomeView hiển thị cho nhân viên quản lí.

### b. *Chức năng đặt phòng*

Phân tích chi tiết chức năng đặt phòng (bỏ qua giai đoạn đăng nhập) diễn ra như sau:

- Sau khi đăng nhập thành công, giao diện chính của nhân viên bán hàng hiện ra -> đề xuất lớp SellerHomeView, có ít nhất nút nhấn chức năng đặt phòng.
- Khi khách hàng gọi điện đến, nhân viên bán hàng click chức năng đặt phòng -> giao diện tìm phòng trống hiện ra -> đề xuất lớp SearchFreeRoomView, có các ô nhập ngày checkin, checkout, nút tìm, danh sách kết quả.
- Nhân viên nhập ngày checkin, checkout và click nút tìm sau khi hỏi yêu cầu khách hàng -> hệ thống tìm kiếm phòng trống trong khoảng thời gian trên -> cần một chức năng searchFreeRoom() -> chức năng này nên là hành động của đối tượng Room.
- Hệ thống tìm kiếm phòng trống xong sẽ trả lại lớp SearchFreeRoomView để hiển thị cho nhân viên bán hàng.
- Nhân viên báo lại với khách và click vào phòng tương ứng với lựa chọn của khách -> hệ thống chuyển sang giao diện tìm khách hàng -> đề xuất lớp SearchClientView, có ô nhập tên, nút tìm kiếm, và danh sách kết quả.

- Nhân viên bán hàng nhập tên khách hàng và click tìm kiếm sau khi hỏi thông tin khách hàng -> hệ thống tìm các khách hàng có tên vừa nhập -> cần một chức năng searchClient() -> chức năng này nên là hành động của đối tượng Client.
- Hệ thống tìm kiếm khách hàng và trả lại kết quả cho lớp SearchClientView để hiển thị cho nhân viên. Nhân viên click vào dòng chứa thông tin khách hàng đúng như khách hàng khai báo (nếu không có sẽ click chức năng thêm mới).
- Giao diện xác nhận thông tin đặt phòng hiện lên -> đề xuất lớp ConfirmView, có phần hiển thị thông tin đặt phòng, nút xác nhận và quay lại.
- Nhân viên click xác nhận sau khi nhận xác nhận từ khách hàng -> hệ thống lưu thông tin đặt phòng vào CSDL -> cần chức năng addBooking() -> chức năng này nên là hành động của đối tượng Booking.
- Hệ thống lưu xong báo lại lớp ConfirmView, lớp ConfirmView báo thành công cho nhân viên -> nhân viên click OK -> hệ thống quay về lớp SellerHomeView.

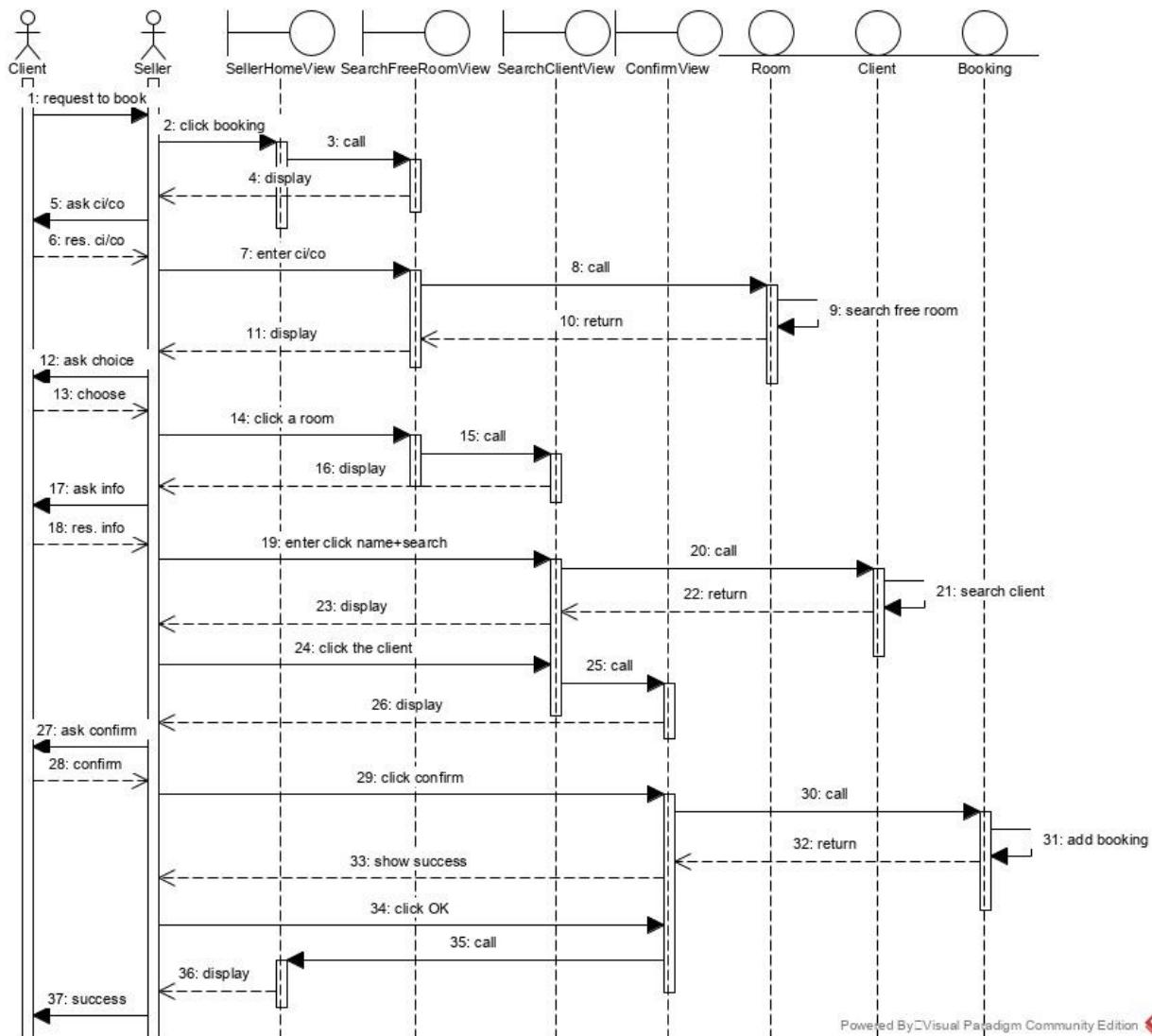


Hình 7.13: Biểu đồ lớp phân tích chức năng đặt phòng

Powered By Visual Paradigm Community Edition

Như vậy, kết quả thu được biểu đồ lớp cho chức năng đặt phòng như trong Hình 7.13. Với biểu đồ lớp này, kịch bản chi tiết cho chức năng đặt phòng diễn ra như sau (Hình 7.14):

- Nhân viên bán hàng click chức năng đặt phòng trên giao diện SellerHomeView sau khi nhận yêu cầu đặt phòng của khách hàng (qua điện thoại).
- Lớp SellerHomeView gọi lớp SearchFreeRoomView.
- Lớp SearchFreeRoomView hiển thị cho nhân viên.
- Nhân viên hỏi khách hàng ngày checkin, checkout.
- Khách hàng trả lời ngày checkin, checkout.



Hình 7.14: Biểu đồ tuần tự phân tích chức năng đặt phòng

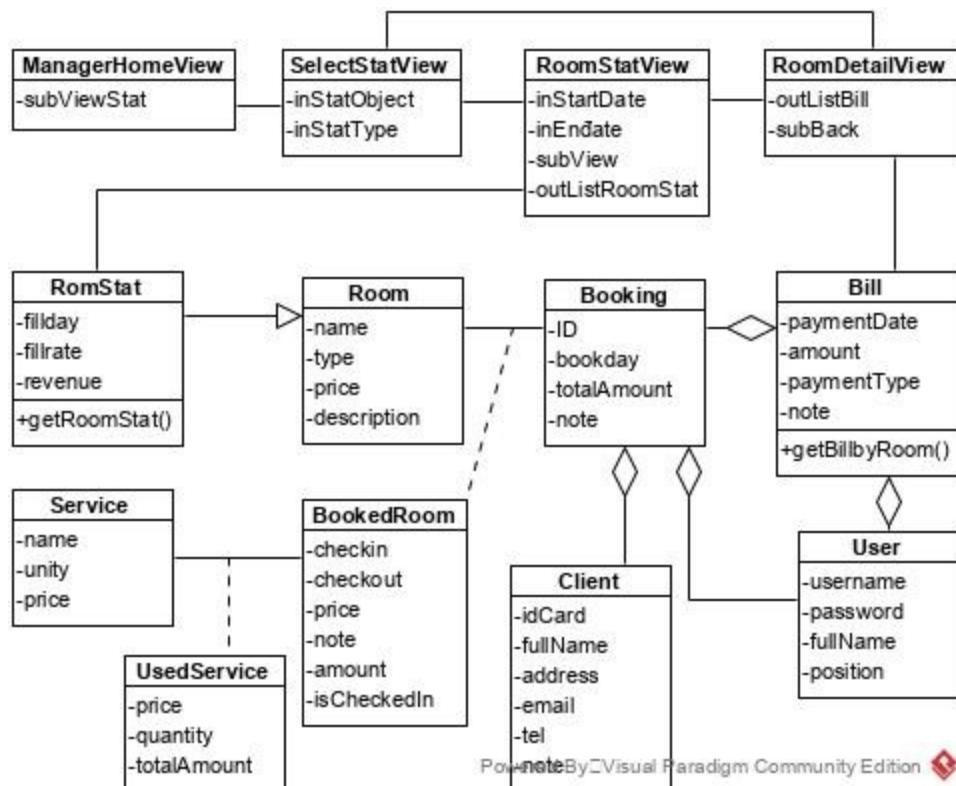
Powered By Visual Paradigm Community Edition

- Nhân viên nhập ngày checkin, checkout vào và click tìm.
- Lớp SearchFreeRoomView yêu cầu lớp Room xử lí.
- Lớp Room thực hiện tìm kiếm phòng trống.
- Tìm xong, lớp Room trả kết quả về cho lớp SearchFreeRoomView.
- Lớp SearchFreeRoomView hiển thị kết quả lên cho nhân viên.
- Nhân viên thông báo kết quả cho khách hàng và yêu cầu khách hàng chọn phòng.
- Khách hàng chọn 1 phòng theo yêu cầu.
- Nhân viên click vào phòng theo yêu cầu của khách hàng.
- Lớp SearchFreeRoomView gọi sang lớp SearchClientView hiển thị.
- Lớp SearchClientView hiển thị lên cho nhân viên.
- Nhân viên hỏi thông tin cá nhân khách hàng.
- Khách hàng cung cấp thông tin cho nhân viên.
- Nhân viên gõ tên và click tìm kiếm.
- Lớp SearchClientView gọi lớp Client xử lí.
- Lớp Client thực hiện chức năng tìm khách hàng theo tên.
- Lớp Client trả kết quả về cho lớp SearchClientView.
- Lớp SearchClientView hiển thị cho nhân viên.
- Nhân viên click vào dòng chứa thông tin khách hàng trùng với thông tin khách hàng đã khai báo.
- Lớp SearchClientView gọi sang lớp ConfirmView.
- Lớp ConfirmView hiển thị thông tin đặt phòng lên cho nhân viên
- Nhân viên yêu cầu khách hàng xác nhận lại toàn bộ thông tin.
- Khách hàng xác nhận OK.
- Nhân viên click nút xác nhận.
- Lớp ConfirmView gọi lớp Booking yêu cầu xử lí

- Lớp Booking thực hiện chức năng thêm thông tin đặt phòng vào CSDL.
- Lớp Booking trả lại thông tin cho lớp ConfirmView.
- Lớp ConfirmView hiện thông báo đặt phòng thành công.
- Nhân viên click vào nút OK của thông báo.
- Lớp ConfirmView gọi lại lớp SellerHomeView
- Lớp SellerHomeView hiển thị cho nhân viên
- Nhân viên báo đặt phòng thành công cho khách hàng.

### c. . Chức năng xem thống kê phòng theo doanh thu

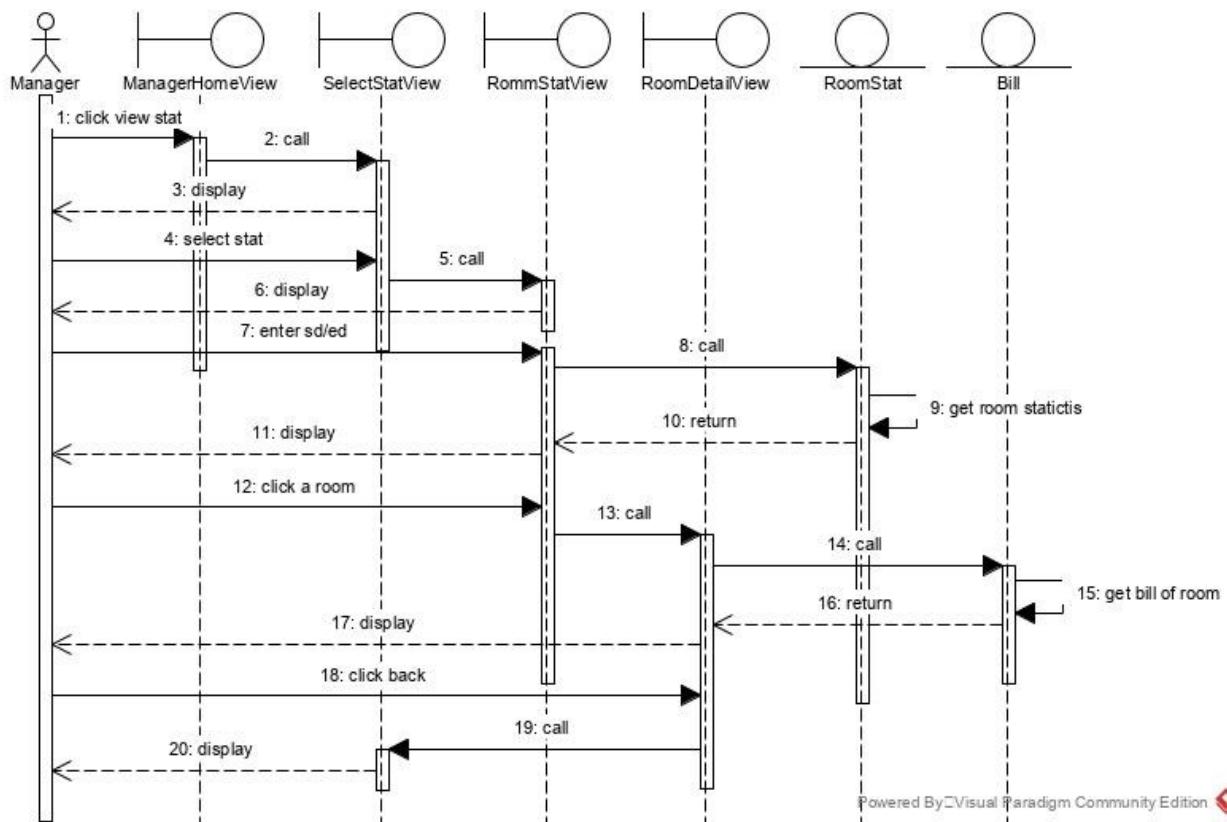
Phân tích chi tiết chức năng xem thống kê phòng theo doanh thu (bỏ qua giai đoạn đăng nhập) diễn ra như sau:



Hình 7.15: Biểu đồ lớp phân tích chức năng xem thống kê phòng theo doanh thu

- Sau khi nhân viên quản lý đăng nhập thành công, giao diện chính của quản lý hiện ra -> để xuất lớp ManagerHomeView, có ít nhất nút chọn chức năng xem báo cáo thống kê.

- Nhân viên click nút xem báo cáo thống kê -> giao diện chọn loại thống kê hiện ra -> để xuất lớp SelectStatView, có ô chọn đối tượng thống kê và thể loại thống kê.
- Nhân viên chọn đối tượng thống kê là phòng, thể loại là theo doanh thu -> giao diện thống kê phòng theo doanh thu hiện ra -> để xuất lớp RoomStatView, có ô nhập nhảy bắt đầu/kết thúc thống kê, nút xem, và bảng danh sách thống kê.
- Nhân viên nhập ngày bắt đầu/kết thúc thống kê và click xem -> hệ thống tìm kiếm thông tin thống kê -> cần chức năng getRoomStat() -> chức năng này nên là hành động của đối tượng RoomStat.
- Kết quả thống kê được hệ thống trả lại cho RoomStatView để hiển thị cho nhân viên.
- Nhân viên click vào một dòng để xem chi tiết 1 phòng -> hệ thống chuyển sang giao diện xem thống kê chi tiết 1 phòng -> để xuất lớp RoomDetailView, chứa thông tin phòng được chọn, và bảng danh sách thống kê các hóa đơn chi tiết.
- Để lớp RoomDetailView có dữ liệu để hiển thị, hệ thống cần thực hiện chức năng tìm danh sách các hóa đơn của một phòng trong khoảng thời gian thống kê getBillbyRoom() -> chức năng này nên là hành động của đối tượng Bill.



Hình 7.16: Biểu đồ tuần tự phân tích chức năng xem thống kê phòng theo doanh thu

Như vậy, kết quả thu được biểu đồ lớp cho chức năng xem thống kê phòng theo doanh thu như trong Hình 7.15. Với biểu đồ lớp này, kịch bản chi tiết cho chức năng xem thống kê phòng theo doanh thu diễn ra như sau (Hình 7.16):

- Sau khi đăng nhập thành công, nhân viên quản lý click vào chức năng xem thống kê.
- Lớp ManagerHomeView gọi lớp SelectStatView hiển thị.
- Lớp SelectStatView hiển thị cho nhân viên.
- Nhân viên chọn đối tượng thống kê là phòng, thể loại thống kê là theo doanh thu.
- Lớp SelectStatView gọi lớp RoomStatView hiển thị
- Lớp RoomStatView hiển thị.
- Nhân viên nhập ngày bắt đầu/kết thúc thống kê và click xem thống kê.
- Lớp RoomStatView gọi lớp RoomStat xử lí.
- Lớp RoomStat gọi chức năng lấy thống kê phòng theo doanh thu.
- Kết quả được lớp RoomStat gửi lại cho lớp RoomStatView.
- Lớp RoomStatView hiển thị kết quả thống kê cho nhân viên.
- Nhân viên click vào một dòng để xem chi tiết một phòng.
- Lớp RoomStatView gọi lớp RoomDetailView.
- Lớp RoomDetailView yêu cầu lớp Bill lấy thông tin.
- Lớp Bill gọi chức năng lấy thông tin hóa đơn của phòng trong khoảng thời gian thống kê.
- Kết quả được lớp Bill trả về cho lớp RoomDetailView
- Lớp RoomDetailView hiển thị kết quả thống kê chi tiết cho nhân viên.
- Nhân viên xem xong click quay về.
- Lớp RoomDetailView gọi lớp SelectStatView
- Lớp SelectStatView lại hiển thị cho nhân viên quản lý.

## CHƯƠNG 8: PHA THIẾT KẾ

### 8.1 TỔNG QUAN VỀ PHA THIẾT KẾ

#### 8.1.1 Dữ liệu và các hành động

- Hai khía cạnh của một sản phẩm phần mềm
  - Những hành động thực hiện trên dữ liệu
  - Dữ liệu mà các hành động thao tác trên dữ liệu đó
- Hai cách cơ bản của thiết kế hệ thống phần mềm
  - Thiết kế hướng hành động
  - Thiết kế hướng dữ liệu
- Cách thứ ba
  - Các phương pháp lai
  - Chẳng hạn, thiết kế hướng đối tượng

#### 8.1.2 Thiết kế và trừu tượng

- Các hoạt động thiết kế cổ điển
  - Thiết kế kiến trúc
  - Thiết kế chi tiết
  - Kiểm thử thiết kế
- Thiết kế kiến trúc
  - Đầu vào: Những đặc tả
  - Đầu ra: Sự phân nhỏ thành các mô đun
- Thiết kế chi tiết
  - Mỗi mô đun được thiết kế
    - Các thuật toán đặc trưng, các cấu trúc dữ liệu

### 8.1.3 Thiết kế

- Tổng kết luồng công việc thiết kế:
  - Các tài liệu luồng công việc thiết kế được lặp và tích hợp đến khi người lập trình có thể sử dụng được chúng
- Các quyết định bao gồm:
  - Ngôn ngữ lập trình
  - Sử dụng lại
  - Tính khả chuyên
- Ý tưởng của việc phân tách luồng công việc lớp thành những luồng công việc nhỏ độc lập (các gói) được thực hiện ở luồng công việc phân tích
- Mục tiêu là chia nhỏ luồng công việc cài đặt thành những phần có thể quản lý được
  - *Các hệ thống con*
- Tại sao phần mềm được chia nhỏ thành các hệ thống con:
  - Dễ dàng để cài đặt một số hệ thống con hơn là một hệ thống lớn
  - Nếu các hệ thống con độc lập với nhau thì chúng có thể được cài đặt bởi các đội lập trình khác nhau cùng một lúc
    - Khi đó toàn bộ sản phẩm phần mềm được chuyển giao sớm
- Kiến trúc của sản phẩm phần mềm bao gồm:
  - Các thành phần khác nhau
  - Cách chúng ăn khớp với nhau
  - Phân chia các thành phần vào các hệ thống con
- Công việc của thiết kế kiến trúc được chuyên môn hóa
  - Nó được thực hiện bởi các kiến trúc sư phần mềm
- Kiến trúc sư (architect)cần có sự cân bằng về:
  - Mỗi sản phẩm phần mềm phải đáp ứng các yêu cầu chức năng của chúng (các use case)

- Mỗi sản phẩm phần mềm cũng phải đáp ứng các yêu cầu phi chức năng, bao gồm:
  - Khả chuyền, đáng tin, mạnh mẽ, bảo trì và bảo mật
- Mỗi sản phẩm phần mềm phải thực hiện tất cả những yêu cầu này trong ràng buộc chi phí và thời gian
- Kiến trúc sư phải giúp khác hàng bằng cách sắp xếp những cân bằng này.
- Thường không thể đáp ứng tất cả các yêu cầu chức năng và phi chức năng trong ràng buộc về chi phí và thời gian
  - Có một vài sự sắp xếp được thực hiện
- Khách hàng phải
  - Giảm bớt một số yêu cầu;
  - Tăng chi phí; và /hoặc
  - Thay đổi thời gian chuyền giao
- Kiến trúc của sản phẩm phần mềm là quan trọng
  - Luồng công việc xác định yêu cầu có thể được sửa lại (fix) trong suốt luồng phân tích
  - Luồng công việc phân tích có thể được sửa lại trong suốt luồng công việc thiết kế
  - Luồng công việc thiết kế có thể được sửa lại trong suốt luồng công việc cài đặt
- Nhưng không có cách để phục hồi từ kiến trúc gần tốt nhất
  - Kiến trúc phải được thiết kế lại ngay lập tức

#### 8.1.4 Kiểm thử trong pha thiết kế

- Rà soát thiết kế phải được thực hiện
  - Thiết kế phải phản ánh chính xác đặc tả
  - Chính thiết kế phải chính xác
- Kiểm tra kỹ lưỡng hướng giao tác (Transaction-driven inspections)
  - Cần thiết cho các phần mềm hướng giao tác

- Tuy nhiên, những kiểm tra hướng giao tác là chưa đủ nên những kiểm tra hướng đặc tả cũng được yêu cầu

#### 8.1.5 Kỹ thuật hình thức cho thiết kế chi tiết

- Việc cài đặt một phần mềm hoàn thiện và sau đó chứng minh nó là chính xác là rất khó
- Tuy nhiên, sử dụng kỹ thuật hình thức trong suốt thiết kế chi tiết có thể giúp:
  - Việc chứng minh tính chính xác có thể được áp dụng đối với các phần mô đun
  - Thiết kế có ít lỗi hơn nếu nó được phát triển song song với sự kiểm chứng tính chính xác
  - Nếu cùng một người lập trình thực hiện cả thiết kế chi tiết và cài đặt
    - Người lập trình sẽ có một quan điểm tích cực đối với thiết kế chi tiết
    - Chính điều này dẫn đến ít lỗi

#### 8.1.6 Kỹ thuật thiết kế hệ thống thời gian thực

- Những khó khăn của hệ thống thời gian thực
  - Đầu vào lấy từ thế giới thực
    - Phần mềm không có điều khiển bộ định thời của các đầu vào  
( Software has no control over the timing of the inputs)
  - Được cài đặt thường xuyên trên phần mềm phân tán
    - Communications implications
    - Những vấn đề định thời (Timing issues)
  - Những vấn đề của đồng bộ
    - Race conditions
    - Bế tắc (Deadlock - deadly embrace)
- Những khó khăn chính trong thiết kế hệ thống thời gian thực
  - Xác định liệu những ràng buộc về thời gian có được đáp ứng bởi thiết kế không?
- Hầu hết các phương thức thiết kế thời gian thực là những sự mở rộng của các phương thức phi thời gian thực (Most real-time design methods are extensions of non-real-time methods to real-time )

- Chúng ta đã hạn chế những thử nghiệm trong cách sử dụng bất cứ phương thức thời gian thực nào. (We have limited experience in the use of any real-time methods)
- The state-of-the-art is not where we would like it to be

### 8.1.7 Công cụ CASE cho thiết kế

- Rất quan trọng để kiểm tra tài liệu thiết kế hợp nhất với mọi khía cạnh của phân tích
  - Để xử lý tài liệu phân tích và thiết kế chúng ta cần một công cụ upperCASE
- Công cụ upperCASE
  - Được xây dựng xung quanh từ điển dữ liệu(Are built around a data dictionary)
  - They incorporate a consistency checker, and
  - Màn ảnh và các bản tường trình (Screen and report generators)
  - Công cụ quản lý đối khi bao gồm
    - Uớc lượng
    - Lập kế hoạch
- Ví dụ của các công cụ cho thiết kế hướng đối tượng
  - Công cụ thương mại
    - Software through Pictures
    - IBM Rational Rose
    - Together
  - Công cụ mã nguồn mở
    - ArgoUML

### 8.1.8 Thước đo cho thiết kế

- Thước đo chất lượng thiết kế
  - Kết dính (Cohesion)
  - Sự kết nối (Coupling)
  - Thống kê lỗi

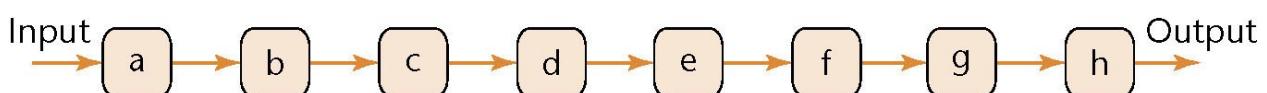
- Cyclomatic complexity is problematic
  - Độ phức tạp dữ liệu được bỏ qua
  - Nó không được sử dụng nhiều với mô hình hướng đối tượng
- Các thước đo được đề xuất cho mô hình hướng đối tượng
  - Chúng đã chứng tỏ được tính hữu ích trong cả lý thuyết và thử nghiệm.(They have been challenged on both theoretical and experimental grounds)

### 8.1.9 Những thách thức của pha thiết kế

- Đội thiết kế không nên làm quá nhiều
  - Thiết kế chi tiết không nên là những người viết mã
- Đội thiết kế không nên làm quá ít
  - Đầu để cho đội thiết kế để đưa ra một thiết kế chi tiết hoàn thiện
- Chúng ta cần “phát triển” những người thiết kế giỏi
- Những người thiết kế giỏi tiềm năng phải
  - Được nhận định,
  - Đã được đào tạo một cách hình thức,
  - Đang học nghề để trở thành một người thiết kế giỏi, và
  - Được phép giao tiếp với những người thiết kế khác
- Phải có một hướng đi nghề nghiệp cụ thể cho những người thiết kế này, với một chế độ thường thích hợp

## 8.2 THIẾT KẾ HƯỚNG HÀNH ĐỘNG

- Phân tích luồng dữ liệu
  - Sử dụng phân tích luồng dữ liệu với hầu hết các phương pháp đặc tả (ở đây là phân tích hệ thống theo hướng cấu trúc)
- Điểm chính: Chúng ta đã chi tiết hóa các thông tin hành động từ DFD

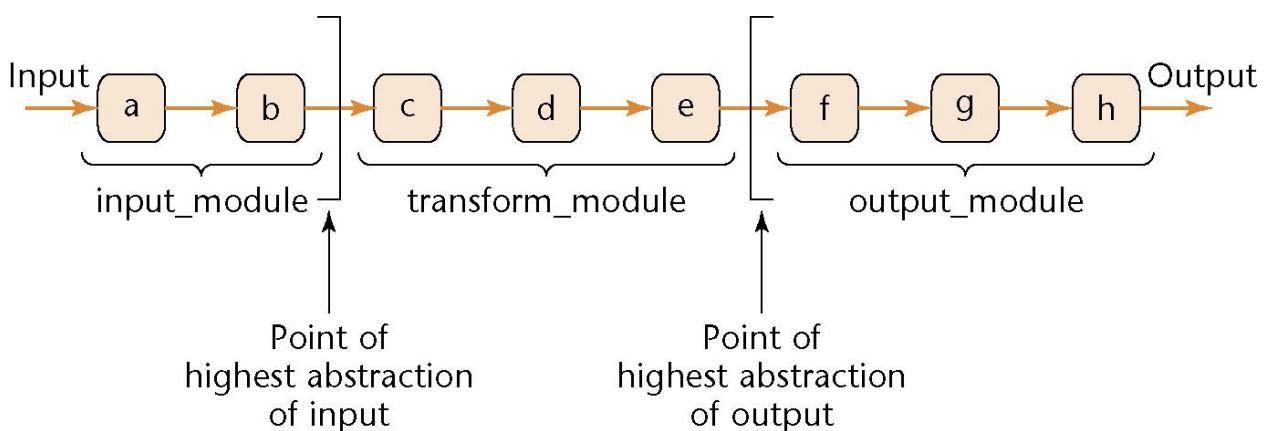


Hình 8.1: Luồng dữ liệu

### 8.3 PHÂN TÍCH VÀ THIẾT KẾ DÒNG DỮ LIỆU

#### 8.3.1 Phân tích dòng dữ liệu

- Mỗi sản phẩm phần mềm biến đổi từ đầu vào thành đầu ra
- Xác định
  - “Điểm trừu tượng nhất của đầu vào”
  - “Điểm trừu tượng nhất của đầu ra”

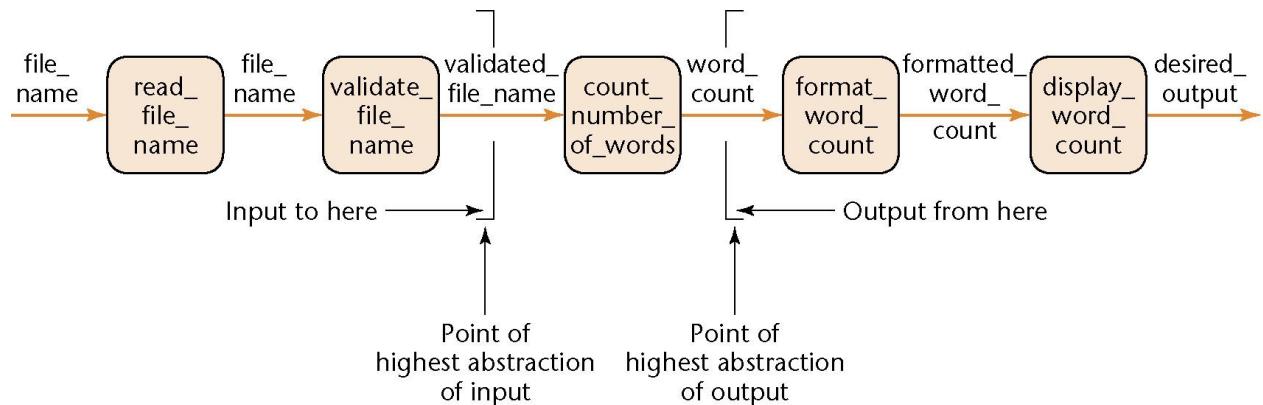


Hình 8.2: Tìm các điểm trừu tượng đầu vào, đầu ra

- Phân chia phần mềm thành 3 mô đun
- Lặp lại các bước cho đến khi mỗi mô đun có độ dính kết cao (cohesion)
  - Những chỉnh sửa phụ có thể cần thiết để giảm bớt độ kết nối (coupling)

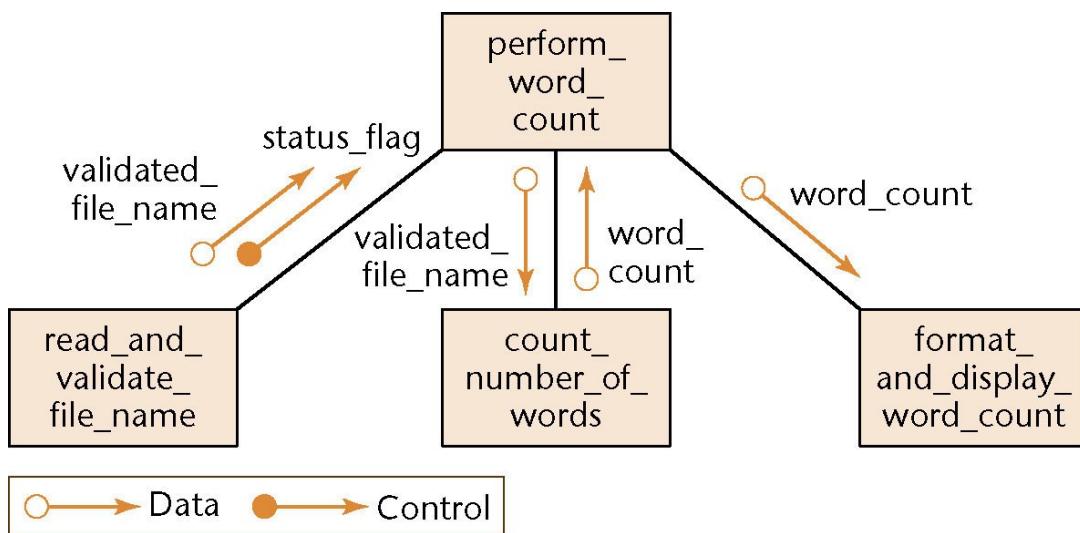
#### 8.3.1.1 Bài toán đếm từ

- Ví dụ: Thiết kế một hệ thống phần mềm với đầu vào là một tên tệp, và kết quả trả về là số lượng từ trong file đó( giống như UNIX wc )



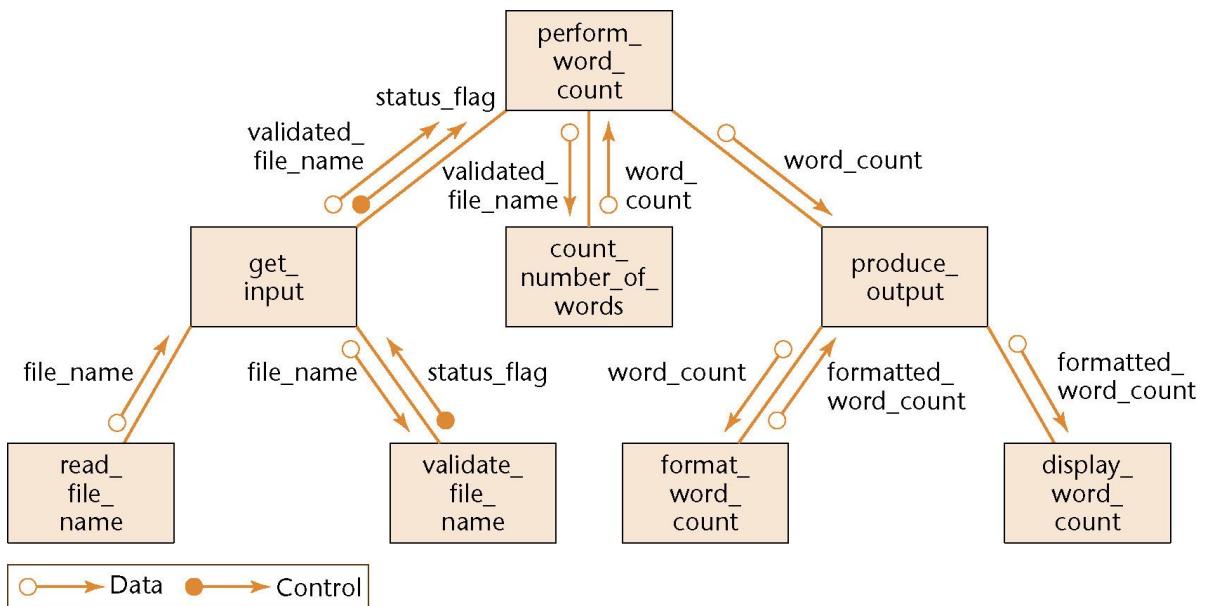
Hình 8.3: Xác định điểm trùu tượng vào/ra cho bài toán đếm từ

- Quá trình làm mịn thứ nhất



Hình 8.4: Quá trình làm mịn thứ nhất

- Làm mịn hai mô đun ở mức dính kết giao tiếp (communicational cohesion)
- Bước làm mịn thứ hai



Hình 8.5: Quá trình làm mịn thứ hai

- Tất cả 8 mô đun đều ở mức dính kết chức năng (functional cohesion)
- Thiết kế kiến trúc đã hoàn thiện, vì thế thực hiện thiết kế chi tiết.
- Hai định dạng để biểu diễn thiết kế chi tiết:
  - Tabular

Module name	<b>read_file_name</b>
Module type	Function
Return type	<b>string</b>
Input arguments	None
Output arguments	None
Error messages	None
Files accessed	None
Files changed	None
Modules called	None
Narrative	<p>The product is invoked by the user by means of the command string</p> <p style="text-align: center;"><b>word_count &lt;file_name&gt;</b></p> <p>Using an operating system call, this module accesses the contents of the command string input by the user, extracts <b>&lt;file_name&gt;</b>, and returns it as the value of the module.</p>

Module name	<b>validate_file_name</b>
Module type	Function
Return type	<b>Boolean</b>
Input arguments	<b>file_name : string</b>
Output arguments	None
Error messages	None
Files accessed	None
Files changed	None
Modules called	None
Narrative	This module makes an operating system call to determine whether file <b>file_name</b> exists. The module returns <b>true</b> if the file exists and <b>false</b> otherwise.
Module name	<b>produce_output</b>
Module type	Function
Return type	<b>void</b>
Input arguments	<b>word_count : integer</b>
Output arguments	None
Error messages	None
Files accessed	None
Files changed	None
Modules called	<b>format_word_count</b> arguments: <b>word_count : integer</b> <b>formatted_word_count : string</b> <b>display_word_count</b> arguments: <b>formatted_word_count : string</b>
Narrative	This module takes the integer <b>word_count</b> passed to it by the calling module and calls <b>format_word_count</b> to have that integer formatted according to the specifications. Then it calls <b>display_word_count</b> to have the line printed.

- o Mã giả (PDL — program design language – ngôn ngữ thiết kế chương trình)

```

void perform_word_count()
{
    String validate_file_name;
    int word_count;

    if (get_input (validate_file_name) is false)
        print "error 1: file does not exist";
    else
    {
        set word_count equal to count_number_of_words (validate_file_name);
        if (word_count is equal to -1)
            print "error 2: file is not a text file";
        else
            produce_output (word_count);
    }
}

Boolean get_input (String validate_file_name)
{
    String file_name;

    file_name = read_file_name ();
    if (validate_file_name (file_name) is true)
    {
        set validate_file_name equal to file_name;
        return true;
    }
    else
        return false;
}

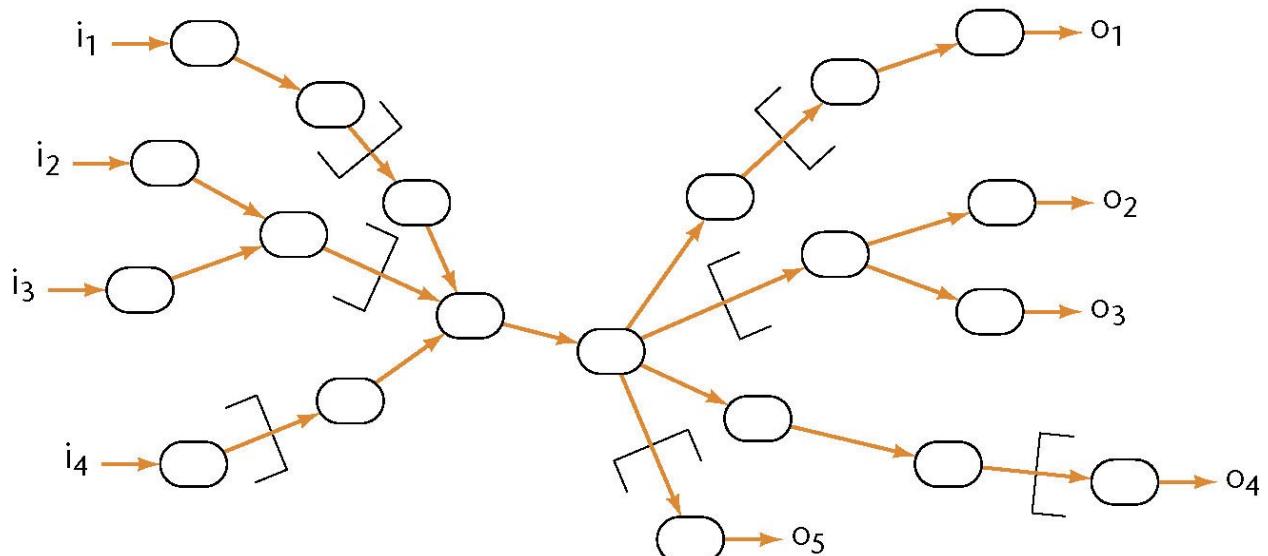
void display_word_count (String formatted_word_count)
{
    print formatted_word_count, left justified;
}

String format_word_count (int word_count);
{
    return "File contains" word_count "words";
}

```

### 8.3.1.2 Mở rộng phân tích luồng dữ liệu

- Sản phẩm phần mềm trong thực tế, có
  - Nhiều hơn một luồng đầu vào
  - Nhiều hơn một luồng đầu ra
- Tìm điểm trùu tượng nhất của mỗi luồng



Hình 8.6: mở rộng luồng phân tích dữ liệu

- Tiếp tục thực hiện cho đến khi mỗi mô đun có độ dính kết cao
  - Điều chỉnh kết nối (coupling) nếu cần thiết

### 8.3.2 Thiết kế dòng dữ liệu

- Nguyên lý cơ bản
  - Cấu trúc của một phần mềm phải phù hợp với cấu trúc của dữ liệu của nó
- Có ba phương pháp rất giống nhau
  - Michael Jackson [1975], Warnier [1976], Orr [1981]
  - Thiết kế hướng dữ liệu
  - Chưa từng phổ biến như thiết kế hướng hành động
  - Với sự ra đời của OOD, thiết kế hướng dữ liệu đã bị lỗi thời

## 8.4 THIẾT KẾ HƯỚNG ĐỐI TƯỢNG

- Mục đích
  - Thiết kế phần mềm dưới dạng các lớp được trích rút trong suốt phân tích hướng đối tượng (OOA)
- Nếu chúng ta đang sử dụng một ngôn ngữ mà không có tính kế thừa như C, Ada83...
  - Sử dụng thiết kế kiểu dữ liệu trừu tượng

- Nếu chúng ta đang sử dụng ngôn ngữ không có khai báo kiểu như FOTRAN, COBAL...
  - Sử dụng đóng gói dữ liệu

OOD gồm 2 bước:

**Bước 1.** Hoàn thiện biểu đồ lớp

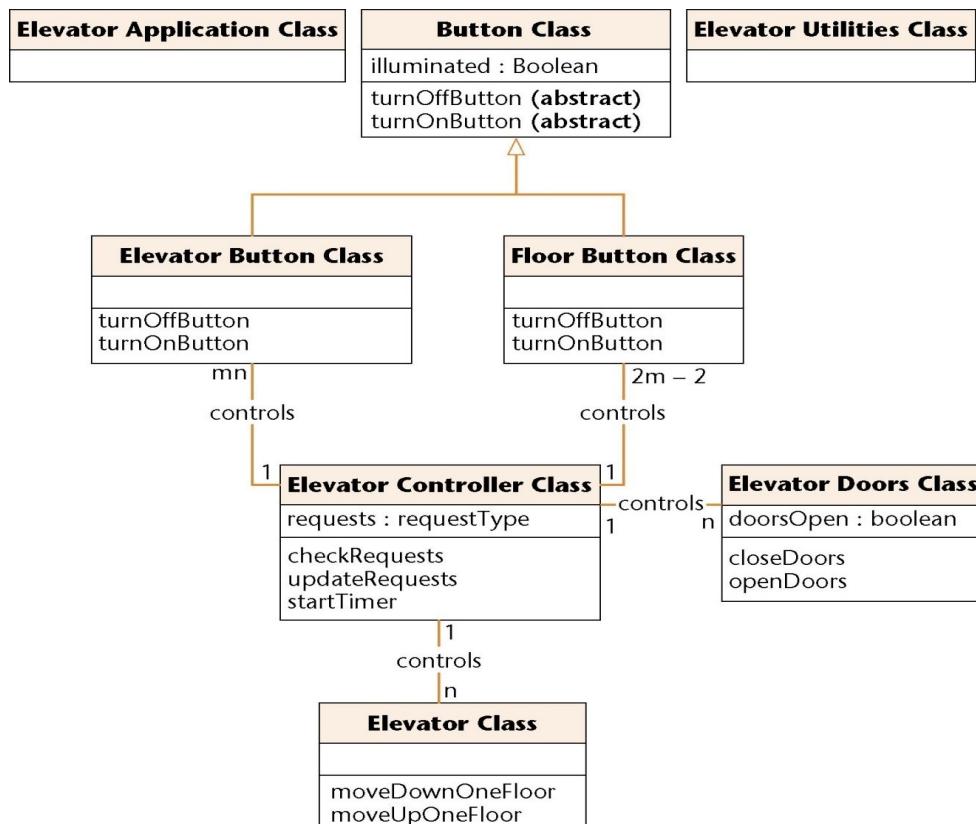
- Xác định định dạng của các thuộc tính : Định dạng của các thuộc tính có thể được rút ra từ tài liệu phân tích .Ví dụ: Ngày theo định dạng của Mỹ (mm/mm/yyyy) và định dạng của Châu Âu (dd/mm/yyyy). Trong cả hai trường hợp đều yêu cầu 10 ký tự. Định dạng có thể được thêm vào trong suốt quá trình phân tích. Để cực tiểu hóa việc làm lại, không bao giờ thêm một mục vào biểu đồ UML cho đến lúc cần thiết
- Gán mỗi phương thức cho lớp hoặc một client đã gửi thông điệp tới đối tượng của lớp đó. Với 3 nguyên lý: Nguyên lý A: Ân giấu thông tin. Nguyên lý B: Nếu một phương thức được gọi bởi nhiều client của một đối tượng thì sẽ gán phương thức đó cho đối tượng chứ không phải các client. Nguyên lý C: thiết kế hướng trách nhiệm
- Xem xét vòng lặp thứ hai của thẻ CRC của điều khiển thang máy

CLASS
<b>Elevator Controller Class</b>
RESPONSIBILITY
1. Send message to <b>Elevator Button Class</b> to turn on button 2. Send message to <b>Elevator Button Class</b> to turn off button 3. Send message to <b>Floor Button Class</b> to turn on button 4. Send message to <b>Floor Button Class</b> to turn off button 5. Send message to <b>Elevator Class</b> to move up one floor 6. Send message to <b>Elevator Class</b> to move down one floor 7. Send message to <b>Elevator Doors Class</b> to open 8. Start timer 9. Send message to <b>Elevator Doors Class</b> to close after timeout 10. Check requests 11. Update requests
COLLABORATION
1. <b>Elevator Button Class</b> (subclass) 2. <b>Floor Button Class</b> (subclass) 3. <b>Elevator Doors Class</b> 4. <b>Elevator Class</b>

Hình 8.7: Xác định trách nhiệm từ thẻ CRC

- Trách nhiệm sau được gán trách nhiệm cho lớp điều khiển thang máy
  - 8. Bắt đầu đặt thời gian
  - 10. Kiểm tra yêu cầu
  - 11. Cập nhật yêu cầu
- Bởi vì chúng được thực hiện bởi lớp điều khiển thang máy

- Tám trách nhiệm còn lại có cùng một dạng:
  - “Gửi một thông điệp tới một lớp khác để yêu cầu nó làm một cái gì đó”
- Những trách nhiệm này được gán cho các lớp khác
  - Thiết kế hướng trách nhiệm
  - Xem xét độ an toàn
- Các phương thức *mở cửa, đóng cửa* được gán cho lớp **Cửa thang máy**
- Các phương thức *mở nút, tắt nút* được gán cho lớp **Nút Tầng** và lớp **Thang Máy**
- Biểu đồ lớp chi tiết cho bài toán thang máy:



Hình 8.8: Sơ đồ lớp chi tiết cho bài toán thang máy

## Bước 2. Thực hiện thiết kế chi tiết

- Thiết kế chi tiết của vòng lặp sự kiện thang máy được đưa ra từ biểu đồ trạng thái

```

void elevatorEventLoop (void)
{
    while (TRUE)
    {
        if (a button has been pressed)
            if (button is not on)
            {
                updateRequests;
                button::turnOnButton;
            }
        else if (elevator is moving up)
        {
            if (there is no request to stop at floor f)
                elevator::moveUpOneFloor;
            else
            {
                stop elevator by not sending a message to move;
                elevatorDoors::openDoors;
                startTimer;
                if (elevatorButton is on)
                    elevatorButton::turnOffButton;
                updateRequests;
            }
        }
        else if (elevator is moving down)
            [similar to up case]
        else if (elevator is stopped and request is pending)
        {
            elevatorDoors::closeDoors;
            determine direction of next request;
            if (appropriate floorButton is on)
                floorButton::turnOffButton;
            elevator::moveUp/DownOneFloor;
        }
        else if (elevator is at rest and not (request is pending))
            elevatorDoors::closeDoors;
        else
            there are no requests, elevator is stopped with elevatorDoors closed, so do nothing;
    }
}

```

## 8.5 CASE STUDY CHO PHA THIẾT KẾ

Mục này tiếp tục tiến hành thiết kế hệ thống cho ứng dụng quản lý đặt phòng khách sạn đã được trình bày trong case study của các pha lấy yêu cầu và pha phân tích. Nội dung bao gồm:

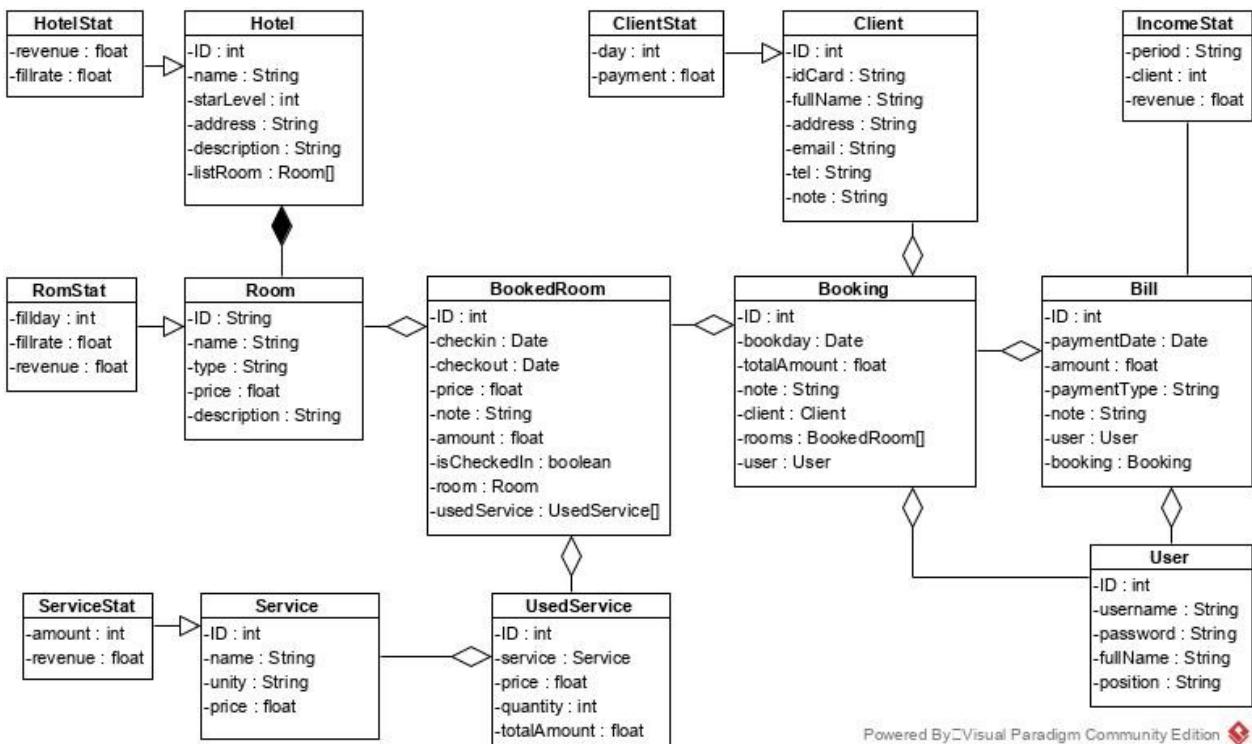
- Thiết kế lớp thực thể
- Thiết kế cơ sở dữ liệu
- Thiết kế chi tiết các modul.

### 8.5.1 Thiết kế lớp thực thể

Sử dụng biểu đồ lớp thực thể từ pha phân tích, các bước tiến hành như sau:

- Bổ sung kiểu dữ liệu cho các thuộc tính đã có

- Bổ sung thuộc tính id cho các lớp không kế thừa từ lớp khác.
- Chuyển đổi các quan hệ association thành nhiều quan hệ thành phần. Quan hệ Room + Booking -> BookedRoom chuyển thành quan hệ: Room là thành phần của BookedRoom, BookedRoom là thành phần của Booking. Quan hệ BookedRoom + Service -> UsedService chuyển thành quan hệ: Service là thành phần của UsedService, UsedService là thành phần của BookedRoom.
- Bổ sung các thuộc tính tương ứng với quan hệ thành phần.

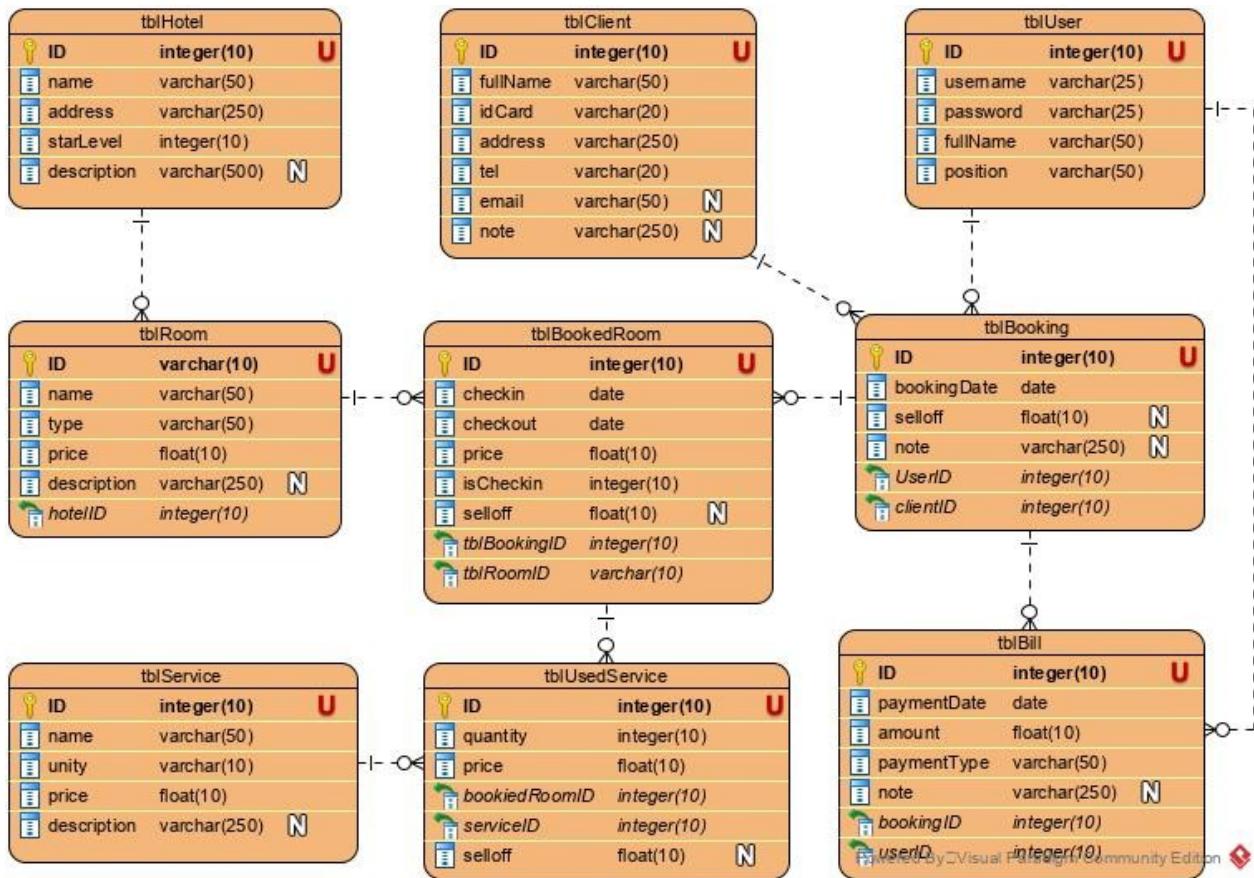


Hình 8. 9: Biểu đồ thiết kế lớp thực thể toàn hệ thống

### 8.5.2 Thiết kế cơ sở dữ liệu

Dựa vào sơ đồ lớp thực thể đã trích được trong pha phân tích (Hình 8.10), chúng ta có thể đề xuất các bảng dữ liệu như sau:

- tblHotel: lưu các thông tin về khách sạn, bao gồm: id, tên, địa chỉ, hạng sao, và mô tả về khách sạn.
- tblRoom: lưu các thông tin về phòng khách sạn, bao gồm: id (cũng là số hiệu phòng), kiểu phòng, giá hiển thị, và mô tả chung về phòng.



Hình 8.10: Biểu đồ thiết kế CSDL của hệ thống

- **tblClient**: lưu thông tin các khách hàng đặt phòng, bao gồm: id, số thẻ căn cước, kiểu thẻ căn cước, họ tên đầy đủ, ngày sinh, và địa chỉ.
- **tblBooking**: lưu thông tin về từng lần đặt chỗ của khách hàng, bao gồm: ngày đặt, khuyến mãi nếu có
- **tblBookedRoom**: lưu thông tin từng phòng đặt: ngày đến, ngày đi, giá đặt, trạng thái đã checkin hay chưa, khuyến mãi nếu có.
- **tblUser**: lưu thông tin về người dùng của phần mềm: id, tên đăng nhập, mật khẩu, và vị trí công việc.
- **tblService**: lưu thông tin các dịch vụ kèm theo phòng và có tính phí cho khách hàng, mỗi dịch vụ được lưu bởi: id, tên dịch vụ, đơn vị tính, đơn giá hiển thị.
- **tblUsedService**: lưu thông tin các dịch vụ đã được sử dụng bởi khách đặt phòng, bao gồm: id, số lượng theo đơn vị tính, đơn giá thực trả.

- tblBill: lưu thông tin hóa đơn mỗi lần thanh toán của khách hàng, bao gồm: id, ngày trả, số tiền thanh toán cho lần trả đó, hình thức thanh toán.

Quan hệ giữa các bảng được mô tả như Hình 8.10:

- Bảng tblHotel quan hệ 1-n với bảng tblRoom
- Bảng tblRoom quan hệ 1-n với bảng tblBookedRoom
- Bảng tblClient quan hệ 1-n với bảng tblBooking
- Bảng tblBooking quan hệ 1-n với bảng tblBookedRoom
- Bảng tblService và bảng tblBookedRoom đều quan hệ 1-n với bảng tblUsedService
- Bảng tblUser đều quan hệ 1-n với bảng tblBooking và tblBill.
- Bảng tblBooking quan hệ 1-n với bảng tblBill.

### 8.5.3 Thiết kế chi tiết các modul

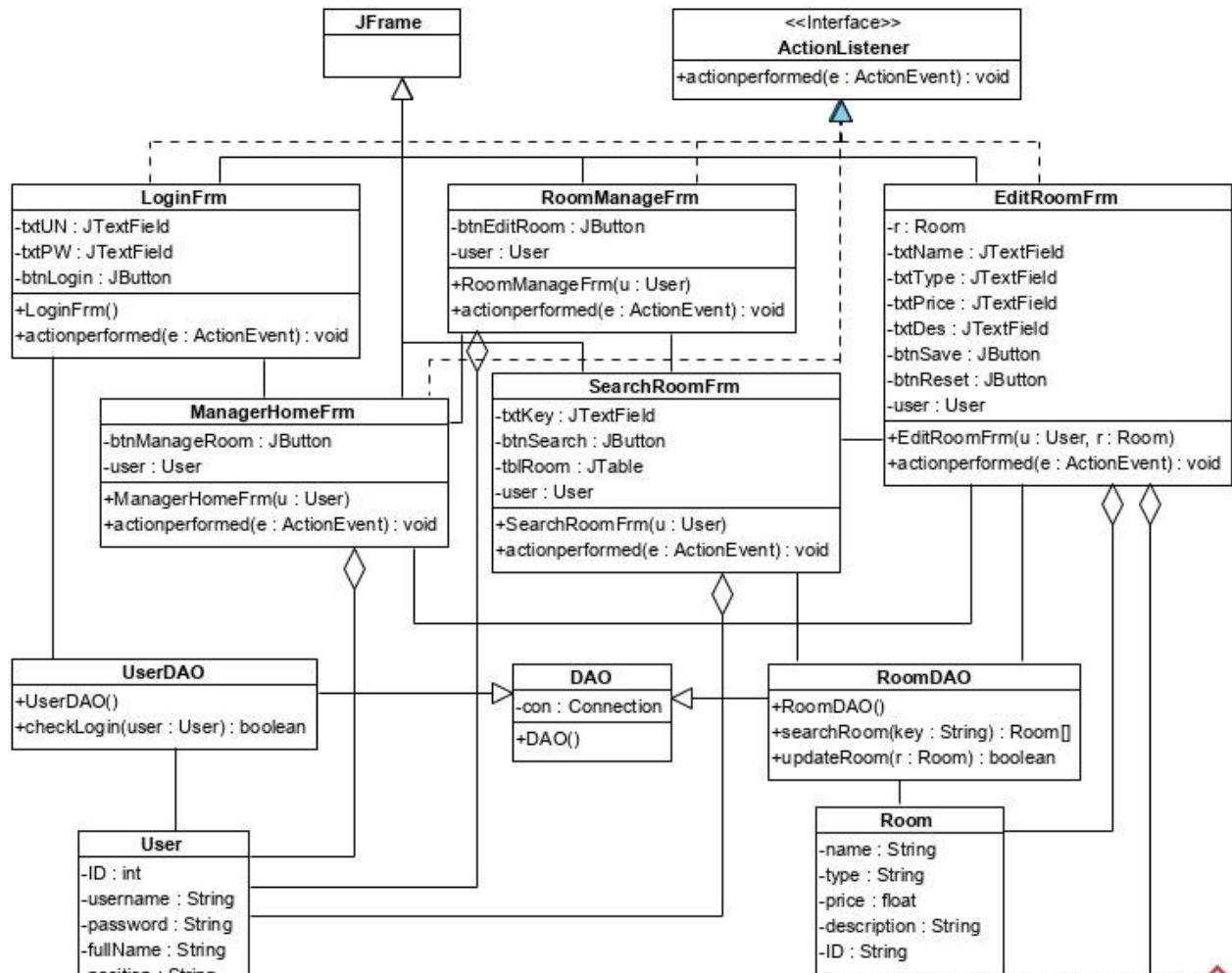
Nội dung phần này tiếp tục trình bày các bước thiết kế chi tiết cho ba modul đã được chọn minh họa trong chương phân tích: sửa thông tin phòng, đặt phòng, xem thông kê phòng. Trong đó, modul đặt phòng và xem thông kê phòng được loại bỏ giai đoạn nhân viên đăng nhập, vì lí do trùng lặp với các bước đăng nhập đã được thiết kế chi tiết trong chức năng sửa thông tin phòng.

#### a. Thiết kế chi tiết cho chức năng sửa thông tin phòng.

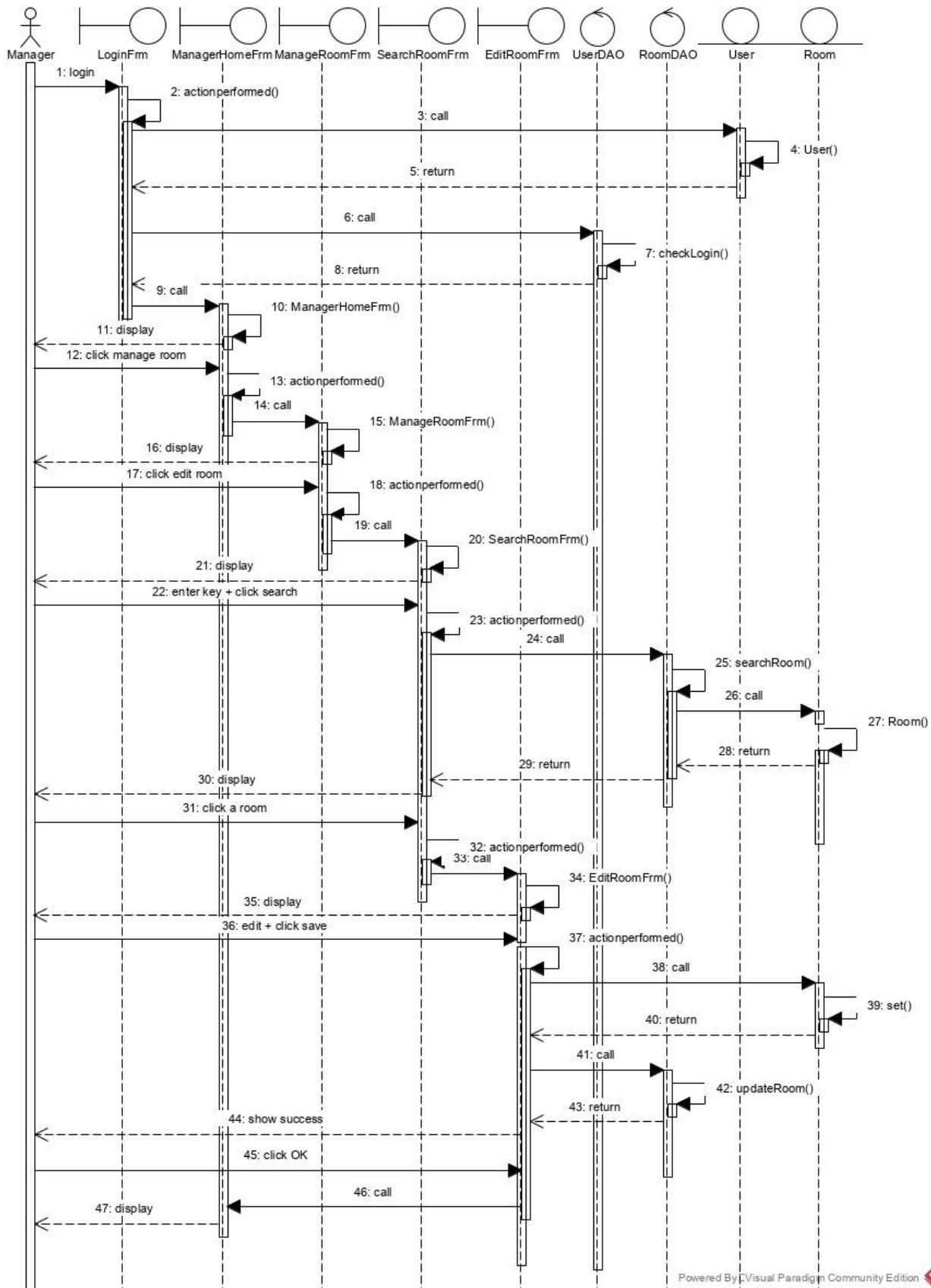
Ở bước thiết kế tĩnh, chúng ta cần các lớp theo mô hình MVC cho modul này như sau (Hình 8.11):

- Các lớp tầng giao diện: LoginFrm là giao diện đăng nhập. ManagerHomeFrm là giao diện chính của nhân viên quản lý. ManageRoomFrm là giao diện quản lý thông tin phòng. SearchRoomFrm là giao diện tìm phòng để sửa. EditRoomFrm là giao diện sửa một phòng đã chọn.
- Các lớp tầng điều khiển: UserDAO là lớp truy cập dữ liệu xử lý thông tin liên quan đến thành viên hệ thống. RoomDAO là lớp truy cập dữ liệu xử lý thông tin liên quan đến phòng. Hai lớp này đều kế thừa lớp DAO để xử lý cơ chế dùng chung truy cập vào cơ sở dữ liệu.
- Các lớp tầng thực thể liên quan: User, Room.

Đối với thiết kế động, các bước thực hiện diễn ra trong modul này như sau (Hình 8.12):



- Nhân viên quản lí nhập tên đăng nhập, mật khẩu và click đăng nhập trên giao diện **LoginFrm**.
- Hàm `actionPerformed()` của lớp **LoginFrm** được gọi.
- Hàm `actionPerformed()` gọi lớp **User** để đóng gói thông tin đăng nhập.
- Lớp **User** đóng gói thông tin vào thực thể **User**
- Lớp **User** trả đối tượng **User** về cho phương thức `actionPerformed()`.
- Phương thức `actionPerformed()` gọi phương thức `checkLogin()` của lớp **UserDAO**.
- Phương thức `checkLogin()` kiểm tra thông tin đăng nhập.
- Phương thức `checkLogin()` gọi lớp **User** để đóng gói bổ sung thuộc tính `name`, `position`.



Hình 8.12: Biểu đồ tuần tự thiết kế của chức năng sửa thông tin phòng

9. Lớp User gọi các hàm setName(), setPosition() để đóng gói bổ sung.
10. Lớp User trả kết quả về cho phương thức checkLogin().
11. Phương thức checkLogin() trả kết quả về cho phương thức actionPerformed().
12. Phương thức actionPerformed() gọi lớp ManagerHomeFrm.
13. Phương thức ManagerHomeFrm() được gọi
14. Giao diện tương ứng hiển thị.
15. Nhân viên click chọn chức năng quản lý thông tin phòng.
16. Phương thức actionPerformed() được kích hoạt.
17. Phương thức actionPerformed() gọi lớp RoomManageFrm.
18. Phương thức RoomManageFrm() được kích hoạt
19. Giao diện tương ứng hiển thị.
20. Nhân viên click nút sửa phòng.
21. Phương thức actionPerformed() được kích hoạt.
22. Phương thức actionPerformed() gọi lớp SearchRoomFrm.
23. Phương thức SearchRoomFrm() được kích hoạt
24. Giao diện tương ứng hiển thị.
25. Nhân viên nhập tên phòng và click tìm.
26. Phương thức actionPerformed() được kích hoạt.
27. Phương thức actionPerformed() gọi phương thức searchRoom() của lớp RoomDAO.
28. Phương thức searchRoom() tìm phòng theo tên.
29. Phương thức searchRoom() gọi lớp Room để đóng gói kết quả tìm kiếm.
30. Lớp Room thực hiện đóng gói vào thực thể Room.
31. Lớp Room trả đối tượng về cho phương thức searchRoom().
32. Phương thức searchRoom() trả kết quả lại cho phương thức actionPerformed() của lớp SearchRoomFrm.

33. Phương thức actionPerformed() hiện danh sách kết quả được hiện lên cho nhân viên.
34. Nhân viên click 1 phòng trong danh sách để sửa.
35. Phương thức actionPerformed() lại được kích hoạt.
36. Phương thức actionPerformed() gọi lớp EditRoomFrm.
37. Phương thức EditRoomFrm() được gọi,
38. Giao diện tương ứng được hiển thị.
39. Nhân viên sửa một vài thuộc tính của phòng và click lưu.
40. Phương thức actionPerformed() của lớp EditRoomFrm được kích hoạt
41. Phương thức actionPerformed() gọi lớp Room để set lại thuộc tính được sửa.
42. Lớp Room thực hiện set lại các thuộc tính cho thực thể Room
43. Lớp Room trả về đối tượng cho phương thức actionPerformed().
44. Phương thức actionPerformed() gọi phương thức updateRoom() của lớp RoomDAO.
45. Phương thức updateRoom() cập nhật thông tin phòng vào CSDL
46. Phương thức updateRoom() trả quyền điều khiển về phương thức actionPerformed().
47. Phương thức actionPerformed() hiện thông báo thành công.
48. Nhân viên click vào OK của thông báo.
49. Phương thức actionPerformed() gọi lại lớp ManagerHomeFrm.
50. Lớp ManagerHomeFrm lại hiển thị.

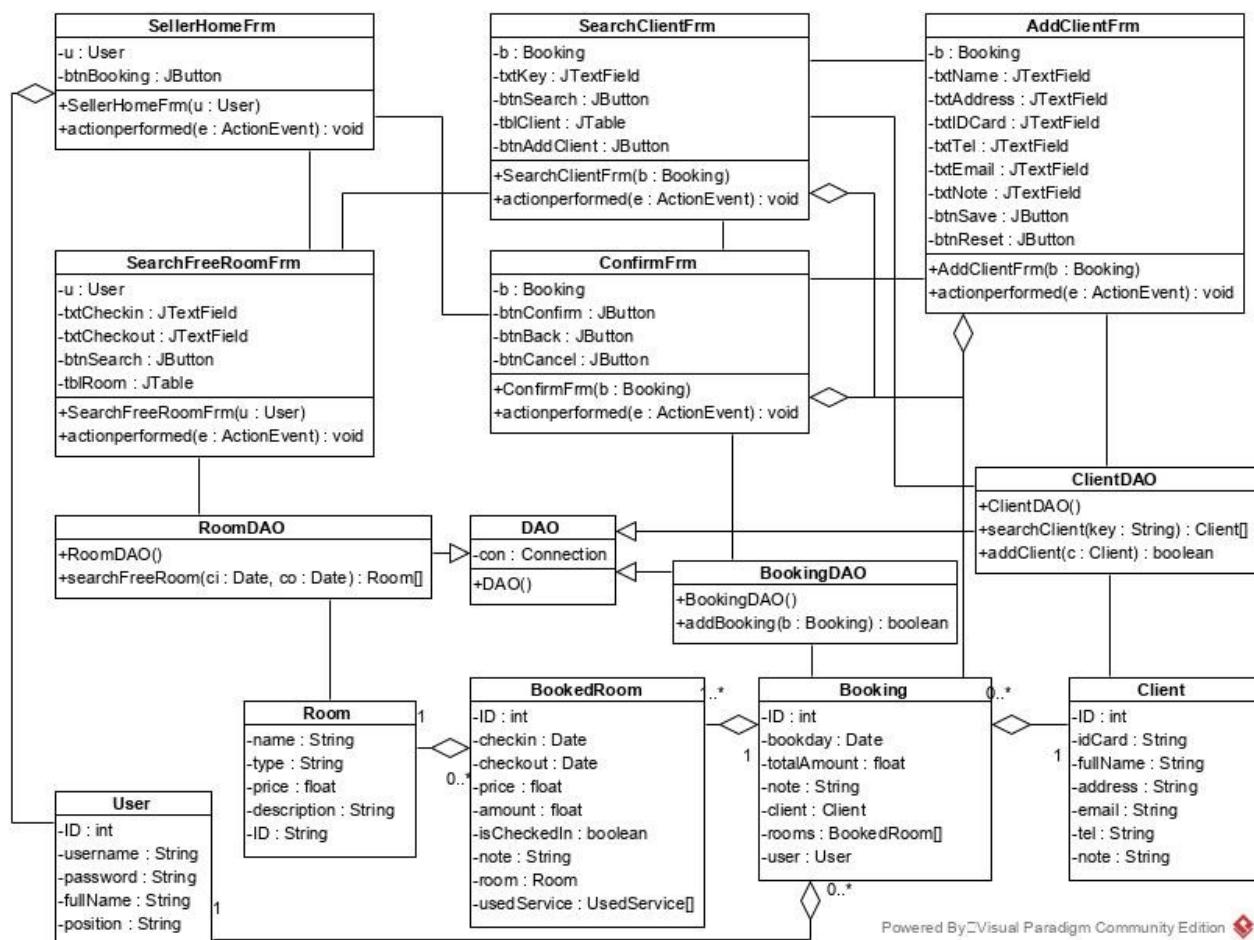
**b. Thiết kế chi tiết cho chức năng đặt phòng.**

Đối với thiết kế tinh, bỏ qua phần đăng nhập, modul này cần các lớp theo ba tầng MVC như sau (Hình 8.13):

- Tầng giao diện: SellerHomeFrm là giao diện chính của nhân viên bán hàng. SearchFreeRoomFrm là giao diện tìm phòng trống. SearchClientFrm là giao diện tìm khách hàng. AddClientFrm là giao diện thêm khách hàng trong trường hợp khách hàng mới. ConfirmFrm là giao diện xác nhận đặt phòng.

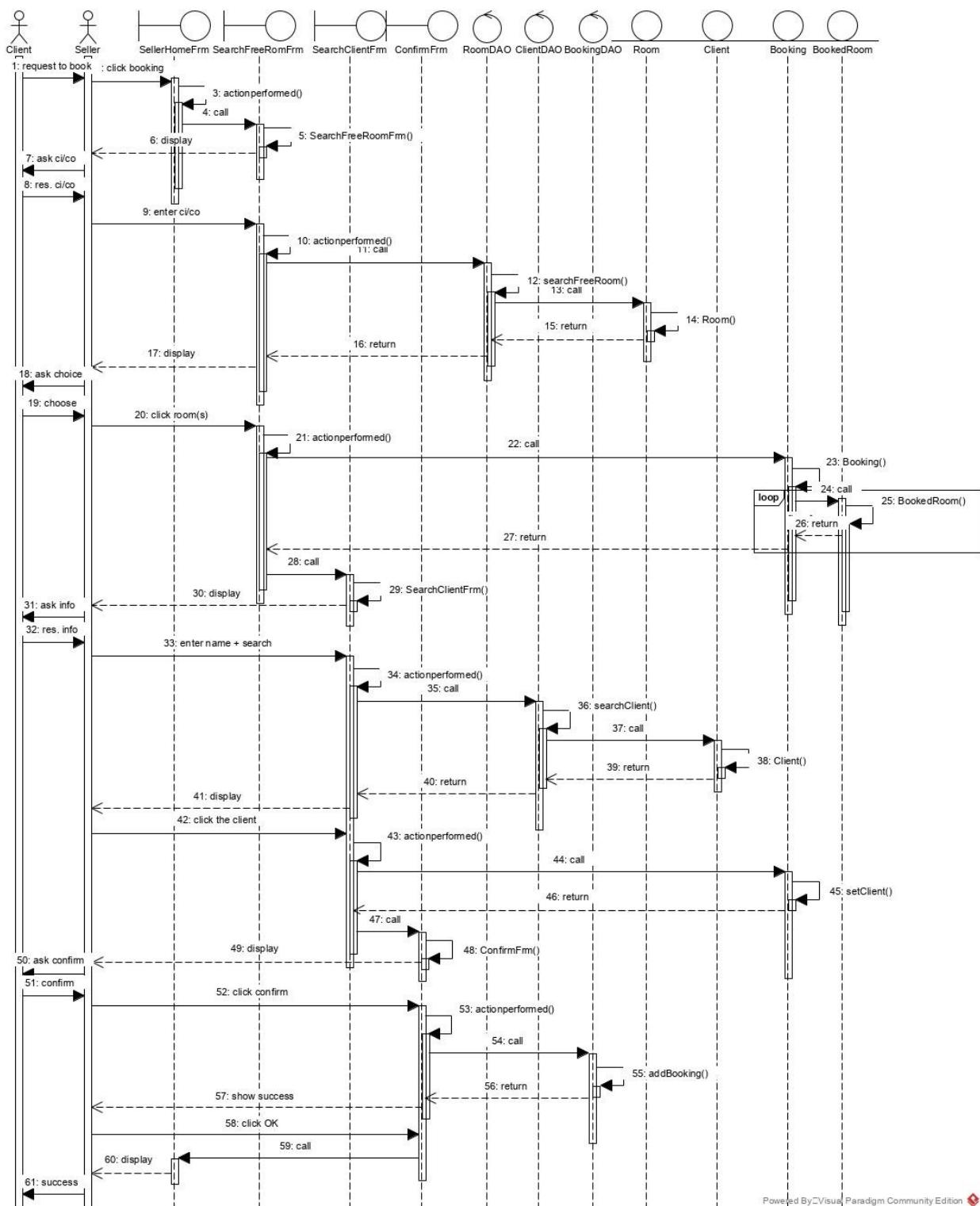
- Tầng điều khiển: RoomDAO cần để thực hiện tìm phòng trống. ClientDAO cần để thực hiện tìm khách hàng và thêm khách hàng, nếu cần. BookingDAO cần để thêm thông tin đặt phòng mới. Các lớp này đều kế thừa từ lớp DAO để điều khiển truy nhập chung vào CSDL.
- Tầng thực thể: Cần các lớp Room, Client, Booking. Lớp Booking cần có các lớp thành phần BookedRoom và User.

Các bước hoạt động của modul này, bỏ qua phần xử lý đăng nhập, được mô tả tuần tự như trong Hình 8.14.



Hình 8.13: Biểu đồ lớp thiết kế chi tiết chức năng đặt phòng

1. Khách hàng gọi điện cho nhân viên bán hàng yêu cầu đặt phòng.
2. Nhân viên click chức năng đặt phòng trên giao diện chính của mình (sau khi đăng nhập).
3. Phương thức actionPerformed() của lớp SellerHomeFrm được kích hoạt.



Hình 8.14: Biểu đồ tuần tự thiết kế chi tiết chức năng đặt phòng

4. Phương thức actionPerformed() gọi lớp SearchFreeRomFrm.

5. Phương thức SearchFreeRoomFrm() được kích hoạt
6. Giao diện tương ứng hiển thị.
7. Nhân viên hỏi khách hàng ngày nhận/trả phòng.
8. Khách hàng nói ngày nhận/trả phòng mà mình mong muốn cho nhân viên.
9. Nhân viên nhập ngày nhận/trả phòng và click tìm.
10. Phương thức actionPerformed() của lớp SearchFreeRoomFrm được kích hoạt.
11. Phương thức actionPerformed() gọi phương thức searchFreeRoom() của lớp RoomDAO.
12. Phương thức searchFreeRoom() thực hiện tìm kiếm trong CSDL.
13. Phương thức searchFreeRoom() gọi lớp Room để đóng gói kết quả tìm kiếm được.
14. Lớp Room đóng gói đối tượng
15. Lớp Room trả về cho phương thức searchFreeRoom().
16. Phương thức searchFreeRoom() trả kết quả về cho phương thức actionPerformed()
17. Phương thức actionPerformed() hiển thị kết quả lên giao diện SearchFreeRoomFrm.
18. Nhân viên thông báo các loại phòng còn trống cho khách hàng và yêu cầu khách hàng chọn phòng mong muốn.
19. Khách hàng yêu cầu về phòng cho nhân viên.
20. Nhân viên click vào phòng tương ứng.
21. Phương thức actionPerformed() của lớp SearchFreeRoomFrm lại được kích hoạt.
22. Phương thức actionPerformed() gọi lớp Booking để đóng gói thông tin để truyền cho các giao diện tiếp theo.
23. Lớp Booking gọi các hàm đóng gói
24. Lớp Booking gọi lớp BookedRoom để đóng gói thuộc tính thành phần
25. Lớp BookedRoom đóng gói các thuộc tính của mình
26. Lớp BookedRoom trả lại lớp Booking.
27. Lớp Booking trả lại phương thức actionPerformed().
28. Phương thức actionPerformed() gọi lớp SearchClientFrm.

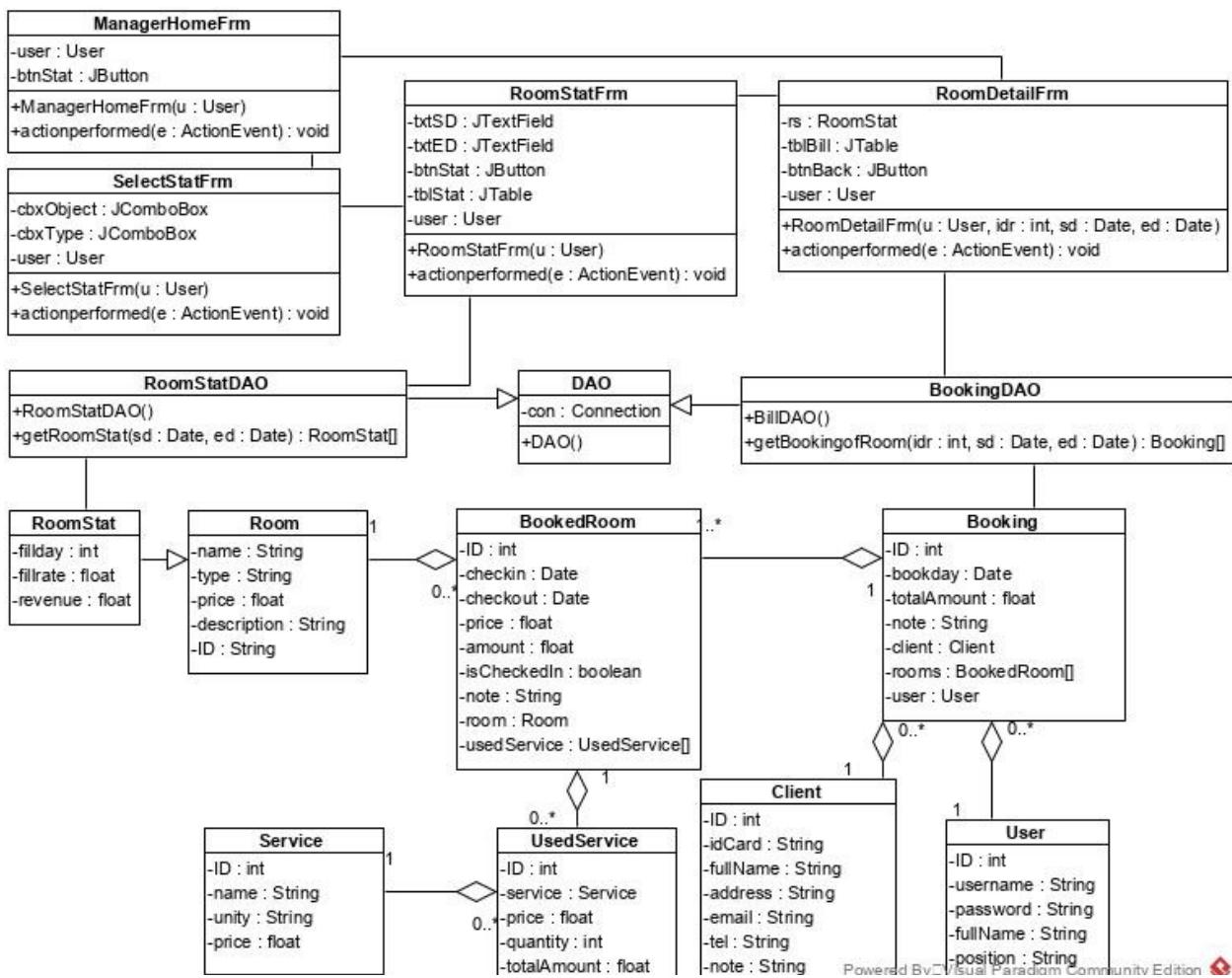
29. Phương thức SearchClientFrm() được kích hoạt
30. Giao diện tương ứng hiển thị.
31. Nhân viên hỏi xin thông tin cá nhân của khách hàng.
32. Khách hàng cung cấp thông tin cá nhân cho nhân viên.
33. Nhân viên gõ tên khách hàng vào và click tìm kiếm.
34. Phương thức actionPerformed() của lớp SearchClientFrm được kích hoạt.
35. Phương thức actionPerformed() gọi phương thức searchClient() của lớp ClientDAO.
36. Phương thức searchClient() thực hiện tìm kiếm khách hàng từ CSDL.
37. Phương thức searchClient() gọi lớp Client để đóng gói thông tin đã tìm được.
38. Lớp Client đóng gói các đối tượng
39. Lớp Client trả về cho phương thức searchClient().
40. Phương thức searchClient() trả kết quả lại cho phương thức actionPerformed().
41. Phương thức actionPerformed() hiển thị kết quả lên giao diện SearchClientFrm.
42. Nhân viên click vào dòng đúng thông tin khách hàng vừa cung cấp.
43. Phương thức actionPerformed() của lớp SearchClientFrm lại được kích hoạt.
44. Phương thức actionPerformed() gọi lớp Booking để bổ sung thuộc tính khách hàng
45. Lớp Booking gọi hàm setClient()
46. Lớp Booking trả lại phương thức actionPerformed()
47. Phương thức actionPerformed() gọi lớp ConfirmFrm.
48. Phương thức ConfirmFrm() được kích hoạt
49. Giao diện tương ứng hiển thị.
50. Nhân viên xác nhận lại các thông tin đặt phòng với khách hàng.
51. Khách hàng xác nhận đúng và đồng ý đặt phòng.
52. Nhân viên click vào nút xác nhận.
53. Phương thức actionPerformed() của lớp ConfirmFrm được kích hoạt.

54. Phương thức actionPerformed() gọi phương thức addBooking() của lớp BookingDAO.
55. Phương thức addBooking() lưu thông tin đặt phòng vào CSDL.
56. Phương thức addBooking() trả lại quyền điều khiển cho phương thức actionPerformed()
57. Phương thức actionPerformed() thông báo đặt phòng thành công
58. Nhân viên click OK của thông báo.
59. Phương thức actionPerformed() gọi lại giao diện SellerHomeFrm.
60. Giao diện SellerHomeFrm được hiển thị.
61. Nhân viên xác nhận đặt phòng thành công với khách hàng.

**c. Thiết kế chi tiết cho chức năng xem thống kê phòng theo doanh thu.**

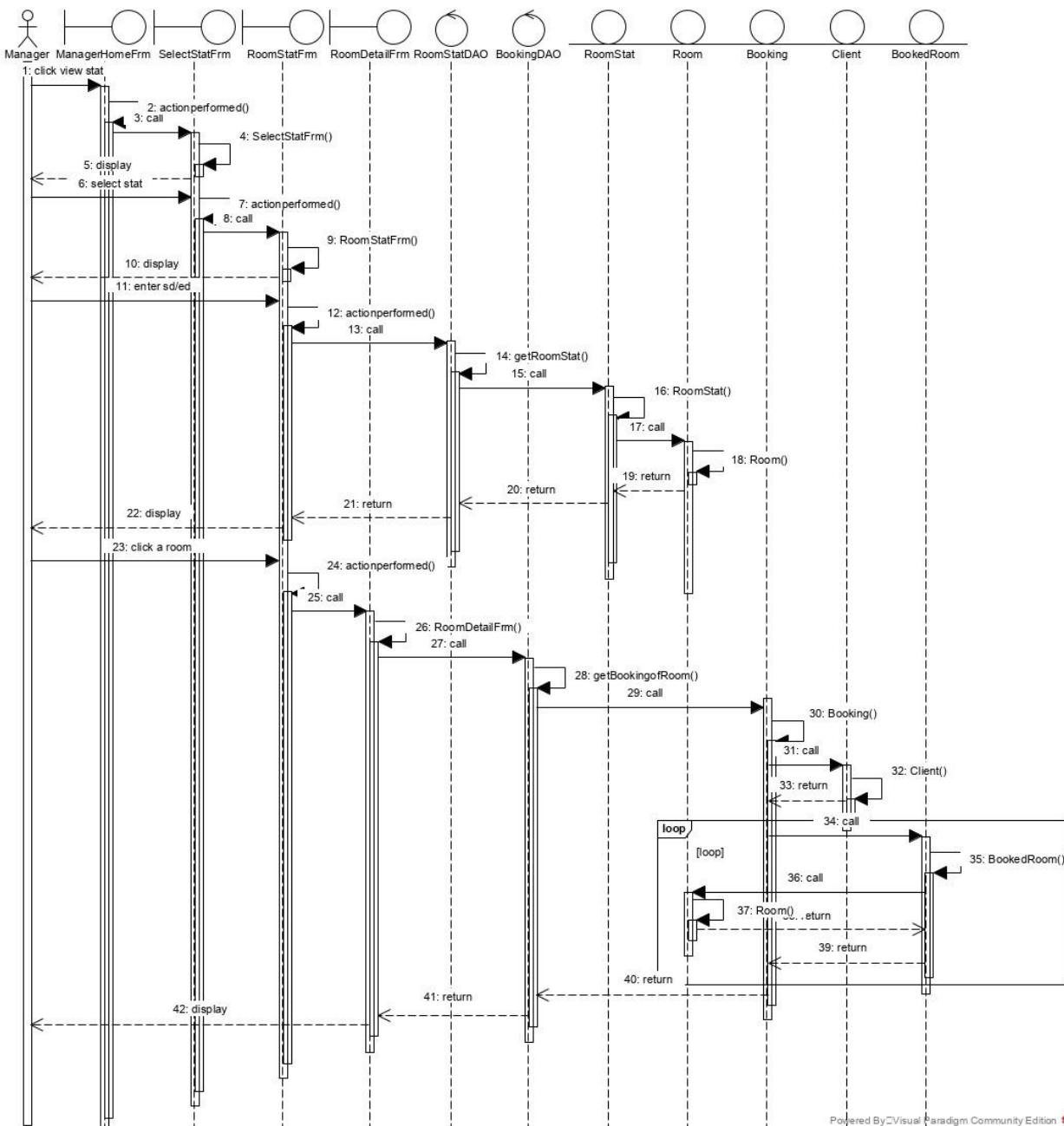
Bỏ qua phần xử lý đăng nhập, modul này cần các lớp cho thiết kế tĩnh theo mô hình MVC như sau (Hình 8.15):

- Tầng giao diện: ManagerHomeFrm là giao diện chính của nhân viên quản lý. SelectStatFrm là giao diện lựa chọn thể loại thống kê. RoomStatFrm là giao diện thống kê phòng. RoomDetailFrm là giao diện thống kê chi tiết một phòng.
- Tầng điều khiển: RoomStatDAO cần phương thức getRoomStat() để thống kê phòng theo doanh thu. BillDAO cần phương thức getBillbyRoom() để thống kê cho biết các hóa đơn của một phòng. Hai lớp này đều được kế thừa lớp DAO để điều khiển truy cập chung vào CSDL.
- Tầng thực thể: RoomStat và Bill. RoomStat kế thừa từ Room. Bill cần các lớp thành phần User và Booking. Booking cần các lớp thành phần BookedRoom, Client và UsedService. Lớp UsedService cần lớp thành phần Service.



Chi tiết hoạt động của chức năng này được mô tả trong Hình 8.16:

- Nhân viên quản lý click chọn chức năng xem thống kê từ giao diện chính sau khi đăng nhập.
- Phương thức actionPerformed() của lớp ManagerHomeFrm được kích hoạt.
- Phương thức actionPerformed() gọi lớp SelectStatFrm.
- Phương thức SelectStatFrm() được kích hoạt, giao diện tương ứng được hiển thị.
- Nhân viên click chọn thẻ loại thống kê là thống kê phòng theo doanh thu.
- Phương thức actionPerformed() của lớp SelectStatFrm được kích hoạt
- Phương thức actionPerformed() gọi lớp RoomStatFrm.
- Phương thức RoomStatFrm() được kích hoạt, hiển thị giao diện tương ứng.



Hình 8.16: Biểu đồ tuần tự thiết kế chi tiết chức năng xem thông kê phòng theo doanh thu

9. Nhân viên chọn ngày bắt đầu/kết thúc thời gian thống kê và click xem thống kê.
10. Phương thức actionPerformed() của lớp RoomStatFrm được kích hoạt.
11. Phương thức actionPerformed() gọi phương thức getRoomStat() của lớp RoomStatDAO.
12. Phương thức getRoomStat() lấy số liệu thống kê từ CSDL.

13. Phương thức getRoomStat() gọi các phương thức của lớp RoomStat để đóng gói kết quả thống kê.
14. Lớp RoomStat cũng phải gọi các phương thức của lớp Room để đóng gói.
15. Kết quả được RoomStat trả lại cho phương thức actionPerformed()
16. Phương thức actionPerformed() hiển thị kết quả thống kê lên giao diện RoomStatFrm.
17. Nhân viên click vào một phòng để xem chi tiết.
18. Phương thức actionPerformed() của lớp RoomStatFrm lại được kích hoạt.
19. Phương thức actionPerformed() gọi lớp RoomDetailFrm
20. Phương thức RoomDetailFrm() được kích hoạt.
21. Phương thức RoomDetailFrm() gọi phương thức getBookingOfRoom() của lớp BookingDAO.
22. Phương thức getBookingOfRoom() lấy dữ liệu từ CSDL.
23. Phương thức getBookingOfRoom() gọi các phương thức của lớp Booking để đóng gói dữ liệu.
24. Lớp Booking gọi các phương thức lớp BookedRoom, Client để đóng gói thuộc tính thành phần.
25. BookedRoom gọi các phương thức của lớp Room để đóng gói thuộc tính thành phần
26. Kết quả được lớp BookingDAO trả lại cho phương thức RoomDetailFrm()
27. Phương thức RoomDetailFrm() hiển thị danh sách hóa đơn chi tiết lên giao diện RoomDetailFrm.
28. Nhân viên xem xong và click quay lại

## CHƯƠNG 9: PHA CÀI ĐẶT VÀ TÍCH HỢP

### 9.1 CÁC PHƯƠNG PHÁP CÀI ĐẶT VÀ TÍCH HỢP

#### 9.1.1 Luồng công việc cài đặt

- Mục đích của luồng công việc cài đặt là cài đặt phần mềm đích
- Phần mềm lớn được chia thành các hệ thống con
  - Được cài đặt song song bởi các đội việt mã
- Các hệ thống con bao gồm các thành phần và các mô đun mã
- Một khi người lập trình đã cài đặt một mô đun, người ấy thực hiện kiểm thử đơn vị mô đun đó
- Sau đó mô đun được chuyển qua nhóm SQA để kiểm thử mức cao hơn
  - Kiểm thử này là một phần của luồng công việc kiểm thử

##### 9.1.1.1 Chọn ngôn ngữ lập trình

- Ngôn ngữ thường được chỉ rõ trong hợp đồng
- Nhưng điều gì sẽ xảy ra khi hợp đồng chỉ ra rằng:
  - Sản phẩm phần mềm được cài đặt bằng ngôn ngữ lập trình “phù hợp nhất”
- Ngôn ngữ nào nên được chọn?
- Ví dụ:
  - Tổ chức QQQ đã viết bằng ngôn ngữ COBOL suốt 25 năm qua
  - Trên 200 nhân viên phần mềm, tất cả đều là chuyên gia về COBOL
  - Ngôn ngữ lập trình nào là phù hợp nhất?
- Hiển nhiên COBOL
- Chuyện gì xảy ra khi ngôn ngữ lập trình mới (như C++) được giới thiệu:
  - Những chuyên gia C++ phải thuê
  - Những chuyên gia COBOL vốn có phải đào tạo lại
  - Những sản phẩm trong tương lai được viết bằng C++
  - Những sản phẩm phần mềm COBOL sẵn có phải được bảo trì
  - Có hai kiểu người lập trình khác nhau
    - Những người bảo trì COBOL (bị coi nhẹ)
    - Những người lập trình C++ (được trả nhiều tiền hơn)
  - Yêu cầu phần mềm và phần cứng đặt tiền để chạy ngôn ngữ lập trình
  - Hàng trăm chuyên gia COBOL bị bỏ phế
- Kết luận duy nhất là:
  - COBOL là ngôn ngữ lập trình phù hợp nhất
- Và ngôn ngữ phù hợp nhất cho dự án mới nhất có thể là C++
  - COBOL phù hợp với những ứng dụng chỉ xử lý dữ liệu

- Cách chọn ngôn ngữ lập trình
  - Phân tích lợi nhuận – chi phí
  - Tính toán chi phí và lợi nhuận của tất cả các ngôn ngữ liên quan
- Ngôn ngữ hướng đối tượng nào thích hợp nhất?
  - C++ giống C (C++ is (unfortunately) C-like)
  - Do đó, mỗi chương trình C có thể tự động là chương trình C++
  - Java được yêu cầu đổi với mô hình hướng đối tượng
  - Việc đào tạo trong mô hình hướng đối tượng là cần thiết trước khi áp dụng bất cứ ngôn ngữ hướng đối tượng nào
- Còn việc lựa chọn ngôn ngữ thế hệ thứ tư? (Fourth generation language -4GL)?

#### 9.1.1.2 Ngôn ngữ thế hệ thứ tư

- Ngôn ngữ thế hệ thứ nhất
  - Ngôn ngữ máy
- Ngôn ngữ thế hệ thứ hai
  - Hợp ngữ
- Ngôn ngữ thế hệ thứ ba
  - Ngôn ngữ bậc cao (COBOL, FORTRAN, C++, Java)
- Ngôn ngữ thế hệ thứ tư (4GLs)
  - Một câu lệnh ngôn ngữ thế hệ thứ ba tương đương với 5 đến 10 câu lệnh hợp ngữ
  - Mỗi câu lệnh ngôn ngữ thứ tư tương đương với 30 hoặc 50 câu lệnh hợp ngữ
- Hy vọng rằng ngôn ngữ thứ tư sẽ:
  - Tăng nhanh tốc độ xây dựng ứng dụng
  - Kết quả của các ứng dụng là dễ dàng xây dựng và nhanh chóng thay đổi
    - Giảm chi phí bảo trì
  - Đơn giản trong việc gỡ lỗi
  - Tạo ngôn ngữ thân thiện người dùng
- Có thể thực hiện được nếu ngôn ngữ thứ tư là ngôn ngữ bậc cao, thân thiện với người dùng
- Những đóng góp vào thị trường:
  - Không có một ngôn ngữ thế hệ thứ 4 nào chiếm ưu thế trong thị trường phần mềm
  - Có hàng trăm 4GL
  - Hàng tá nhóm người dùng cỡ lớn
  - Oracle, DB2, và PowerBuilder cực kỳ phổ biến
- Lý do
  - Không có một 4GL có đủ những đặc trưng cần thiết
- Kết luận

- Đặc biệt quan tâm đến việc lựa chọn 4GL thích hợp

#### *Tăng hiệu năng với 4GL*

- Bức tranh không phải toàn màu hồng
- Playtex used ADF, obtained an 80 to 1 productivity increase over COBOL
  - However, Playtex then used COBOL for later applications
- 4GL productivity increases of 10 to 1 over COBOL have been reported
  - Tuy nhiên, có quá nhiều bản tường trình của những thử nghiệm tồi

#### *Những thử nghiệm thực tế với 4GL*

- Nhiều ngôn ngữ thế hệ thứ tư được hỗ trợ bởi môi trường CASE mạnh mẽ
  - Đây là một vấn đề đối với những tổ chức ở mức CMM 1 hoặc 2
  - Một vài thất bại của ngôn ngữ thế hệ thứ 4 được ghi lại là do môi trường CASE vật lý
- Quan điểm của 43 tổ chức đối với 4GLs
  - Việc sử dụng 4GL đã giảm sự thất vọng của người dùng
  - Đáp ứng nhanh hơn từ bộ phận DP (Quicker response from DP department )
  - 4GLs are slow and inefficient, on average
  - Nhìn chung, 28 tổ chức sử dụng 4GL trong 3 năm thấy lợi nhuận thu được vượt quá chi phí bỏ ra

#### *Những nguy hiểm với ngôn ngữ thứ tư*

- End-user programming
  - Những người lập trình được đào tạo để nghi ngờ các đầu ra của máy tính (Programmers are taught to mistrust computer output)
  - Những dùng cuối được dạy để tin tưởng vào đầu ra của máy tính (End users are taught to believe computer output)
  - Người dùng cuối đang cập nhật cơ sở dữ liệu có thể đặc biệt nguy hiểm (An end-user updating a database can be particularly dangerous)
- Những nguy hiểm tiềm năng đối với quản lý (Potential pitfalls for management)
  - Trường hợp giới thiệu sớm môi trường CASE (Premature introduction of a CASE environment)
  - Đào tạo không đủ đối với độ phát triển (Providing insufficient training for the development team)
  - Chọn 4GL sai

#### *9.1.1.3 Lập trình tốt trong thực tế (Good Programming Practice)*

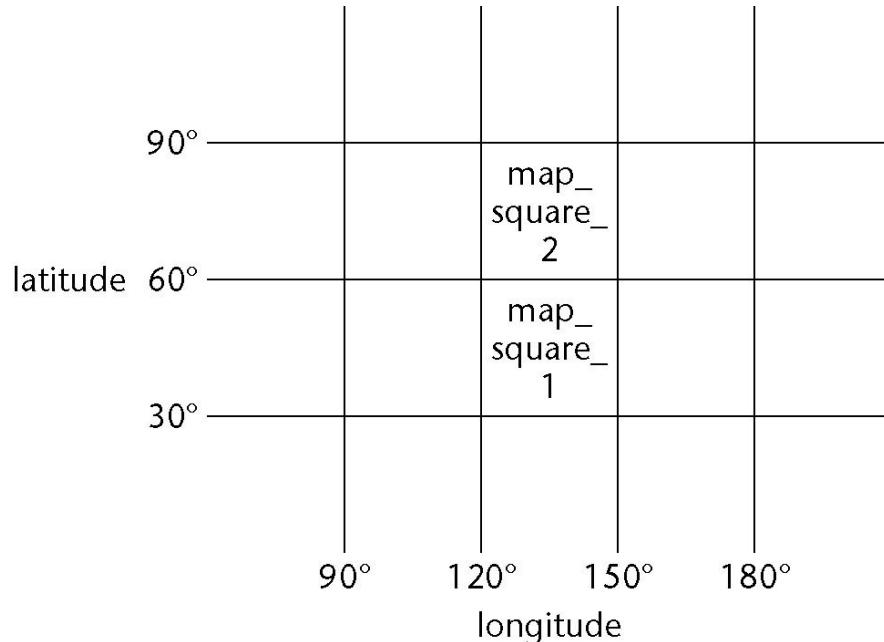
- Sử dụng tên biến nhất quán và có ý nghĩa
  - “Có ý nghĩa” để những người lập trình bảo trì trong tương lai
  - “Nhất quán” để trợ giúp cho những người bảo trì trong tương lai

- Tài liệu mã bao gồm tên các biến như freqAverage, frequencyMaximum, minFr, frqncytotl
- Người lập trình bảo trì phải biết nếu freq, frequency, fr, frqncyt đều liên quan đến cùng một thứ
  - Nếu sử dụng từ đồng nhất, tốt nhất là frequency, có thể freq hoặc frqncyt, không thể fr
  - Nếu không sử dụng một từ khác (như: rate) cho một số lượng khác
- Chúng ta có thể sử dụng frequencyAverage, frequencyMaximum, frequencyMinimum, frequencyTotal
- Chúng ta cũng có thể sử dụng averageFrequency, maximumFrequency, minimumFrequency, totalFrequency
- Nhưng bốn tên phải xuất phát cùng một tập
- Vấn đề của **Self-Documenting Code**
  - Self-documenting code cực kỳ hiếm
  - Vấn đề chính: tài liệu mã có thể được hiểu một cách dễ dàng và không nhầm lẫn bởi:
    - Đội SQA
    - Những người lập trình bảo trì
    - Những người khác đọc mã
  - Ví dụ:
    - Tài liệu mã bao gồm biến xCoordinateOfPositionOfRobotArm
    - Biến này được viết tắt là xCoord
    - Biến này rất tốt, bởi vì toàn bộ mô đun xử lý sự di chuyển của cánh tay rõ ràng
    - Nhưng người lập trình bảo trì có biết điều này không?
  - Những lời giải thích mở đầu tối thiểu cho một tài liệu viết mã artifact

The name of the code artifact  
A brief description of what the code artifact does  
The programmer's name  
The date the code artifact was coded  
The date the code artifact was approved  
The name of the person who approved the code artifact  
The arguments of the code artifact  
A list of the name of each variable of the code artifact, preferably in alphabetical order, and a brief description of its use  
The names of any files accessed by this code artifact  
The names of any files changed by this code artifact  
Input–output, if any  
Error-handling capabilities  
The name of the file containing test data (to be used later for regression testing)  
A list of each modification made to the code artifact, the date the modification was made, and who approved the modification  
Any known faults

- Những đề nghị (Suggestion): Những lời giải thích là cần thiết khi mã được viết theo cách không rõ ràng hoặc cách sử dụng một khía cạnh tinh tế nào đó của ngôn ngữ
- Lời giải thích vô nghĩa (Nonsense): Viết lại mã theo cách rõ ràng hơn, chúng ta không bao giờ thúc đẩy/bỏ qua việc lập trình tồi. Tuy nhiên, những lời giải thích có thể trợ giúp những người lập trình bảo trì trong tương lai
- 
- Việc sử dụng các tham số
  - Gần như không có những hằng số thực
  - Một giải pháp:
    - Sử dụng câu lệnh **const** (C++), hoặc
    - Sử dụng câu lệnh **public static final** (Java)
  - Một cách tốt hơn:
    - Đọc những giá trị “hằng số” từ tệp tham số
- Việc bố trí mã để tăng khả năng có thể đọc được
  - Sử dụng thụt đầu dòng
  - Tốt hơn, sử dụng (Better, use a pretty-printer)
  - Sử dụng nhiều dòng trống
    - Để tách những khối lệnh lớn (To break up big blocks of code)
- Câu lệnh if lồng

- o Ví dụ: Một bản đồ bao gồm hai hình vuông. Viết mã để xác định liệu một điểm nằm trên bề mặt trái đất nằm trong map\_square\_1 hoặc map\_square\_2, hoặc không nằm trong hình vuông nào



Hình 9.1: ví dụ xác định vị trí theo kinh độ, tọa độ

- o Giải pháp 1: Được định dạng tồi

```
if (latitude > 30 && longitude > 120) {if (latitude <= 60 && longitude <= 150)
mapSquareNo = 1; else if (latitude <= 90 && longitude <= 150) mapSquareNo = 2
else print "Not on the map"; else print "Not on the map";
```

- o Định dạng tốt, được cấu trúc tồi

```
if (latitude > 30 && longitude > 120)
{
    if (latitude <= 60 && longitude <= 150)
        mapSquareNo = 1;
    else
        if (latitude <= 90 && longitude <= 150)
            mapSquareNo = 2;
        else
            print "Not on the map";
}
else
    print "Not on the map";
```

- o Cấu trúc lồng được chấp nhận

```

if (longitude > 120 && longitude <= 150 && latitude > 30 && latitude <= 60)
    mapSquareNo = 1;
else
    if (longitude > 120 && longitude <= 150 && latitude > 60 && latitude <= 90)
        mapSquareNo = 2;
    else
        print "Not on the map";
    o Sự kết hợp của câu lệnh if-if và if-else-if thường rất khó đọc
    o Đơn giản: Sự kết hợp if-if
        if <condition1>
            if <condition2>
                tương đương với điều kiện đơn
                if <condition1> && <condition2>
            o Quy luật ngón tay cái (Rule of thumb) : Câu lệnh if được lồng lớn hơn ba lần nên
                tránh vì đó là cách lập trình tồi

```

#### 9.1.1.4 Những chuẩn lập trình

- Standards can be both a blessing and a curse
- Những mô đun có độ kết dính ngẫu nhiên (coincidental cohesion) sinh ra từ các luật giống nhau:
  - o “Mỗi mô đun sẽ bao gồm 35 và 50 câu lệnh có thể thực thi được”
- Tốt hơn
  - o “Những người lập trình nên hỏi ý kiến những người quản lý của họ trước khi xây dựng một mô đun với ít hơn 35 hoặc nhiều hơn 50 câu lệnh có thể thực thi được”

Nhận xét:

- Chưa từng có một chuẩn nào được chấp nhận phổ biến
- Các chuẩn đã áp đặt từ bên trên sẽ được bỏ qua
- Các chuẩn có thể kiểm tra bằng máy
- Mục đích của chuẩn là để bảo trì dễ dàng
  - o Nếu các chuẩn làm cho việc phát triển gặp khó khăn, thì chúng phải được chỉnh sửa
  - o Những chuẩn giới hạn quá mức phản tác dụng (Overly restrictive standards are counterproductive)
  - o Chất lượng phần mềm có áp dụng các chuẩn (The quality of software suffers)

#### Ví dụ của chuẩn lập trình tốt

- “Việc xếp lồng vào nhau các câu lệnh **if** không nên vượt quá 3 lần, ngoại trừ được phê chuẩn trước từ đội trưởng”
- “Các mô đun gồm từ 35 đến 50 câu lệnh, ngoại trừ có sự phê chuẩn từ trước của đội trưởng”
- “Sử dụng câu lệnh **goto** nên tránh. Tuy nhiên, với sự phê chuẩn từ trước của đội trưởng, lệnh **goto** có thể được sử dụng để xử lý lỗi”

#### 9.1.1.5 Sử dụng lại mã

- Sử dụng lại mã là một dạng sử dụng lại phổ biến nhất
- Tuy nhiên, tài liệu của tất cả các luồng công việc có thể được sử dụng lại

#### 9.1.1.6 Công cụ CASE cho cài đặt

- Công cụ CASE sử dụng cho tích hợp gồm
  - Công cụ điều khiển phiên bản, công cụ điều khiển cấu hình, và công cụ xây dựng
  - Ví dụ:
    - rcs, sccs, PCVS, SourceSafe
- Công cụ điều khiển cấu hình
  - Mang tính thương mại
    - PCVS, SourceSafe
  - Mã nguồn mở
    - CVS

#### Các công cụ CASE cho tiến trình phần mềm hoàn thiện

- Một tổ chức lớn cần một môi trường
- Một tổ chức với cỡ trung bình có thể quản lý sử dụng workbench (A medium-sized organization can probably manage with a workbench)
- Một tổ chức nhỏ có thể quản lý mà chỉ sử dụng các công cụ

#### Môi trường phát triển đã tích hợp

- Ý nghĩa của từ “tích hợp”
  - Tích hợp giao diện người dùng
  - Tương tự “nhìn và cảm nhận”
  - Thành công nhất trên hệ điều hành Macintosh
- Cũng có các kiểu tích hợp khác
- Tích hợp công cụ
  - Tất cả các công cụ giao tiếp sử dụng cùng một định dạng
  - Ví dụ:
    - Unix Programmer’s Workbench
- Tích hợp tiến trình
  - Môi trường hỗ trợ một tiến trình riêng biệt
  - Tập con: Môi trường dựa trên kỹ thuật
    - Trước đây: “môi trường dựa trên phương thức”
    - Hỗ trợ một kỹ thuật riêng biệt hơn là một tiến trình hoàn thiện
    - Môi trường của các kỹ thuật là: Phân tích hệ thống trúc (Structured systems analysis) và Petri nets
- Môi trường dựa trên kỹ thuật
  - Đồ họa hỗ trợ cho phân tích, thiết kế

- Tùy biến dữ liệu
- Việc vài kiểm tra tính nhất quán
- Hỗ trợ quản lý
- Hỗ trợ và hình thức hóa tiến trình bằng tay
- Ví dụ:
  - Analyst/Designer
  - Software through Pictures
  - IBM Rational Rose
  - Rhapsody (for Statecharts)
- Thuận lợi:
  - Người dùng ép buộc phải sử dụng một phương pháp cụ thể, chính xác
- Bất lợi:
  - Người dùng bị ép buộc sử dụng một phương thức cụ thể, vì thế phương thức đó phải là một phần của tiến trình phần mềm của tổ chức đó (The user is forced to use one specific method, so that the method must be part of the software process of that organization)

#### *Môi trường của các ứng dụng doanh nghiệp*

- Nhấn mạnh tính dễ dàng khi sử dụng bao gồm
  - Bộ sinh giao diện người dùng thân thiện
  - Chuẩn màn hình cho đầu vào và đầu ra, và
  - Bộ sinh mã
    - Thiết kế chi tiết là mức thấp nhất của trừu tượng
    - Thiết kế chi tiết là đầu vào của bộ sinh mã
- Việc sử dụng ngôn ngữ lập trình này làm tăng hiệu năng
- Ví dụ: Oracle Development Suite
- PCTE — Portable common tool environment
  - Không phải là một môi trường
  - Là một cơ sở hạ tầng để hỗ trợ giúp công cụ CASE (tương tự với cách hệ điều hành cung cấp các dịch vụ cho các sản phẩm phần mềm người dùng)
  - Được chấp nhận bởi ECMA (European Computer Manufacturers Association)
- Ví dụ sự cài đặt:
  - IBM, Emeraude

#### *Những vấn đề xảy ra với môi trường*

- Không có môi trường lý tưởng cho tất cả các tổ chức
  - Mỗi môi trường có điểm mạnh điểm yếu
- Cảnh báo 1
  - Chọn môi trường sai có thể tồi hơn khi không có môi trường

- Ép buộc kỹ thuật sai là phản tác dụng
- Cảnh báo 2
  - Những môi trường Shun CASE dưới mức 3 CMM
  - Không thể tự động hóa một tiến trình không tồn tại
  - Tuy nhiên, công cụ CASE hoặc workbench CASE là rất tốt
- Năm thước đo cơ bản cùng với
  - Thước đo độ phức tạp
- Thống kê lỗi là quan trọng
  - Số lượng trường hợp kiểm thử
  - Phần trăm trường hợp kiểm thử sinh ra lỗi
  - Tổng số lượng lỗi
- Dữ liệu lỗi được hợp nhất với danh sách kiểm tra (checklist) đối với quá trình kiểm tra kỹ lưỡng mã

#### 9.1.1.7 Thước đo của luồng công việc cài đặt

- Năm thước đo cơ bản cùng với
  - Thước đo độ phức tạp
- Thống kê lỗi là quan trọng
  - Số lượng trường hợp kiểm thử
  - Phần trăm trường hợp kiểm thử sinh ra lỗi
  - Tổng số lượng lỗi
- Dữ liệu lỗi được hợp nhất với danh sách kiểm tra (checklist) đối với quá trình kiểm tra kỹ lưỡng mã

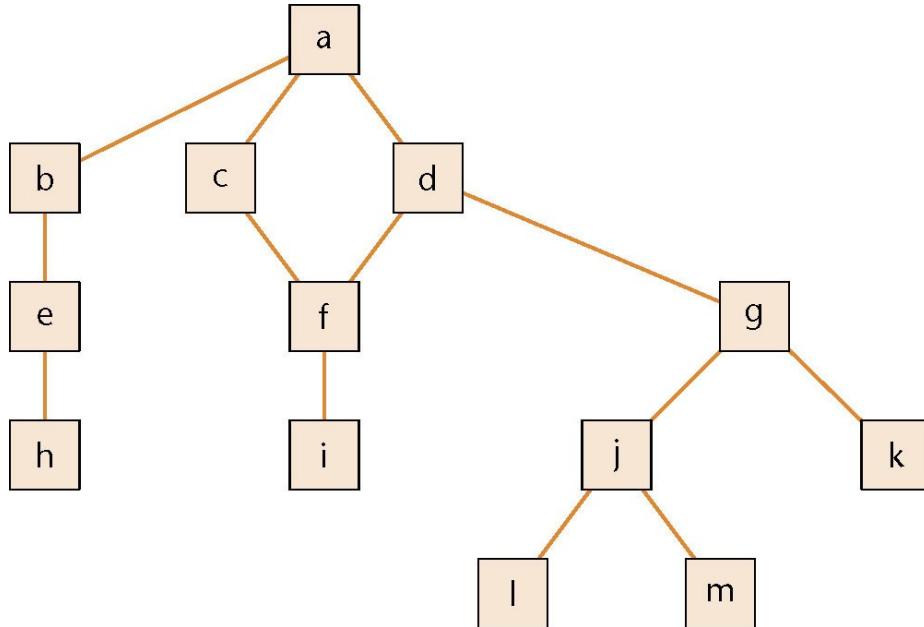
#### 9.1.1.8 Những thách thức của luồng công việc cài đặt

- Những vấn đề quản lý có ý nghĩa lớn ở đây
  - Các công cụ CASE thích hợp
  - Lập kế hoạch kiểm thử
  - Truyền đạt những thay đổi tới tất cả nhân viên
  - Quyết định khi nào dùng kiểm thử
- Sử dụng lại mã cần được đưa vào phần mềm từ lúc bắt đầu
  - Sử dụng lại phải là yêu cầu của khách hàng
  - Kế hoạch quản lý dự án phần mềm phải hợp nhất với việc sử dụng lại
- Cài đặt dễ hiểu về mặt kỹ thuật
  - Những thử thách trong việc quản lý

## 9.1.2 Tích hợp

- Cho đến tận bây giờ phương pháp phổ biến là:

- Tích hợp theo sau tích hợp
- Đây là phương pháp tồi
- Tốt hơn:
  - Kết hợp giữa cài đặt và tích
- Sản phẩm phần mềm với 13 mô đun



Hình 9.2: Ví dụ các modul phải tích hợp

- Cài đặt sau đó tích hợp
  - Viết mã và kiểm thử tài liệu viết mã là tách biệt
  - Liên kết 13 tài liệu với nhau, kiểm thử toàn bộ phần mềm
- **Driver và stub**
  - Để kiểm thử tài liệu a, các tài liệu b,c,d phải trở thành những stub
    - Một tài liệu trống hoặc
    - In ra một thông báo ("Procedure radarCalc called"), hoặc
    - Trả lại một phần giá trị từ các trường hợp kiểm thử đã được lập kế hoạch
  - Để kiểm thử tài liệu h thì yêu cầu một bộ điều khiển (driver) mà sẽ gọi mô đun h
    - Một lần, hoặc
    - Một vài lần, hoặc
    - Nhiều lần, mỗi lần kiểm tra giá trị được trả lại
  - Để kiểm thử tài liệu d yêu cầu một bộ điều khiển và 2 stubs
- Ván đè 1
  - Stubs và drivers phải được viết sau đó phải được kiểm thử đơn vị hoàn thiện mới được sử dụng
- Ván đè 2
  - Thiếu sự cô lập lỗi

- Mọi lỗi có thể nằm ở bất cứ chỗ nào của 13 mô đun ( artifact ) được tạo ra hoặc 13 giao diện (interface)
- Với một phần mềm lớn, có 103 mô đun (artifact) và 108 giao diện (interface), thì phải có 211 chỗ lỗi có thể xảy ra
- Giải pháp cho cả hai vấn đề
  - Kết hợp giữa kiểm thử đơn vị và tích hợp

#### 9.1.2.1 Tích hợp trên xuồng

- Nếu mô đun mAbove gửi một thông điệp tới mô đun mBelow, thì mAbove phải được cài đặt và tích hợp trước mBelow
- Thứ tự từ trên xuống có thể là
  - a,b,c,d,e,f,g,h,i,j,k,l,m
- Một thứ tự khác từ trên xuống
 

a	
[a]	b,e,h
[a]	c,d,f,i
[a,d]	g,j,k,l,m
- Thuận lợi 1: Cô lập lỗi
  - Trường hợp kiểm thử thành công trước đó bị lỗi khi mô đun mNew được thêm vào những cái đã được kiểm thử cho đến lúc này
    - Lỗi phải nằm trong mô đun mNew hoặc giao diện giữa mNew và phần còn lại của phần mềm
- Thuận lợi 2: Stubs không bị lãng phí
  - Mỗi stub được sử dụng vào mô đun hoàn thiện tương ứng ở mỗi bước thích hợp
- Thuận lợi 3: Những thiếu sót thiết kế chính được đưa ra từ sớm
- Các mô đun lô gic bao gồm luồng điều khiển đưa ra quyết định (Logic artifacts include the decision-making flow of control)
  - Trong ví dụ, các mô đun a,b,c,d,g,j
- Các mô đun hoạt động thực hiện những thao tác thực sự của phần mềm
  - Trong ví dụ, các mô đun e,f,h,i,k,l,m
- Các mô đun lô gic được xây dựng trước các mô đun hoạt động
- Vấn đề 1
  - Các mô đun có thể sử dụng lại không được kiểm thử một cách đáng tin cậy
  - Các mô đun mức thấp hơn (mức hành động) không được kiểm thử thường xuyên
  - Vấn đề càng trở nên trầm trọng nếu sản phẩm phần mềm thiết kế tốt (The situation is aggravated if the product is well designed)
- Defensive programming (fault shielding)
  - Ví dụ:
    - if ( $x \geq 0$ )

- $y = \text{computeSquareRoot}(x, \text{errorFlag});$
- computeSquareRoot không bao giờ kiểm thử với  $x < 0$
- Điều này ngụ ý để sử dụng lại

#### 9.1.2.2 Tích hợp dưới lên

- Nếu mô đun mAbove gọi mô đun mBelow, thì mBelow được cài đặt và tích hợp trước mAbove
- Thứ tự tích hợp từ dưới lên có thể là : l,m,h,i,j,k,e,f,g,b,c,d,a
- Một thứ tự tích hợp từ dưới lên khác có thể là  
h,e,b  
i,f,c,d  
l,m,j,k,g            [d]  
a                    [b,c,d]
- Thuận lợi 1
  - Các mô đun hoạt động được kiểm thử kỹ lưỡng
- Thuận lợi 2
  - Các mô đun hoạt động được kiểm thử với các bộ điều khiển (driver), không có tám chắn lỗi, các mô đun được lập trình một cách ....(Operational artifacts are tested with drivers, not by fault shielding, defensively programmed artifacts)
  - Thuận lợi
  - Cô lập lỗi
- Khó khăn 1
  - Các lỗi thiết kế được phát hiện muộn
- Giải pháp
  - Kết hợp chiến lược tích hợp dưới lên và trên xuống để tận dụng điểm mạnh của cả hai chiến lược và cực tiểu điểm yếu của chúng

#### 9.1.2.3 Tích hợp Sandwich

- Các mô đun lô gic được tích hợp trên xuống (Logic artifact)
- Các mô đun hoạt động tích hợp dưới lên (Operational artifacts)
- Cuối cùng, các giao diện của hai nhóm mô đun trên được kiểm thử
- Thuận lợi 1
  - Các lỗi thiết kế chính được tìm thấy sớm
- Thuận lợi 2
  - Các mô đun hoạt động được kiểm thử kỹ lưỡng
  - Chúng có thể được sử dụng lại một cách tin tưởng
- Thuận lợi 3
  - Luôn luôn có sự cô lập lỗi

Tổng kết:

Approach	Strengths	Weaknesses
Implementation then integration (Section 14.6)	—	No fault isolation Major design faults show up late Potentially reusable code artifacts are not adequately tested
Top-down integration (Section 14.6.1)	Fault isolation Major design faults show up early	Potentially reusable code artifacts are not adequately tested
Bottom-up integration (Section 14.6.2)	Fault isolation Potentially reusable code artifacts are adequately tested	Major design faults show up late
Sandwich integration (Section 14.6.3)	Fault isolation Major design faults show up early Potentially reusable code artifacts are adequately tested	—

#### 9.1.2.4 Tích hợp các phần mềm hướng đối tượng

- Cài đặt và tích hợp hướng đối tượng
  - Hầu hết các phần mềm đều cài đặt và tích hợp sandwich
  - Các đối tượng được tích hợp dưới lên
  - Các mô đun khác được tích hợp trên xuống

#### 9.1.2.5 Quản lý tích hợp

- Ví dụ:
  - Tài liệu thiết kế được sử dụng bởi người lập trình P1 (người đã viết mã đối tượng o1) chỉ ra đối tượng o1 gửi thông điệp tới đối tượng o2 với bốn tham số
  - Tài liệu thiết kế được sử dụng bởi người lập trình P2 (người đã viết mã đối tượng o2) chỉ rõ ràng rằng chỉ 3 đối số được truyền tới đối tượng o2
- Giải pháp:
  - Tiến trình tích hợp phải được quản lý bởi nhóm SQA

## 9.2 KIỂM THỬ PHA CÀI ĐẶT VÀ TÍCH HỢP

### 9.2.1 Luồng công việc kiểm thử cài đặt

- Kiểm thử đơn vị
  - Kiểm thử đơn vị không hình thức được thực hiện bởi người lập trình

- Kiểm thử đơn vị một cách cẩn thận có phương pháp bởi nhóm SQA
  - Có hai loại kiểm thử đơn vị có phương pháp
    - Kiểm thử không có sự thực thi
    - Kiểm thử dựa trên sự thực thi
  - Lựa chọn trường hợp kiểm thử
    - Cách tồi nhất – kiểm thử ngẫu nhiên
      - Không có thời gian để kiểm thử tất cả nhưng phần trăm nhỏ nhất của các trường hợp có thể kiểm thử được trên tổng số là 10% hoặc hơn
    - Chúng ta cần có một cách có hệ thống để xây dựng các trường hợp kiểm thử
- 9.2.1.1 Kiểm thử với các đặc tả so với kiểm thử với mã**
- *Kiểm thử với các đặc tả* (*Test to specifications*) (cũng được gọi là kiểm thử hướng đầu vào/đầu ra, kiểm thử hướng dữ liệu, hoặc kiểm thử chức năng hoặc kiểm thử hộp đen)
    - Lờ qua mã – sử dụng các đặc tả để lựa chọn các trường hợp kiểm thử
  - *Kiểm thử với mã* (*Test to code*) (cũng được gọi là kiểm thử hướng đường dẫn, cấu trúc, hướng lô gic, kiểm thử hộp kính)
    - Lờ đi các đặc tả - sử dụng mã để lựa chọn trường hợp kiểm thử

### Tính khả thi của việc kiểm thử với đặc tả

- Ví dụ:
  - Các đặc tả đối với một phần mềm xử lý dữ liệu gồm 5 loại nhiệm vụ và 7 (types of discount)
  - 35 trường hợp kiểm thử
- We cannot say that commission and discount are computed in two entirely separate artifacts — the structure is irrelevant
- Mục đích của các đặc tả bao gồm 20 nhân tố, mỗi nhân tố, mỗi nhân tố đảm nhiệm 4 giá trị
  - Có  $4^{20}$  hoặc  $1.1 \cdot 10^{12}$  trường hợp kiểm thử
  - Nếu mỗi nhân tố sử dụng mất 30 giây để thực thi thì việc chạy tất cả các trường hợp kiểm thử mất nhiều hơn 1 triệu năm
- Sự tăng nhanh tổ hợp làm cho kiểm thử sử dụng các đặc tả không thể thực hiện được

### Tính khả thi của kiểm thử với mã

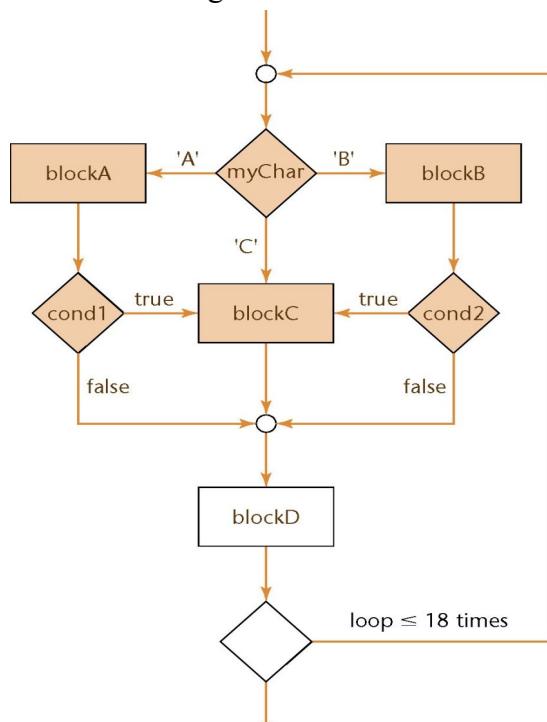
- Mỗi đường dẫn tiến trình xuyên suốt mỗi mô đun phải được thử hiện ít nhất một lần
  - Sự tăng nhanh tổ hợp (Combinatorial explosion)
- Ví dụ mã:

```

read (kmax)                                // kmax is an integer between 1 and 18
for (k = 0; k < kmax; k++) do
{
    read (myChar)                         // myChar is the character A, B, or C
    switch (myChar)
    {
        case 'A':
            blockA;
            if (cond1) blockC;
            break;
        case 'B':
            blockB;
            if (cond2) blockC;
            break;
        case 'C':
            blockC;
            break;
    }
    blockD;
}

```

- Biểu đồ luồng có hơn  $10^{12}$  đường dẫn khác nhau



Hình 9.3: Ví dụ khôi lệnh

- Kiểm thử với mã không đáng tin cậy

```

if ((x + y + z)/3 == x)
    print "x, y, z are equal in value";
else
    print "x, y, z are unequal";

```

Test case 1:      $x = 1, y = 2, z = 3$

Test case 2:      $x = y = z = 2$

- Chúng ta có thể thi hành từng đường dẫn mà không phát hiện ra hết lỗi
- Mỗi đường dẫn có thể được kiểm thử chỉ khi nó đưa ra
- Khi người lập trình bỏ sót kiểm thử với giá trị  $d = 0$  trong mã thì họ không ý thức được mức độ nguy hiểm có thể xảy ra

```

if (d == 0)
    zeroDivisionRoutine ();
else
    x = n/d;

```

(a)

$x = n/d;$

(b)

- Tiêu chuẩn (“thực hiện mọi đường dẫn”) là không đáng tin
  - Products exist for which some data exercising a given path detect a fault, and other data exercising the same path do

#### 9.2.1.2 Kỹ thuật kiểm thử đơn vị hộp đen

- Cả kiểm thử với các đặc tả và kiểm thử với mã đều không có tính khả thi
- Nghệ thuật kiểm thử:
  - Lựa chọn một tập nhỏ, có thể quản lý được của các trường hợp kiểm thử để
  - Cực đại những cơ hội phát hiện lỗi, trong khi
  - Cực tiểu những cơ hội lãng phí trường hợp kiểm thử
- Mỗi trường hợp kiểm thử phải phát hiện ra một lỗi không phát hiện được trước đó
- Chúng ta cần một phương thức làm nổi bật nên nhiều lỗi nhất có thể
  - Đầu tiên thực hiện các trường hợp kiểm thử hộp đen (Kiểm thử với các đặc tả)
  - Sau đó là các phương thức kiểm thử hộp kính (kiểm thử với mã)
- a. Kiểm thử tương đương và phân tích các giá trị biên
- Ví dụ

- Các đặc tả cho DBMS chỉ ra rằng sản phẩm phần mềm phải xử lý một số lượng bản ghi nằm trong khoảng từ 1 đến 16,383 ( $2^{14} - 1$ )
- Nếu hệ thống có thể xử lý trong 34 bản ghi và 14,870 bản ghi, thì nó có thể làm việc tốt với 8,252 bản ghi
- Nếu hệ thống làm việc với bất cứ trường hợp kiểm thử nào nằm trong khoảng (1... 16,383) If the , thì nó có thể làm việc tốt với bất cứ trường hợp kiểm thử nào thuộc khoảng đó
  - Dãy (1..16,383) tạo thành lớp tương đương
- Kiểm thử tương đương
  - Bất cứ thành viên nào của lớp tương đương cũng là một trường hợp kiểm thử tốt bằng các thành viên khác của lớp tương đương
  - Dãy (1..16,383) xác định ra ba lớp tương đương:
    - Lớp tương đương 1: Ít hơn 1 bản ghi
    - Lớp tương đương 2: Giữa 1 và 16,383 bản ghi
    - Lớp tương đương thứ 3: Nhiều hơn 16,383 bản ghi
- Phân tích các giá trị biên
 

Lựa chọn trường hợp kiểm thử chỉ dựa trên biên của các lớp tương đương

  - Điều này làm tăng nhanh xác suất phát hiện lỗi

Ví dụ cơ sở dữ liệu

  - Trường hợp kiểm thử 1: 0 bản ghi, thành viên của lớp tương đương 1 và kè sát với giá trị biên
  - Trường hợp kiểm thử 2: 1 bản ghi, giá trị biên
  - Trường hợp kiểm thử 3: 2 bản ghi, kè sát với giá trị biên
  - Trường hợp kiểm thử 4: 723 bản ghi là thành viên của lớp tương đương 2
  - Trường hợp kiểm thử 5: 16,382 giá trị bản ghi, kè sát với giá trị biên
  - Trường hợp kiểm thử 6: 16,383 giá trị bản ghi, giá trị biên
  - Trường hợp kiểm thử 7: 16,384 giá trị bản ghi, là thành viên thuộc lớp tương đương 3 và kè sát với giá trị biên
- Kiểm thử tương đương của các đặc tả đầu ra
  - Chúng ta cũng cần thực hiện kiểm thử tương đương các đặc tả đầu ra
  - Ví dụ:
    - Năm 2006, In 2006, the minimum Social Security (OASDI) deduction from any one paycheck was \$0.00, and the maximum was \$5,840.40
    - Test cases must include input data that should result in deductions of exactly \$0.00 and exactly \$5,840.40

- Also, test data that might result in deductions of less than \$0.00 or more than \$5,840.40
- Chiến lược tổng quan
  - Các lớp tương đương sử dụng chung phân tích giá trị biên để kiểm thử cả đặc tả đầu vào và đặc tả đầu ra
    - Phương pháp này sinh ra một tập nhỏ dữ liệu kiểm thử với khả năng phát hiện ra một số lượng lớn lỗi
- b. Kiểm thử chức năng
  - Một kiểu khác của kiểm thử hộp đen đối với phần mềm cổ điển
    - Dữ liệu kiểm thử dựa trên những chức năng của các mô đun
    - Mỗi mục của chức năng hoặc chức năng được xác định
    - Dữ liệu kiểm thử được nghĩ ra để kiểm thử chức năng ở mức thấp hơn một cách tách biệt
  - Sau đó, các chức năng mức cao đã kết hợp với các chức năng ở mức thấp được kiểm thử
  - Tuy nhiên, trong thực tế
    - Không phải các chức năng mức cao hơn luôn luôn được xây dựng tách biệt khỏi những chức năng mức thấp hơn bằng cách sử dụng cấu trúc của lập trình cấu trúc( Higher-level functions are not always neatly constructed out of lower-level functions using the constructs of structured programming)
    - Thay vì đó, chức năng mức thấp hơn thường được đan xen vào
  - Cũng như thế, các biên về mặt chức năng không phải luôn luôn trùng khớp với biên của các mô đun mã
    - Sự phân biệt giữa kiểm thử đơn vị và kiểm thử tích hợp trở thành lu mờ
    - Vấn đề này cũng có thể nảy sinh trong mô hình hướng đối tượng khi thông điệp được truyền giữa hai đối tượng
  - Mỗi quan hệ qua lại ngẫu nhiên giữa các mô đun mã có thể dẫn đến kết quả tiêu cực trong quản lý
    - Các mốc quan trọng và thời hạn cuối cùng có thể trở nên không rõ ràng
    - Sau đó rất khó để xác định trạng thái của dự án

#### 9.2.1.3 Kỹ thuật kiểm thử đơn vị hộp kính

- Phủ dòng lệnh (statement coverage)
- Phủ nhánh(Branch coverage)
- Phủ đường dẫn (Path coverage)
- Chuỗi mã tuyển tính
- Phủ đường dẫn sử dụng tất cả các định nghĩa (All-definition-use path coverage)
  - a. Kiểm thử cấu trúc: phủ dòng lệnh, nhánh và đường dẫn

### Phủ dòng lệnh

- Thực hiện tập trường hợp kiểm thử trong đó mỗi dòng lệnh được thực hiện ít nhất một lần
  - Công cụ CASE cần được sử dụng để kiểm tra
  - Nhược điểm
    - Câu lệnh rẽ nhánh
    - Cả hai câu lệnh đều được thực thi mà không chỉ ra lỗi
- if** ( $s > 1 \ \&\& t == 0$ )  
 $x = 9;$

Test case:  $s = 2, t = 0.$

### Phủ nhánh

- Việc thực hiện một tập các trường hợp kiểm thử trong đó mỗi nhánh được thực hiện ít nhất một lần (cũng như tất cả các câu lệnh)
  - Điều này giải quyết vấn đề ở phủ dòng lệnh
  - Công cụ CASE là cần thiết

### Phủ đường dẫn

- Thực hiện tập các trường hợp kiểm thử trong đó mỗi đường dẫn được thực hiện ít nhất một lần (cũng như tất cả các câu lệnh)
- Vấn đề:
  - Số lượng của các đường dẫn có thể rất lớn
  - Chúng ta muốn một điều kiện ít thực hiện ít đường dẫn hơn nhưng lại chỉ ra được nhiều lỗi hơn phủ nhánh

### Chuỗi mã truyền tính

- Xác định ra tập các điểm L mà từ các điểm đó luồng điều khiển có thể nhảy đến một vị trí nào đó, cộng các đầu mục và thoát khỏi các điểm (Identify the set of points L from which control flow may jump, plus entry and exit points)
- Hạn chế các trường hợp kiểm thử so với phủ đường dẫn bằng cách bắt đầu và kết thúc với các thành phần của L
- Phương pháp này phát hiện ra nhiều lỗi mà không phải kiểm thử mọi đường dẫn
- *Phủ đường dẫn sử dụng tất cả các định nghĩa*
- Mỗi biến cố của biến zz được gán nhãn hoặc là:
  - Định nghĩa của một biến (The definition of a variable)  
 $zz = 1$  or read (zz)
  - Hoặc sự sử dụng của một biến (or the use of variable)  
 $y = zz + 3$  or if (zz < 9) errorB ()

- Xác định tất cả các đường dẫn từ sự định nghĩa của một biến tới sự sử dụng của định nghĩa đó
- Điều này có thể được thực hiện bằng một công cụ tự động
- Mỗi trường hợp kiểm thử được thiết lập cho mỗi đường dẫn như vậy
- Bất lợi:
  - Với  $d$  nhánh thì có trên  $2^d$  số lượng đường dẫn
- (Upper bound on number of paths is  $2^d$ , where  $d$  is the number of branches)
- Trong thực tế:
  - Số lượng đường dẫn thực tế tương ứng với  $d$
- Do đó đây là kỹ thuật lựa chọn trường hợp kiểm thử thực tế
- Không thể kiểm thử một câu lệnh cụ thể
  - Chúng ta có thể có một đường dẫn không khả thi (“mã chết”) trong mô đun
- Thường đây là dấu hiệu của lỗi
 

```
if ( $k < 2$ )
{
    if ( $k > 3$ )           [should be k > -3]
    ↑
     $x = x * k;$ 
}
```

(a)

```
for ( $j = 0; j < 0; j++$ ) [should be j < 10]
↑
total = total + value[j];

```

(b)

- b. Các thước đo độ phức tạp
- Là một phương pháp đảm bảo chất lượng để kiểm thử hộp kính
  - Mô đun m1 phức tạp hơn mô đun m2
    - Về mặt trực quan, m1 có khả năng sinh ra nhiều lỗi hơn mô đun m2
  - Nếu độ phức tạp vượt quá mức độ cho phép thì nên thiết kế lại và sau đó viết mã lại thì mô đun viết mã
    - Rẻ hơn và nhanh hơn việc cố gắng sửa lỗi mô đun viết mã có thể xảy ra lỗi

#### Số lượng dòng mã

- Thước đo đơn giản nhất của độ phức tạp
  - Giả định cơ bản: có một xác suất một dòng mã chứa lỗi là  $p$
- Ví dụ
  - Người kiểm thử tin tưởng mỗi dòng mã có 2% khả năng sinh ra lỗi.

- Nếu tài liệu mã kiểm thử có 200 dòng lệnh thì có thể chứa 2 lỗi
- Thực vậy, số lượng lỗi liên quan tới kích cỡ của toàn bộ sản phẩm

#### Các thước đo khác để đo độ phức tạp

- Cyclomatic complexity  $M$  (McCabe)
  - Số lượng các điểm quyết định (các nhánh) trong mô đun mã
  - Dễ dàng tính toán
  - Thước đo tốt của các lỗi (xem slide sau)
- Trong 1 thử nghiệm, các mô đun mã với  $M > 10$  đã thông chỉ ra nhiều lỗi hơn về mặt thống kê

#### Vấn đề với thước đo độ phức tạp

- Thước đo độ phức tạp, đặc biệt là cyclomatic complexity, đã trải qua những thử thách trong
  - Lý thuyết
  - Thủ nghiệm và
  - Có liên quan nhiều với LOC
- Về bản chất, chúng ta đang đo số lượng dòng mã, không phải độ phức tạp
- Rà soát mã sẽ phát hiện lỗi nhanh và kỹ lưỡng
  - Giảm tới 95% chi phí bảo trì

#### 9.2.1.4 So sánh các kỹ thuật kiểm thử đơn vị

- So sánh các thử nghiệm
  - Kiểm thử hộp đen
  - Kiểm thử hộp kính
  - Các kiểu rà soát
  - [Myers, 1978] 59 lập trình viên có kinh nghiệm tốt
  - Cả ba phương pháp đều có hiệu quả bằng nhau trong việc phát hiện lỗi
  - Rà soát kỹ lưỡng mã có hiệu năng về chi phí thấp (Code inspections were less cost-effective)
- [Hwang, 1981]
  - Cả ba phương pháp đều có hiệu quả bằng nhau
- [Basili and Selby, 1987] 42 sinh viên tiên tiến trong 2 nhóm, 32 lập trình viên chuyên nghiệp
  - Những sinh viên tiên tiến ở nhóm 1
  - Không có sự khác nhau đáng kể trong ba phương pháp kiểm thử
  - Những sinh viên tiên tiến ở nhóm 2
  - Rà soát mã và kiểm thử hộp đen tốt bằng nhau
  - Cả hai việc trên làm tốt hơn kiểm thử hộp kính
- Những người lập trình chuyên nghiệp

- Rà soát mã phát hiện được nhiều lỗi hơn
- Rà soát mã có tốc độ phát hiện lỗi nhanh hơn
- Kết luận
  - Rà soát kỹ lưỡng mã ít nhất cũng phát hiện được số lượng lỗi bằng với kiểm thử hộp đen và hộp kính

#### 9.2.1.5 Cleanroom

- Là một phương pháp tiếp cận khác đối với phát triển phần mềm
- Hợp nhất
  - Mô hình tiến trình tăng
  - Các kỹ thuật hình thức
  - Các kiểu rà soát
- Prototype automated documentation system for the U.S. Naval Underwater Systems Center
- 1820 dòng của FoxBASE
  - 18 lỗi được phát hiện bởi “xác minh chức năng”
  - Kiểm tra không hình thức được sử dụng
  - 19 lỗi được phát hiện bằng rà soát lượt qua trước khi biên dịch
  - Không có lỗi biên dịch
  - Không có sự thất bại ở thời gian thực thi

Sự khác nhau trong việc tính toán tỷ lệ lỗi kiểm thử:

- Các mô hình thông thường:
  - Đếm số lỗi sau khi kiểm thử không hình thức được hoàn thành (Khi SQA bắt đầu)
- Cleanroom
  - Đếm số lỗi sau khi rà soát kỹ lưỡng được hoàn thành (khi biên dịch được bắt đầu )

Báo cáo về 17 sản phẩm phần mềm Cleanroom

- Hệ điều hành
  - 350,000 LOC
  - Phát triển trong 18 tháng
  - Bởi một đội 70 người
  - Tỷ lệ lỗi kiểm thử chỉ là 1.0 lỗi trên KLOC
- Các loại phần mềm khác nhau với tổng số 1 triệu LOC
  - Tỷ lệ lỗi kiểm thử trung bình có đánh trọng số: 2.3 lỗi trên KLOC
- “[R]emarkable quality achievement”

#### 9.2.1.6 Những vấn đề có khả năng xảy ra khi kiểm thử các đối tượng

- Phải kiểm tra kỹ lưỡng các lớp và các đối tượng
- Có thể chạy các trường hợp kiểm thử đối với các đối tượng (nhưng không đối với các lớp)
- Một mô đun có điển hình :

- Gồm khoảng 50 câu lệnh có thể thực thi
- Sinh ra các tham số đầu vào, kiểm tra các tham số đầu ra
- Các đối tượng điển hình:
  - Gồm khoảng 30 phương thức, mỗi phương thức chỉ với 2 hoặc 3 câu lệnh
  - Một phương thức thường không trả lại một giá trị tới người gọi – thay vào đó thì nó thay đổi trạng thái
  - Nó không thể kiểm tra trạng thái bởi vì sự ẩn giấu thông tin
  - Ví dụ: phương thức determineBalance —cần biết trước accountBalance
- Chúng ta cần thêm vào các phương thức để trả lại các giá trị của các biến trạng thái
  - Đó là một phần của kế hoạch kiểm thử
  - Biến dịch điều kiện có thể phải được sử dụng (Conditional compilation may have to be used)
- Các phương thức đã kế thừa có thể vẫn phải được kiểm thử (xem ví dụ sau đây)

Ví dụ cài đặt java của một cây phân cấp

```

class RootedTreeClass
{
    ...
    void displayNodeContents (Node a);
    void printRoutine (Node b);
    //
    // method displayNodeContents uses method printRoutine
    //
    ...
}

class BinaryTreeClass extends RootedTreeClass
{
    ...
    void displayNodeContents (Node a);
    //
    // method displayNodeContents defined in this class uses
    // method printRoutine inherited from ClassRootedTree
    //
    ...
}

class BalancedBinaryTreeClass extends BinaryTreeClass
{
    ...
    void printRoutine (Node b);
    //
    // method displayNodeContents (inherited from BinaryTreeClass) uses this
    // local version of printRoutine within class BalancedBinaryTreeClass
    //
    ...
}

```

Nửa trên Khi displayNodeContents được gọi trong **BinaryTreeClass**, nó sử dụng uses **RootedTreeClass.printRoutine**

```

class RootedTreeClass
{
    ...
    void displayNodeContents (Node a);
    void printRoutine (Node b);
    //
    // method displayNodeContents uses method printRoutine
    //
    ...
}

class BinaryTreeClass extends RootedTreeClass
{
    ...
    void displayNodeContents (Node a);
    //
    // method displayNodeContents defined in this class uses
    // method printRoutine inherited from ClassRootedTree
    //
    ...
}

```

Nửa dưới khi **displayNodeContents** được gọi trong **BalancedBinaryTreeClass**, nó sử dụng **BalancedBinaryTreeClass.printRoutine**

```

class BinaryTreeClass extends RootedTreeClass
{
    ...
    void displayNodeContents (Node a);
    //
    // method displayNodeContents defined in this class uses
    // method printRoutine inherited from ClassRootedTree
    //
    ...
}

class BalancedBinaryTreeClass extends BinaryTreeClass
{
    ...
    void printRoutine (Node b);
    //
    // method displayNodeContents (inherited from BinaryTreeClass) uses this
    // local version of printRoutine within class BalancedBinaryTreeClass
    //
    ...
}

```

- Tin xấu
  - **BinaryTreeClass.displayNodeContents** phải được kiểm thử lại từ ban đầu khi sử dụng **BalancedBinaryTreeClass**
  - Nó gọi một phiên bản khác của **printRoutine**
- Tin xấu hơn
  - Với những lý do lý thuyết, cần sử dụng toàn bộ những trường hợp kiểm thử khác nhau để kiểm thử

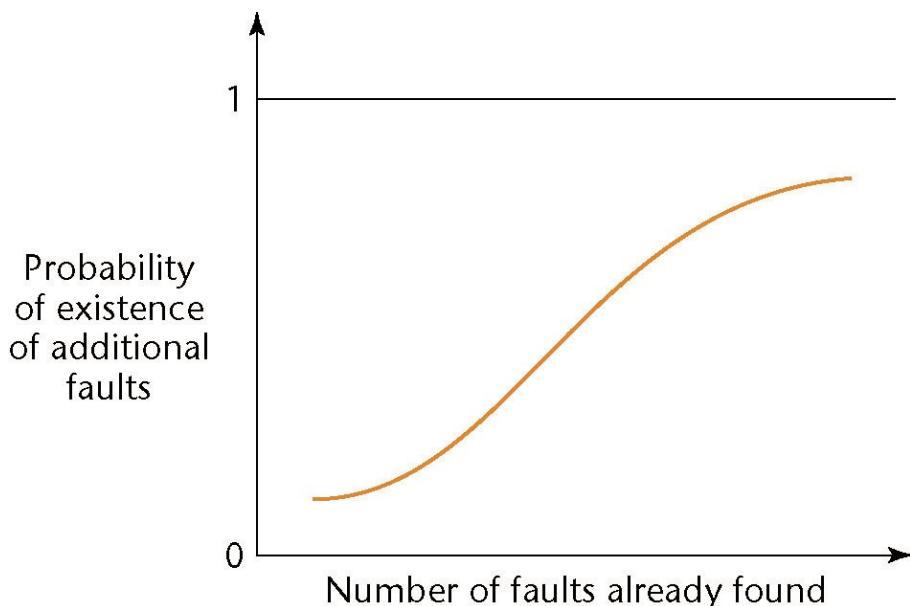
- Làm cho các biến trạng thái có thể thấy được
  - Vấn đề nhỏ
- Việc kiểm thử lại trước khi sử dụng lại
  - Chỉ xuất hiện khi các phương thức tương tác
  - Chúng ta có thể xác định khi nào cần kiểm thử lại
- Đây không phải là những lý do để từ bỏ mô hình hướng đối tượng

#### 9.2.1.7 Các khía cạnh quản lý của kiểm thử đơn vị

- Cần biết khi nào dừng kiểm thử
- Các kỹ thuật khác nhau có thể được sử dụng
  - Phân tích chi phí – lợi nhuận
  - Phân tích rủi ro
  - Các kỹ thuật thống kê

#### 9.2.1.8 Khi nào viết lại hơn là gỡ lỗi

- Khi mô đun mã có quá nhiều lỗi
  - Thiết kế lại và viết mã lại rẻ hơn



Hình 9.4: Quan hệ giữa rủi ro và chi phí các lỗi thêm

- Rủi ro và chi phí của các lỗi thêm nữa sẽ rất lớn
- Với mỗi mô đun, việc quản lý phải xác định trước số lượng lỗi lớn nhất được cho phép trong suốt quá trình kiểm thử
- Nếu con số này đạt tới thì
  - Bỏ qua
  - Thiết kế lại

- Viết mã lại
- Số lượng lỗi lớn nhất được cho phép sau khi chuyển giao là 0

*Sự phân bố của lỗi trong mô đun là không đồng nhất*

- [Myers, 1979]
  - 47% lỗi trong OS/370 chỉ nằm ở 4% các mô đun
- [Endres, 1975]
  - 512 lỗi nằm trong 202 mô đun của DOS/VS (phát hành lần thứ 28)
  - 112 mô đun chỉ có 1 lỗi
  - Có các mô đun chỉ có 14, 15, 19 và 28 lỗi
  - Mô đun với 28 lỗi là mô đun lớn nhất trong phần mềm, với hơn 3000 dòng lệnh của ngôn ngữ hợp ngữ macro DOS (The latter three were the largest modules in the product, with over 3000 lines of DOS macro assembler language )
  - Mô đun với 14 lỗi là mô đun tương đối nhỏ và rất bất định
  - Giải pháp để ra là bỏ qua, thiết kế lại, viết mã lại

### 9.2.2 Kiểm thử tích hợp

- Là việc kiểm thử khi có một mô đun mã mới được thêm vào nhóm các mô đun đã được kiểm thử
- Vấn đề đặc biệt có thể nảy sinh khi kiểm thử giao diện người dùng

*Kiểm thử tích hợp giao diện người dùng*

- Các trường hợp kiểm thử GUI bao gồm
  - Những cú nhấp chuột, và
  - Nhấn phím
- Những kiểu trường hợp kiểm thử này không thể được lưu giữ theo cách thông thường
  - Yêu cầu các công cụ CASE đặc biệt
- Ví dụ:
  - QAPartner
  - XRunner

## 9.3 KIỂM THỬ SẢN PHẨM

- Kiểm thử sản phẩm cho phần mềm COTS
  - Kiểm thử Alpha, beta
- Kiểm thử sản phẩm đối với phần mềm tùy chỉnh
  - Nhóm SQA phải đảm bảo rằng phần mềm chuyển qua kiểm thử chấp nhận
  - Thất bại kiểm thử chấp nhận gây ra tác động xấu đến tổ chức phát triển

*Kiểm thử sản phẩm đối với phần mềm tùy chỉnh*

- Đội SQA phải cố gắng thực hiện kiểm thử gần giống với kiểm thử chấp nhận

- Thực hiện các trường hợp kiểm thử hộp đen đối với toàn sản phẩm phần mềm
- Tính mạnh mẽ của toàn bộ sản phẩm
  - *Stress testing* (under peak load)
  - *Volume testing* (e.g., can it handle large input files?)
- Tất cả những ràng buộc phải được kiểm tra
- Tất cả tài liệu phải được
  - Kiểm tra tính chính xác
  - Kiểm tra sự tương thích với chuẩn
  - Xác minh lại với phiên bản hiện thời của phần mềm
- Sản phẩm phần mềm (tài liệu và mã) được chuyển giao cho tổ chức khách hàng để thực hiện kiểm thử chấp nhận

#### 9.4 KIỂM THỬ CHẤP NHẬN

- Khách hàng xác định liệu phần mềm thoả mãn những đặc tả của nó
- Kiểm thử chấp nhận được thực hiện bởi
  - Tổ chức khách hàng, hoặc
  - Đội SQA cùng với sự có mặt của đại diện của khách hàng, hoặc
  - Đội SQA độc lập được khách hàng thuê
- Bốn thành phần chính của kiểm thử chấp nhận là
  - Tính chính xác
  - Tính mạnh mẽ
  - Hiệu năng
  - Tài liệu
- Những thành phần này được kiểm thử cẩn thận bởi người phát triển trong suốt quá trình kiểm thử sản phẩm
- Sự khác nhau giữa kiểm thử sản phẩm và kiểm thử chấp nhận là:
  - Kiểm thử chấp nhận được thực hiện trên dữ liệu thực
  - Kiểm thử sản phẩm được thực hiện trên dữ liệu thử nghiệm, là những cái được định nghĩa không có thực

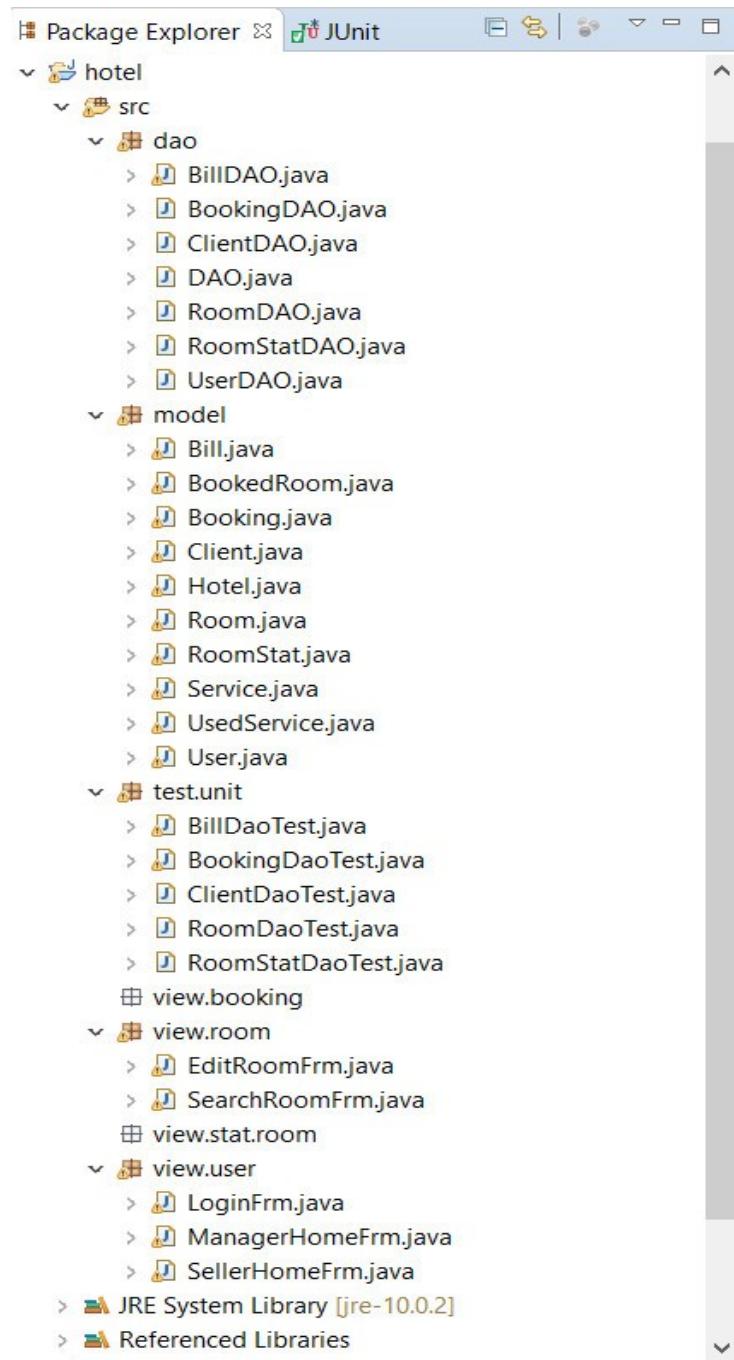
#### 9.5 CASE STUDY CHO PHA CÀI ĐẶT

Nội dung phần này sẽ trình bày quá trình cài đặt và kiểm thử cho các modul đã trình bày trong pha thiết kế của phần mềm quản lý đặt phòng khách sạn:

- Modul sửa thông tin một phòng khách sạn
- Modul đặt phòng khách sạn
- Modul xem thống kê phòng theo doanh thu

### 9.5.1 Cài đặt

#### a. Tổ chức thư mục dự án



Hình 9.5: Cấu trúc tổ chức thư mục của dự án trong Eclipse

Cấu trúc thư mục của dự án được tổ chức như cấu trúc mô tả trong Hình 9.5:

- model: package này chứa toàn bộ các lớp thực thể
- dao: package này chứa toàn bộ các lớp DAO
- test.unit: package này chứa toàn bộ các lớp JUnit test tương ứng với các lớp DAO trong package dao.
- view.room: package này chứa các lớp biên liên quan đến chức năng quản lý phòng
- view.user: package này chứa các lớp biên liên quan đến chức năng quản lý người dùng
- view.booking: package chứa các lớp biên liên quan đến chức năng đặt phòng (các lớp này coi như bài tập cho sinh viên)
- view.stat.room: package này chứa các lớp liên quan đến chức năng thống kê phòng (các lớp này coi như bài tập cho sinh viên)

### b. Code modul sửa thông tin phòng

+ Lớp Room.java:

```
package model;
import java.io.Serializable;

public class Room implements Serializable{
    private int id;
    private String name;
    private String type;
    private float price;
    private String des;

    public Room() {
        super();
    }
    public Room(String name, String type, float price, String des) {
        super();
        this.name = name;
        this.type = type;
        this.price = price;
        this.des = des;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```

        this.name = name;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public float getPrice() {
        return price;
    }
    public void setPrice(float price) {
        this.price = price;
    }
    public String getDes() {
        return des;
    }
    public void setDes(String des) {
        this.des = des;
    }
}

```

+ Lớp User.java:

```

package model;
import java.io.Serializable;

public class User implements Serializable{
    private int id;
    private String username;
    private String password;
    private String name;
    private String position;

    public User() {
        super();
    }
    public User(String username, String password, String name, String position) {
        super();
        this.username = username;
        this.password = password;
        this.name = name;
        this.position = position;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
}

```

```

public void setUsername(String username) {
    this.username = username;
}
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getPosition() {
    return position;
}
public void setPosition(String position) {
    this.position = position;
}
}

```

+ Lớp DAO.java:

```

package dao;
import java.sql.Connection;
import java.sql.DriverManager;

public class DAO {
    public static Connection con;

    public DAO(){
        if(con == null){
            String dbUrl =
                "jdbc:mysql://localhost:3307/hotel?autoReconnect=true&useSSL=false";
            String dbClass = "com.mysql.jdbc.Driver";
            try {
                Class.forName(dbClass);
                con = DriverManager.getConnection (dbUrl, "root", "xxx");
            }catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
}

```

+ Lớp UserDAO.java:

```

package dao;
import java.sql.PreparedStatement;

```

```

import java.sql.ResultSet;
import model.User;

public class UserDAO extends DAO{

    public UserDAO() {
        super();
    }

    public boolean checkLogin(User user) {
        boolean result = false;
        String sql = "SELECT position FROM tblUser WHERE username = ? AND
                     password = ?";
        try {
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, user.getUsername());
            ps.setString(2, user.getPassword());
            ResultSet rs = ps.executeQuery();
            if(rs.next()) {
                user.setName(rs.getString("name"));
                user.setPosition(rs.getString("position"));
                result = true;
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
        return result;
    }
}

```

+ Lớp RoomDAO.java:

```

package dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import model.Room;

public class RoomDAO extends DAO{

    public RoomDAO() {
        super();
    }

    /**
     * search all rooms in the tblRoom whose name contains the @key
     * @param key
     * @return list of room whose name contains the @key
     */
    public ArrayList<Room> searchRoom(String key){
        ArrayList<Room> result = new ArrayList<Room>();

```

```

String sql = "SELECT * FROM tblRoom WHERE name LIKE ?";
try{
    PreparedStatement ps = con.prepareStatement(sql);
    ps.setString(1, "%" + key + "%");
    ResultSet rs = ps.executeQuery();

    while(rs.next()){
        Room rm = new Room();
        rm.setId(rs.getInt("id"));
        rm.setName(rs.getString("name"));
        rm.setType(rs.getString("type"));
        rm.setPrice(rs.getFloat("price"));
        rm.setDes(rs.getString("des"));
        result.add(rm);
    }
} catch(Exception e){
    e.printStackTrace();
}
return result;
}

/**
 * update the @room
 * @param rm
 */
public boolean updateRoom(Room rm){
    String sql = "UPDATE tblroom SET name=?, type=?, price=?, des=?"
                + " WHERE id=?";
    try{
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, rm.getName());
        ps.setString(2, rm.getType());
        ps.setFloat(3, rm.getPrice());
        ps.setString(4, rm.getDes());
        ps.setInt(5, rm.getId());

        ps.executeUpdate();
    } catch(Exception e){
        e.printStackTrace();
        return false;
    }
    return true;
}
}

```

+ Lớp LoginFrm.java

```

package view.user;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

```

```

import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

import dao.UserDAO;
import model.User;

public class LoginFrm extends JFrame implements ActionListener{
    private JTextField txtUsername;
    private JPasswordField txtPassword;
    private JButton btnLogin;

    public LoginFrm(){
        super("Login");
        txtUsername = new JTextField(15);
        txtPassword = new JPasswordField(15);
        txtPassword.setEchoChar('*');
        btnLogin = new JButton("Login");

        JPanel pnMain = new JPanel();
        pnMain.setSize(this.getSize().width-5, this.getSize().height-20);
        pnMain.setLayout(new BoxLayout(pnMain,BoxLayout.PAGE_AXIS));
        pnMain.add(Box.createRigidArea(new Dimension(0,10)));

        JLabel lblHome = new JLabel("Login");
        lblHome.setAlignmentX(Component.CENTER_ALIGNMENT);
        lblHome.setFont (lblHome.getFont ().deriveFont (20.0f));
        pnMain.add(lblHome);
        pnMain.add(Box.createRigidArea(new Dimension(0,20)));

        JPanel pnUsername = new JPanel();
        pnUsername.setLayout(new FlowLayout());
        pnUsername.add(new JLabel("Username:"));
        pnUsername.add(txtUsername);
        pnMain.add(pnUsername);

        JPanel pnPass = new JPanel();
        pnPass.setLayout(new FlowLayout());
        pnPass.add(new JLabel("Password:"));
        pnPass.add(txtPassword);
        pnMain.add(pnPass);;

        pnMain.add(btnLogin);
        pnMain.add(Box.createRigidArea(new Dimension(0,10)));
        btnLogin.addActionListener(this);

        this.setSize(400,200);
        this.setLocation(200,10);
        this.setContentPane(pnMain);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {

```

```

        if((e.getSource() instanceof JButton)
           &&(((JButton)e.getSource()).equals(btnLogin))) {
            User user = new User();
            user.setUsername(txtUsername.getText());
            user.setPassword(txtPassword.getText());

            UserDAO ud = new UserDAO();
            if(ud.checkLogin(user)) {
                if(user.getPosition().equalsIgnoreCase("manager")) {
                    (new ManagerHomeFrm()).setVisible(true);
                    this.dispose();
                }else if(user.getPosition().equalsIgnoreCase("seller")) {
                    (new SellerHomeFrm()).setVisible(true);
                    this.dispose();
                }else
                    JOptionPane.showMessageDialog(this,
                        "The function of the role " + user.getPosition()
                        + " is under construction!");
            }else {
                JOptionPane.showMessageDialog(this,
                    "Incorrect username and/or password!");
            }
        }
    }

    public static void main(String[] args) {
        LoginFrm myFrame = new LoginFrm();
        myFrame.setVisible(true);
    }
}

```

+ Lớp ManagerHomeFrm.java:

```

package view.user;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
import javax.swingBoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import model.User;
import view.room.ManageRoomFrm;

public class ManagerHomeFrm extends JFrame implements ActionListener{
    private JButton btnHotel, btnRoom, btnStat;
    private User user;

```

```

public ManagerHomeFrm(User user) {
    super("Manager home");
    this.user = user;

    JPanel listPane = new JPanel();
    listPane.setLayout(new BoxLayout(listPane, BoxLayout.PAGE_AXIS));

    JPanel lblPane = new JPanel();
    lblPane.setLayout(new BoxLayout(lblPane, BoxLayout.LINE_AXIS));
    lblPane.add(Box.createRigidArea(new Dimension(450, 0)));
    JLabel lblUser = new JLabel("Logged in as: " + user.getName());
    lblUser.setAlignmentX(Component.RIGHT_ALIGNMENT);
    lblPane.add(lblUser);
    listPane.add(lblPane);
    listPane.add(Box.createRigidArea(new Dimension(0,20)));

    JLabel lblHome = new JLabel("Manager's home");
    lblHome.setAlignmentX(Component.CENTER_ALIGNMENT);
    lblHome.setFont (lblHome.getFont ().deriveFont (28.0f));
    listPane.add(lblHome);
    listPane.add(Box.createRigidArea(new Dimension(0,20)));

    btnHotel = new JButton("Hotel management");
    btnHotel.setAlignmentX(Component.CENTER_ALIGNMENT);
    btnHotel.addActionListener(this);
    listPane.add(btnHotel);
    listPane.add(Box.createRigidArea(new Dimension(0,10)));

    btnRoom = new JButton("Room management");
    btnRoom.setAlignmentX(Component.CENTER_ALIGNMENT);
    btnRoom.addActionListener(this);
    listPane.add(btnRoom);
    listPane.add(Box.createRigidArea(new Dimension(0,10)));

    btnStat = new JButton("View statistic");
    btnStat.setAlignmentX(Component.CENTER_ALIGNMENT);
    btnStat.addActionListener(this);
    listPane.add(btnStat);

    this.setSize(600,300);
    this.setLocation(200,10);
    this.add(listPane, BorderLayout.CENTER);
    this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if((e.getSource() instanceof JButton)&&
        (((JButton)e.getSource()).equals(btnRoom))) {
        (new ManageRoomFrm(user)).setVisible(true);
        this.dispose();
    }else {
        JOptionPane.showMessageDialog(this,
            "This function is under construction!");
    }
}

```

```

        }
    }
}
```

+ Lớp ManageRoomFrm.java:

```

package view.room;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
import javax.swingBoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import model.User;

public class ManageRoomFrm extends JFrame implements ActionListener{
    private JButton btnAdd, btnEdit, btnDel;
    private User user;

    public ManageRoomFrm(User user) {
        super("Room management");
        this.user = user;

        JPanel listPane = new JPanel();
        listPane.setLayout(new BoxLayout(listPane, BoxLayout.PAGE_AXIS));

        JPanel lblPane = new JPanel();
        lblPane.setLayout(new BoxLayout(lblPane, BoxLayout.LINE_AXIS));
        lblPane.add(Box.createRigidArea(new Dimension(450, 0)));
        JLabel lblUser = new JLabel("Logged in as: " + user.getName());
        lblUser.setAlignmentX(Component.RIGHT_ALIGNMENT);
        lblPane.add(lblUser);
        listPane.add(lblPane);
        listPane.add(Box.createRigidArea(new Dimension(0,20)));

        JLabel lblHome = new JLabel("Room management");
        lblHome.setAlignmentX(Component.CENTER_ALIGNMENT);
        lblHome.setFont (lblHome.getFont ().deriveFont (28.0f));
        listPane.add(lblHome);
        listPane.add(Box.createRigidArea(new Dimension(0,20)));

        btnAdd = new JButton("Add room");
        btnAdd.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnAdd.addActionListener(this);
        listPane.add(btnAdd);
        listPane.add(Box.createRigidArea(new Dimension(0,10)));
    }
}
```

```

        btnEdit = new JButton("Edit room");
        btnEdit.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnEdit.addActionListener(this);
        listPane.add(btnEdit);
        listPane.add(Box.createRigidArea(new Dimension(0,10)));

        btnDel = new JButton("Delete room");
        btnDel.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnDel.addActionListener(this);
        listPane.add(btnDel);

        this.setSize(600,300);
        this.setLocation(200,10);
        this.add(listPane, BorderLayout.CENTER);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        if((e.getSource() instanceof JButton)&&
           (((JButton)e.getSource()).equals(btnEdit))) {
            (new SearchRoomFrm(user)).setVisible(true);
            this.dispose();
        }else {
            JOptionPane.showMessageDialog(this,
                "This function is under construction!");
        }
    }
}

```

+ Lớp SearchRoomFrm.java:

```

package view.room;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import javax.swing.Box;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import dao.RoomDAO;
import model.Room;

```

```

import model.User;

public class SearchRoomFrm extends JFrame implements ActionListener{
    private ArrayList<Room> listRoom;
    private JTextField txtKey;
    private JButton btnSearch;
    private JTable tblResult;
    private User user;
    private SearchRoomFrm mainFrm;

    public SearchRoomFrm(User user){
        super("Search room to edit");
        this.user = user;
        mainFrm = this;
        listRoom = new ArrayList<Room>();

        JPanel pnMain = new JPanel();
        pnMain.setSize(this.getSize().width-5, this.getSize().height-20);
        pnMain.setLayout(new BoxLayout(pnMain,BoxLayout.Y_AXIS));
        pnMain.add(Box.createRigidArea(new Dimension(0,10)));

        JLabel lblHome = new JLabel("Search a room to edit");
        lblHome.setAlignmentX(Component.CENTER_ALIGNMENT);
        lblHome.setFont (lblHome.getFont ().deriveFont (20.0f));
        pnMain.add(lblHome);
        pnMain.add(Box.createRigidArea(new Dimension(0,20)));

        JPanel pn1 = new JPanel();
        pn1.setLayout(new BoxLayout(pn1,BoxLayout.X_AXIS));
        pn1.setSize(this.getSize().width-5, 20);
        pn1.add(new JLabel("Room name: "));
        txtKey = new JTextField();
        pn1.add(txtKey);
        btnSearch = new JButton("Search");
        btnSearch.addActionListener(this);
        pn1.add(btnSearch);
        pnMain.add(pn1);
        pnMain.add(Box.createRigidArea(new Dimension(0,10)));

        JPanel pn2 = new JPanel();
        pn2.setLayout(new BoxLayout(pn2,BoxLayout.Y_AXIS));
        tblResult = new JTable();
        JScrollPane scrollPane= new JScrollPane(tblResult);
        tblResult.setFillsViewportHeight(false);
        scrollPane.setPreferredSize(new
            Dimension(scrollPane.getPreferredSize().width, 250));

        tblResult.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent e) {
                int column = tblResult.getColumnModel().
                    getColumnIndexAtX(e.getX()); // get the coloum of the button
                int row = e.getY()/tblResult.getRowHeight(); // get row

                // *Checking the row or column is valid or not
                if (row < tblResult.getRowCount() && row >= 0 &&

```

```

        column < tblResult.getColumnCount() && column >= 0) {
            (new EditRoomFrm(user,
                listRoom.get(row))).setVisible(true);
            mainFrm.dispose();
        }
    });
}

pn2.add(scrollPane);
pnMain.add(pn2);
this.add(pnMain);
this.setSize(600,300);
this.setLocation(200,10);
this.setVisible(true);
this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
}

@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    JButton btnClicked = (JButton)e.getSource();
    if(btnClicked.equals(btnSearch)){
        if(txtKey.getText() == null)|| (txtKey.getText().length() == 0))
            return;
        RoomDAO rd = new RoomDAO();
        listRoom = rd.searchRoom(txtKey.getText().trim());

        String[] columnNames = {"Id", "Name", "Type", "Price",
                               "Description"};
        String[][] value = new String[listRoom.size()][5];
        for(int i=0; i<listRoom.size(); i++){
            value[i][0] = listRoom.get(i).getId() + "";
            value[i][1] = listRoom.get(i).getName();
            value[i][2] = listRoom.get(i).getType();
            value[i][3] = listRoom.get(i).getPrice() + "";
            value[i][4] = listRoom.get(i).getDes();
        }
        DefaultTableModel tableModel =
            new DefaultTableModel(value, columnNames) {
                @Override
                public boolean isCellEditable(int row, int column) {
                    //unable to edit cells
                    return false;
                }
            };
        tblResult.setModel(tableModel);
    }
}
}

```

+ Lớp EditRoomFrm.java:

```
package view.room;
```

```

import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.Box;
import javax.swingBoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import dao.RoomDAO;
import model.Room;
import model.User;
import view.user.ManagerHomeFrm;

public class EditRoomFrm extends JFrame implements ActionListener{
    private Room room;
    private JTextField txtId, txtName, txtType, txtPrice, txtDes;
    private JButton btnUpdate, btnReset;
    private User user;

    public EditRoomFrm(User user, Room room){
        super("Edit a room");
        this.user = user;
        this.room = room;

        JPanel pnMain = new JPanel();
        pnMain.setSize(this.getSize().width-5, this.getSize().height-20);
        pnMain.setLayout(new BoxLayout(pnMain,BoxLayout.Y_AXIS));
        pnMain.add(Box.createRigidArea(new Dimension(0,10)));

        JLabel lblHome = new JLabel("Edit a room");
        lblHome.setAlignmentX(Component.CENTER_ALIGNMENT);
        lblHome.setFont (lblHome.getFont ().deriveFont (20.0f));
        pnMain.add(lblHome);
        pnMain.add(Box.createRigidArea(new Dimension(0,20)));

        txtId = new JTextField(15);
        txtId.setEditable(false);
        txtName = new JTextField(15);
        txtType = new JTextField(15);
        txtPrice = new JTextField(15);
        txtDes = new JTextField(15);
        btnUpdate = new JButton("Update");
        btnReset = new JButton("Reset");

        JPanel content = new JPanel();
        content.setLayout(new GridLayout(6,2));
        content.add(new JLabel("Room ID:")); content.add(txtId);
        content.add(new JLabel("Room name:")); content.add(txtName);
        content.add(new JLabel("Type:")); content.add(txtType);
    }
}

```

```

        content.add(new JLabel("Price:")); content.add(txtPrice);
        content.add(new JLabel("Description:")); content.add(txtDes);
        content.add(btnUpdate); content.add(btnReset);
        pnMain.add(content);
        btnUpdate.addActionListener(this);
        btnReset.addActionListener(this);

        initForm();
        this.setContentPane(pnMain);
        this.setSize(600,300);
        this.setLocation(200,10);
        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }

    private void initForm(){
        if(room != null){
            txtId.setText(room.getId()+"");
            txtName.setText(room.getName());
            txtType.setText(room.getType());
            txtPrice.setText(room.getPrice()+"");
            txtDes.setText(room.getDes());
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        JButton btnClicked = (JButton)e.getSource();
        if(btnClicked.equals(btnReset)){
            initForm();
            return;
        }
        if(btnClicked.equals(btnUpdate)){
            btnUpdateClick();
        }
    }
}

private void btnUpdateClick(){
    room.setName(txtName.getText());
    room.setType(txtType.getText());
    room.setPrice(Float.parseFloat(txtPrice.getText()));
    room.setDes(txtDes.getText());

    RoomDAO rd = new RoomDAO();
    if(rd.updateRoom(room)) {
        JOptionPane.showMessageDialog(this,
            "The room is successfully updated!");
        (new ManagerHomeFrm(user)).setVisible(true);
        this.dispose();
    }
}
}

```

### c. Code modul đặt phòng

Chức năng này chỉ trình bày code cho tầng DAO, các lớp tầng giao diện coi như là bài tập cho sinh viên.

+ Lớp RoomDAO.java, bổ sung phương thức:

```
public ArrayList<Room> searchFreeRoom(Date checkin, Date checkout){
    ArrayList<Room> result = new ArrayList<Room>();
    String sql = "SELECT * FROM tblRoom WHERE id NOT IN (SELECT idroom
        FROM tblBookedRoom WHERE checkout > ? AND checkin < ?)";
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    try{
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, sdf.format(checkin));
        ps.setString(2, sdf.format(checkout));
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            Room rm = new Room();
            rm.setId(rs.getInt("id"));
            rm.setName(rs.getString("name"));
            rm.setType(rs.getString("type"));
            rm.setPrice(rs.getFloat("price"));
            rm.setDes(rs.getString("des"));
            result.add(rm);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
```

+ Lớp ClientDAO.java:

```
package dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import model.Client;

public class ClientDAO extends DAO{

    /**
     * search all clients in the tblClient whose name contains the @key
     * @param key
     * @return list of client whose name contains the @key
     */
    public ArrayList<Client> searchClient(String key){
        ArrayList<Client> result = new ArrayList<Client>();
        String sql = "SELECT * FROM tblclient WHERE name LIKE ?";
        try{
            PreparedStatement ps = con.prepareStatement(sql);
```

```

        ps.setString(1, "%" + key + "%");
        ResultSet rs = ps.executeQuery();

        while(rs.next()){
            Client client = new Client();
            client.setId(rs.getInt("id"));
            client.setName(rs.getString("name"));
            client.setIdCard(rs.getString("idcard"));
            client.setAddress(rs.getString("address"));
            client.setTel(rs.getString("tel"));
            client.setEmail(rs.getString("email"));
            client.setNote(rs.getString("note"));
            result.add(client);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}

/**
 * add a new @client into the DB
 * @param client
 */
public void addClient(Client client){
    String sql = "INSERT INTO tblclient(name, idcard, address, tel, email,
        note) VALUES(?, ?, ?, ?, ?, ?)";
    try{
        PreparedStatement ps = con.prepareStatement(sql,
            Statement.RETURN_GENERATED_KEYS);
        ps.setString(1, client.getName());
        ps.setString(2, client.getIdCard());
        ps.setString(3, client.getAddress());
        ps.setString(4, client.getTel());
        ps.setString(5, client.getEmail());
        ps.setString(6, client.getNote());

        ps.executeUpdate();

        //get id of the new inserted client
        ResultSet generatedKeys = ps.getGeneratedKeys();
        if(generatedKeys.next()){
            client.setId(generatedKeys.getInt(1));
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
}

```

+ Lớp BookingDAO.java:

```
package dao;
```

```

import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import model.BookedRoom;
import model.Booking;
import model.Client;

public class BookingDAO extends DAO{

    public BookingDAO() {
        super();
    }

    /**
     * Insert a new booking into the database, including its booked rooms. All are
     * added in a single transaction.
     * @param b
     * @return
     */
    public boolean addBooking(Booking b) {
        String sqlAddBooking = "INSERT INTO tblBooking(idcreator, idclient,
bookingdate, saleoff, note) VALUES(?, ?, ?, ?, ?)";
        String sqlAddBookedRoom = "INSERT INTO tblBookedRoom(idbooking, idroom,
checkin, checkout, price, saleoff, ischeckin) VALUES(?, ?, ?, ?, ?, ?, ?)";
        String sqlCheckbookedRoom = "SELECT * FROM tblBookedRoom WHERE idroom =
? AND checkout > ? AND checkin < ?";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        boolean result = true;
        try {
            con.setAutoCommit(false);
            PreparedStatement ps = con.prepareStatement(sqlAddBooking,
                Statement.RETURN_GENERATED_KEYS);
            ps.setInt(1, b.getCreator().getId());
            ps.setInt(2, b.getClient().getId());
            ps.setString(3, sdf.format(b.getBookedDate()));
            ps.setFloat(4, b.getSaleoff());
            ps.setString(5, b.getNote());

            ps.executeUpdate();
            //get id of the new inserted booking
            ResultSet generatedKeys = ps.getGeneratedKeys();
            if (generatedKeys.next()) {
                b.setId(generatedKeys.getInt(1));

                //insert booked rooms
                for(BookedRoom br: b.getBookedRoom()) {
                    //check if the room is available at the period
                    ps = con.prepareStatement(sqlCheckbookedRoom);
                    ps.setInt(1, br.getRoom().getId());
                    ps.setString(2, sdf.format(br.getCheckin()));
                    ps.setString(3, sdf.format(br.getCheckout()));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
            result = false;
        }
        return result;
    }
}

```

```

        ResultSet rs = ps.executeQuery();
        if(rs.next()) {//unavailable
            result = false;
            try {
                con.rollback();
                con.setAutoCommit(true);
            }catch(Exception ex) {
                result = false;
                ex.printStackTrace();
            }
            return result;
        }

        //insert booked room
        ps = con.prepareStatement(sqlAddBookedRoom,
            Statement.RETURN_GENERATED_KEYS);
        ps.setInt(1, b.getId());
        ps.setInt(2, br.getRoom().getId());
        ps.setString(3, sdf.format(br.getCheckin()));
        ps.setString(4, sdf.format(br.getCheckout()));
        ps.setFloat(5, br.getPrice());
        ps.setFloat(6, br.getSaleoff());
        ps.setBoolean(7, br.isChecked());

        ps.executeUpdate();
        //get id of the new inserted booking
        generatedKeys = ps.getGeneratedKeys();
        if (generatedKeys.next()) {
            br.setId(generatedKeys.getInt(1));
        }
    }

    //con.commit(); //set this line into comment in JUnit test mode
}catch(Exception e) {
    result = false;
    try {
        con.rollback();
    }catch(Exception ex) {
        result = false;
        ex.printStackTrace();
    }
    e.printStackTrace();
}finally {
    try {
        //con.setAutoCommit(true); //set this line into comment in
        //JUnit test mode
    }catch(Exception ex) {
        result = false;
        ex.printStackTrace();
    }
}
return result;
}
}

```

#### d. Code modul xem thống kê phòng theo doanh thu

Chức năng này chỉ trình bày code cho tầng DAO, các lớp tầng giao diện coi như là bài tập cho sinh viên.

+ Lớp RoomStatDAO.java:

```

package dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import model.RoomStat;

public class RoomStatDAO extends DAO{

    public RoomStatDAO() {
        super();
    }

    public ArrayList<RoomStat> getRoomStat(Date startDate, Date endDate){
        ArrayList<RoomStat> result = new ArrayList<RoomStat>();
        String sql = "SELECT a.id, a.name, a.type, a.des, (SELECT
SUM(DATEDIFF(LEAST(b.checkout, ?), GREATEST(b.checkin, ?))) FROM tblBookedRoom b
WHERE b.idroom = a.id AND b.checkout > ? AND b.checkin < ? AND b.ischeckin = 1
GROUP BY b.idroom) as days, (SELECT SUM(DATEDIFF(LEAST(b.checkout, ?),
GREATEST(b.checkin, ?))*b.price) FROM tblBookedRoom b WHERE b.idroom = a.id AND
b.checkout > ? AND b.checkin < ? AND b.ischeckin = 1 GROUP BY b.idroom) as income
FROM tblRoom a ORDER BY income DESC, days DESC";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

        try {
            PreparedStatement ps = con.prepareStatement(sql);
            ps.setString(1, sdf.format(endDate));
            ps.setString(2, sdf.format(startDate));
            ps.setString(3, sdf.format(startDate));
            ps.setString(4, sdf.format(endDate));
            ps.setString(5, sdf.format(endDate));
            ps.setString(6, sdf.format(startDate));
            ps.setString(7, sdf.format(startDate));
            ps.setString(8, sdf.format(endDate));
            ResultSet rs = ps.executeQuery();

            while(rs.next()) {
                RoomStat r = new RoomStat();
                r.setId(rs.getInt("id"));
                r.setName(rs.getString("name"));
                r.setType(rs.getString("type"));
                r.setDes(rs.getString("des"));
                r.setTotalDay(rs.getInt("days"));
                r.setTotalIncome(rs.getFloat("income"));
            }
        }
    }
}

```

```

                result.add(r);
            }
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
    return result;
}
}

```

+ Lớp BookingDAO.java, bổ sung phương thức:

```

/**
 * get list of booking involved the room whose @idroom is given between
startDate and @endDate
 * @param idroom
 * @param startDate
 * @param endDate
 * @return
 */
public ArrayList<Booking> getBookingOfRoom(int idroom, Date startDate, Date
endDate){
    ArrayList<Booking> result = new ArrayList<Booking>();
    String sql = "SELECT a.id as idbookedroom, GREATEST(a.checkin,?) as
checkin, LEAST(a.checkout,?) as checkout, a.price, a.saleoff as roomsaleoff, b.id as
idbooking, b.saleoff as bookingsaleoff, c.id as idclient, c.name, c.address,
c.idcard, c.tel FROM tblBookedRoom a, tblBooking b, tblClient c WHERE a.idroom = ?
AND a.ischeckin = 1 AND a.checkout > ? AND a.checkin < ? AND b.id = a.idbooking AND
c.id = b.idclient";
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    try{
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, sdf.format(startDate));
        ps.setString(2, sdf.format(endDate));
        ps.setInt(3, idroom);
        ps.setString(4, sdf.format(startDate));
        ps.setString(5, sdf.format(endDate));
        ResultSet rs = ps.executeQuery();

        //a == null ? b : (b == null ? a : (a.before(b) ? a : b));
        while(rs.next()){
            Booking b = new Booking();
            b.setId(rs.getInt("idbooking"));
            b.setSaleoff(rs.getFloat("bookingsaleoff"));
            //client
            Client c = new Client();
            c.setId(rs.getInt("idclient"));
            c.setName(rs.getString("name"));
            c.setAddress(rs.getString("address"));
            c.setIdCard(rs.getString("idcard"));
            b.setClient(c);
            //booked room
            BookedRoom br = new BookedRoom();

```

```

        br.setId(rs.getInt("idbookedroom"));
        br.setSaleoff(rs.getFloat("roomsaleoff"));
        br.setPrice(rs.getFloat("price"));
        br.setCheckin(rs.getDate("checkin"));
        br.setCheckout(rs.getDate("checkout"));
        b.getBookedRoom().add(br);
        result.add(b);
    }
} catch(Exception e){
    e.printStackTrace();
}
return result;
}

```

### 9.5.2 Kiểm thử đơn vị - JUnit test

#### a. Lập kế hoạch test

Các trường hợp phải test cho 3 modul này (bỏ quan phần kiểm thử đơn vị cho hàm kiểm tra đăng nhập):

TT	Chức năng	Lớp DAO	Hàm/phương thức	Các trường hợp cần kiểm thử
1	Sửa thông tin phòng	RoomDAO	searchRoom()	Tìm phòng tồn tại
2				Tìm phòng không tồn tại
3			updateRoom()	Phòng đã có trong CSDL
4				Phòng chưa có trong CSDL
5	Đặt phòng	RoomDAO	searchFreeRoom()	Khoảng thời gian ở còn phòng trống
6				Khoảng thời gian ở hết phòng trống
7		ClientDAO	searchClient()	Tìm khách tồn tại
8				Tìm khách không tồn tại
9		BookingDAO	addBooking()	Tồn tại khách hàng, tồn tại phòng, và phòng chưa bị đặt vào thời gian đó
10				Không tồn tại khách hàng
11				Không tồn tại phòng
12				Tồn tại khách hàng, tồn tại phòng, và phòng bị đặt vào thời gian đó
8	Thống kê phòng theo doanh thu	RoomStatDAO	getRoomStat()	Có ít nhất 2 phòng có dữ liệu đặt, có ít nhất 1 phòng

			có 2 lần đặt trong khoảng thống kê
9			Có đúng 1 phòng có dữ liệu đặt, có ít nhất 2 lần đặt trong khoảng thống kê
10			Có đúng 1 phòng có dữ liệu đặt, có đúng 1 lần đặt trong khoảng thống kê
11			Không có phòng nào có dữ liệu đặt
12	BookingDAO	getBookingOfRoom()	Có ít nhất 2 lần đặt nằm trong khoảng thời gian thống kê
13			Có đúng 1 lần đặt nằm trong khoảng thời gian thống kê: start date < checkin < checkout < end date
14			Có đúng 1 lần đặt nằm trong khoảng thời gian thống kê: checkin < start date < checkout < end date
15			Có đúng 1 lần đặt nằm trong khoảng thời gian thống kê: start date < checkin < end date < checkout
16			Không có lần đặt nào trong khoảng thời gian thống kê

### b. Kiểm thử các phương thức của chức năng sửa phòng

+ Lớp RoomDaoTest.java:

```
package test.unit;
import java.sql.Connection;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import org.junit.Assert;
import org.junit.Test;
import dao.DAO;
import dao.RoomDAO;
```

```

import model.Room;

public class RoomDaoTest {
    RoomDAO rd = new RoomDAO();

    @Test
    public void testSearchRoomException1(){
        String key = "xxxxxxxxxx";
        ArrayList<Room> listRoom = rd.searchRoom(key);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(0, listRoom.size());
        return;
    }

    @Test
    public void testSearchRoomException2(){
        String key = "rn";
        ArrayList<Room> listRoom = rd.searchRoom(key);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(0, listRoom.size());
        return;
    }

    @Test
    public void testSearchRoomStandard1(){
        String key = "1";
        ArrayList<Room> listRoom = rd.searchRoom(key);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(6, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertTrue(listRoom.get(i).getName().toLowerCase().
                contains(key.toLowerCase())));
        }
        return;
    }

    @Test
    public void testSearchRoomStandard2(){
        String key = "3";
        ArrayList<Room> listRoom = rd.searchRoom(key);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(2, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertTrue(listRoom.get(i).getName().toLowerCase().
                contains(key.toLowerCase())));
        }
        return;
    }

    @Test
    public void testUpdateRoom(){
        Connection con = DAO.con;
        String newType = "test type";
    }
}

```

```

String newDes = "test des";
float newPrice = 5f;
String key = "102";

try{
    con.setAutoCommit(false);
    ArrayList<Room> lr = rd.searchRoom(key);

    lr.get(0).setType(newType);
    lr.get(0).setDes(newDes);
    lr.get(0).setPrice(newPrice);
    rd.updateRoom(lr.get(0));

    //test the new updated row
    lr.clear();
    lr = rd.searchRoom(key);
    Assert.assertTrue(lr.get(0).getName().contains(key));
    Assert.assertEquals(newType, lr.get(0).getType());
    Assert.assertEquals(newPrice, lr.get(0).getPrice(), 0.000001f);
    Assert.assertEquals(newDes, lr.get(0).getDes());
} catch(Exception e){
    e.printStackTrace();
} finally{
    try{
        con.rollback();
        con.setAutoCommit(true);
    } catch(Exception ex){
        ex.printStackTrace();
    }
}
return;
}
}

```

### c. Kiểm thử các phương thức của chức năng đặt phòng

+ Lớp RoomDaoTest.java, bổ sung các test:

```

@Test
public void testSearchFreeRoom1(){
    Date checkin = new GregorianCalendar(2020,
        Calendar.APRIL, 28).getTime();
    Date checkout = new GregorianCalendar(2020,
        Calendar.APRIL, 30).getTime();
    ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
    Assert.assertNotNull(listRoom);
    Assert.assertEquals(10, listRoom.size());
}

@Test
public void testSearchFreeRoom2(){
    Date checkin = new GregorianCalendar(2020,

```

```

        Calendar.APRIL, 28).getTime();
        Date checkout = new GregorianCalendar(2020, Calendar.MAY, 1).getTime();
        ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(7, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertFalse(listRoom.get(i).getId() == 1);
            Assert.assertFalse(listRoom.get(i).getId() == 3);
            Assert.assertFalse(listRoom.get(i).getId() == 5);
        }
    }

    @Test
    public void testSearchFreeRoom3(){
        Date checkin = new GregorianCalendar(2020, Calendar.MAY, 1).getTime();
        Date checkout = new GregorianCalendar(2020, Calendar.MAY, 6).getTime();
        ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(7, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertFalse(listRoom.get(i).getId() == 1);
            Assert.assertFalse(listRoom.get(i).getId() == 3);
            Assert.assertFalse(listRoom.get(i).getId() == 5);
        }
    }

    @Test
    public void testSearchFreeRoom4(){
        Date checkin = new GregorianCalendar(2020, Calendar.MAY, 2).getTime();
        Date checkout = new GregorianCalendar(2020, Calendar.MAY, 5).getTime();
        ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(8, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertFalse(listRoom.get(i).getId() == 3);
            Assert.assertFalse(listRoom.get(i).getId() == 5);
        }
    }

    @Test
    public void testSearchFreeRoom5(){
        Date checkin = new GregorianCalendar(2020, Calendar.MAY, 9).getTime();
        Date checkout = new GregorianCalendar(2020,
            Calendar.MAY, 11).getTime();
        ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(9, listRoom.size());
        for(int i=0; i<listRoom.size(); i++){
            Assert.assertFalse(listRoom.get(i).getId() == 1);
        }
    }

    @Test
    public void testSearchFreeRoom6(){
        Date checkin = new GregorianCalendar(2020, Calendar.MAY, 11).getTime();

```

```

        Date checkout = new GregorianCalendar(2020,
            Calendar.MAY, 13).getTime();
        ArrayList<Room> listRoom = rd.searchFreeRoom(checkin, checkout);
        Assert.assertNotNull(listRoom);
        Assert.assertEquals(10, listRoom.size());
    }
}

```

+ Lớp BookingDaoTest.java:

```

package test.unit;

import java.sql.Connection;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import org.junit.Assert;
import org.junit.Test;
import dao.BookingDAO;
import dao.DAO;
import dao.RoomDAO;
import model.BookedRoom;
import model.Booking;
import model.Client;
import model.Room;
import model.User;

public class BookingDaoTest {

    @Test
    public void testAddBooking1() {//exception
        //data preparation
        Booking booking = new Booking();
        booking.setSaleoff(0);
        //booked date
        Date bookedDate = new GregorianCalendar(2020,
            Calendar.APRIL, 20).getTime();
        booking.setBookedDate(bookedDate);
        //creator
        User creator = new User();
        creator.setId(2);
        booking.setCreator(creator);
        //client
        Client client = new Client();
        client.setId(6);
        booking.setClient(client);
    }
}

```

```

//room1
Room room = new Room();
room.setId(1);
//booked room1
BookedRoom br = new BookedRoom();
br.setRoom(room);
br.setChecked(false);
br.setPrice(200);
br.setSaleoff(0);
br.setCheckin(new GregorianCalendar(2020,
    Calendar.APRIL, 28).getTime());
br.setCheckout(new GregorianCalendar(2020,
    Calendar.MAY, 1).getTime());
booking.getBookedRoom().add(br);

// test
BookingDAO bd = new BookingDAO();
Connection con = DAO.con;
try{
    con.setAutoCommit(false);
    Assert.assertFalse(bd.addBooking(booking));
}catch(Exception e){
    e.printStackTrace();
}finally{
    try{
        if(!con.getAutoCommit()) {
            con.rollback();
            con.setAutoCommit(true);
        }
    }catch(Exception ex){
        ex.printStackTrace();
    }
}
return;
}

@Test
public void testAddBooking2() {//exception
//data preparation
Booking booking = new Booking();
booking.setSaleoff(0);
//booked date
Date bookedDate = new GregorianCalendar(2020,
    Calendar.APRIL, 20).getTime();
booking.setBookedDate(bookedDate);
}

```

```

//creator
User creator = new User();
creator.setId(2);
booking.setCreator(creator);
//client
Client client = new Client();
client.setId(6);
booking.setClient(client);
//room1
Room room = new Room();
room.setId(1);
//booked room1
BookedRoom br = new BookedRoom();
br.setRoom(room);
br.setChecked(false);
br.setPrice(200);
br.setSaleoff(0);
br.setCheckin(new GregorianCalendar(2020,
    Calendar.APRIL, 25).getTime());
br.setCheckout(new GregorianCalendar(2020,
    Calendar.APRIL, 28).getTime());
booking.getBookedRoom().add(br);
//booked room2
br = new BookedRoom();
br.setRoom(room);
br.setChecked(false);
br.setPrice(200);
br.setSaleoff(0);
br.setCheckin(new GregorianCalendar(2020,
    Calendar.APRIL, 26).getTime());
br.setCheckout(new GregorianCalendar(2020,
    Calendar.APRIL, 28).getTime());
booking.getBookedRoom().add(br);

// test
BookingDAO bd = new BookingDAO();
Connection con = DAO.con;
try{
    con.setAutoCommit(false);
    Assert.assertFalse(bd.addBooking(booking));
}catch(Exception e){
    e.printStackTrace();
}finally{
    try{
        if(!con.getAutoCommit()) {
            con.rollback();
    }
}

```

```

        con.setAutoCommit(true);
    }
}catch(Exception ex){
    ex.printStackTrace();
}
}
return;
}

@Test
public void testAddBooking3() {//standard
    //data preparation
    Booking booking = new Booking();
    booking.setSaleoff(0);
    //booked date
    Date bookedDate = new GregorianCalendar(2020,
        Calendar.APRIL, 20).getTime();
    booking.setBookedDate(bookedDate);
    //creator
    User creator = new User();
    creator.setId(2);
    booking.setCreator(creator);
    //client
    Client client = new Client();
    client.setId(6);
    booking.setClient(client);
    //room1
    Room room = new Room();
    room.setId(1);
    //booked room1
    BookedRoom br = new BookedRoom();
    br.setRoom(room);
    br.setChecked(false);
    br.setPrice(200);
    br.setSaleoff(0);
    br.setCheckin(new GregorianCalendar(2020,
        Calendar.APRIL, 25).getTime());
    br.setCheckout(new GregorianCalendar(2020,
        Calendar.APRIL, 28).getTime());
    booking.getBookedRoom().add(br);

    // test
    BookingDAO bd = new BookingDAO();
    Connection con = DAO.con;
    try{
        con.setAutoCommit(false);

```

```

        Assert.assertTrue(bd.addBooking(booking));

        RoomDAO rd = new RoomDAO();
        ArrayList<Room> fr = rd.searchFreeRoom(new
        GregorianCalendar(2020, Calendar.APRIL, 24).getTime(),
        new GregorianCalendar(2020, Calendar.APRIL,
        27).getTime());
        Assert.assertNotNull(fr);
        for(Room r:fr) {
            Assert.assertFalse(r.getId() == 1);
        }
    }catch(Exception e){
        e.printStackTrace();
    }finally{
        try{
            if(!con.getAutoCommit()) {
                con.rollback();
                con.setAutoCommit(true);
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
    return;
}

@Test
public void testAddBooking4() {//standard
    //data preparation
    Booking booking = new Booking();
    booking.setSaleoff(0);
    //booked date
    Date bookedDate = new GregorianCalendar(2020,
        Calendar.APRIL, 20).getTime();
    booking.setBookedDate(bookedDate);
    //creator
    User creator = new User();
    creator.setId(2);
    booking.setCreator(creator);
    //client
    Client client = new Client();
    client.setId(6);
    booking.setClient(client);
    //room1
    Room room = new Room();
}

```

```

room.setId(1);
//booked room1
BookedRoom br = new BookedRoom();
br.setRoom(room);
br.setChecked(false);
br.setPrice(200);
br.setSaleoff(0);
br.setCheckin(new GregorianCalendar(2020,
    Calendar.APRIL, 25).getTime());
br.setCheckout(new GregorianCalendar(2020,
    Calendar.APRIL, 28).getTime());
booking.getBookedRoom().add(br);
//room2
room = new Room();
room.setId(5);
//booked room2
br = new BookedRoom();
br.setRoom(room);
br.setChecked(false);
br.setPrice(200);
br.setSaleoff(0);
br.setCheckin(new GregorianCalendar(2020,
    Calendar.APRIL, 25).getTime());
br.setCheckout(new GregorianCalendar(2020,
    Calendar.APRIL, 28).getTime());
booking.getBookedRoom().add(br);

// test
BookingDAO bd = new BookingDAO();
Connection con = DAO.con;
try{
    con.setAutoCommit(false);
    Assert.assertTrue(bd.addBooking(booking));

    RoomDAO rd = new RoomDAO();
    ArrayList<Room> fr = rd.searchFreeRoom(new
        GregorianCalendar(2020, Calendar.APRIL, 24).getTime(),
    new GregorianCalendar(2020, Calendar.APRIL, 27).getTime());
    Assert.assertNotNull(fr);
    for(Room r:fr) {
        Assert.assertFalse(r.getId() == 1);
        Assert.assertFalse(r.getId() == 5);
    }
} catch(Exception e){
    e.printStackTrace();
} finally{
}

```

```

        try{
            if(!con.getAutoCommit()) {
                con.rollback();
                con.setAutoCommit(true);
            }
        }catch(Exception ex){
            ex.printStackTrace();
        }
    }
    return;
}
}

```

*d. Kiểm thử các phương thức của chức năng thống kê phòng theo doanh thu*

+ Lớp RoomStatDaoTest.java:

```

package test.unit;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import org.junit.Assert;
import org.junit.Test;
import dao.RoomStatDAO;
import model.RoomStat;

public class RoomStatDaoTest {

    @Test
    public void testGetRoomStat1() {//standard
        RoomStatDAO rsd = new RoomStatDAO();
        Date startDate = new GregorianCalendar(2020,
            Calendar.MAY, 05).getTime();
        Date endDate = new GregorianCalendar(2020,
            Calendar.MAY, 11).getTime();

        ArrayList<RoomStat> lrs =
            rsd.getRoomStat(startDate, endDate);
        Assert.assertNotNull(lrs);
        Assert.assertEquals(10, lrs.size());

        Assert.assertTrue(lrs.get(0).getName().equalsIgnoreCase("101"));
    }
}

```

```

        Assert.assertEquals(5, lrs.get(0).getTotalDay(), 0.00001f);
        Assert.assertEquals(950, lrs.get(0).getTotalIncome(),
                           0.00001f);

        Assert.assertTrue(lrs.get(1).getName().equalsIgnoreCase("105"));
        Assert.assertEquals(2, lrs.get(1).getTotalDay(), 0.00001f);
        Assert.assertEquals(300, lrs.get(1).getTotalIncome(),
                           0.00001f);

        Assert.assertTrue(lrs.get(2).getName().equalsIgnoreCase("102"));
        Assert.assertEquals(0, lrs.get(2).getTotalDay(), 0.00001f);
        Assert.assertEquals(0, lrs.get(2).getTotalIncome(),
                           0.00001f);
    }

    @Test
    public void testGetRoomStat2() { //standard
        RoomStatDAO rsd = new RoomStatDAO();
        Date startDate = new GregorianCalendar(2020,
                                                Calendar.APRIL, 01).getTime();
        Date endDate = new GregorianCalendar(2020,
                                              Calendar.DECEMBER, 31).getTime();

        ArrayList<RoomStat> lrs =
            rsd.getRoomStat(startDate, endDate);
        Assert.assertNotNull(lrs);
        Assert.assertEquals(10, lrs.size());

        Assert.assertTrue(lrs.get(0).getName().equalsIgnoreCase("101"));
        Assert.assertEquals(7, lrs.get(0).getTotalDay(), 0.00001f);
        Assert.assertEquals(1350, lrs.get(0).getTotalIncome(),
                           0.00001f);

        Assert.assertTrue(lrs.get(1).getName().equalsIgnoreCase("103"));
        Assert.assertEquals(6, lrs.get(1).getTotalDay(), 0.00001f);
        Assert.assertEquals(1200, lrs.get(1).getTotalIncome(),
                           0.00001f);

        Assert.assertTrue(lrs.get(2).getName().equalsIgnoreCase("105"));
        Assert.assertEquals(7, lrs.get(2).getTotalDay(), 0.00001f);
        Assert.assertEquals(1050, lrs.get(2).getTotalIncome(),
                           0.00001f);
    }
}

```

```

        0.00001f);

    Assert.assertTrue(lrs.get(3).getName().equalsIgnoreCase("202"));
    Assert.assertEquals(4, lrs.get(3).getTotalDay(), 0.00001f);
    Assert.assertEquals(600, lrs.get(3).getTotalIncome(),
        0.00001f);

    Assert.assertTrue(lrs.get(4).getName().equalsIgnoreCase("102"));
    Assert.assertEquals(0, lrs.get(4).getTotalDay(), 0.00001f);
    Assert.assertEquals(0, lrs.get(4).getTotalIncome(),
        0.00001f);
}

@Test
public void testGetRoomStat3() {//exception
    RoomStatDAO rsd = new RoomStatDAO();
    Date startDate = new GregorianCalendar(2020,
        Calendar.JULY, 01).getTime();
    Date endDate = new GregorianCalendar(2020,
        Calendar.DECEMBER, 31).getTime();

    ArrayList<RoomStat> lrs =
        rsd.getRoomStat(startDate, endDate);
    Assert.assertNotNull(lrs);
    Assert.assertEquals(10, lrs.size());

    Assert.assertTrue(lrs.get(0).getName().equalsIgnoreCase("101"));
    Assert.assertEquals(0, lrs.get(0).getTotalDay(), 0.00001f);
    Assert.assertEquals(0, lrs.get(0).getTotalIncome(),
        0.00001f);
}
}

```

+ Lớp BookingDaoTest.java, bổ sung các test:

```

@Test
public void testGetBillofRoom1() {
    //standard - 1 booked sd<ci<co<ed
    BookingDAO bd = new BookingDAO();
}

```

```

Date startDate = new GregorianCalendar(2020,
    Calendar.MAY, 04).getTime();
Date endDate = new GregorianCalendar(2020,
    Calendar.MAY, 11).getTime();

ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
    startDate, endDate);
Assert.assertNotNull(lbk);
Assert.assertEquals(1, lbk.size());
Assert.assertEquals(950, lbk.get(0).getTotal(), 0.00001f);

Assert.assertEquals("Tom", lbk.get(0).getClient().getName());
}

@Test
public void testGetBillofRoom2() {
//standard - 1 booked cisdcoed
    BookingDAO bd = new BookingDAO();
    Date startDate = new GregorianCalendar(2020,
        Calendar.MAY, 07).getTime();
    Date endDate = new GregorianCalendar(2020,
        Calendar.MAY, 11).getTime();

    ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
        startDate, endDate);
    Assert.assertNotNull(lbk);
    Assert.assertEquals(1, lbk.size());
    Assert.assertEquals(570, lbk.get(0).getTotal(), 0.00001f);

    Assert.assertEquals("Tom", lbk.get(0).getClient().getName());
}

@Test
public void testGetBillofRoom3() {
//standard - 1 booked sdcikedco
    BookingDAO bd = new BookingDAO();
    Date startDate = new GregorianCalendar(2020,
        Calendar.MAY, 04).getTime();
    Date endDate = new GregorianCalendar(2020,
        Calendar.MAY, 07).getTime();

    ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
        startDate, endDate);
    Assert.assertNotNull(lbk);
    Assert.assertEquals(1, lbk.size());
}

```

```

        Assert.assertEquals(380, lbk.get(0).getTotal(), 0.00001f);

    Assert.assertEquals("Tom",lbk.get(0).getClient().getName());
}

@Test
public void testGetBillofRoom4() {
    //standard - 2 booked sd<ci<co<ed
    BookingDAO bd = new BookingDAO();
    Date startDate = new GregorianCalendar(2020,
        Calendar.APRIL, 28).getTime();
    Date endDate = new GregorianCalendar(2020,
        Calendar.MAY, 11).getTime();

    ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
        startDate, endDate);
    Assert.assertNotNull(lbk);
    Assert.assertEquals(2, lbk.size());
    Assert.assertEquals(400, lbk.get(0).getTotal(), 0.00001f);

    Assert.assertEquals("Pierre",lbk.get(0).getClient().getName());

    Assert.assertEquals(950, lbk.get(1).getTotal(), 0.00001f);

    Assert.assertEquals("Tom",lbk.get(1).getClient().getName());
}

@Test
public void testGetBillofRoom5() {
    //exception - 0 booked ci<co<=sd<ed
    BookingDAO bd = new BookingDAO();
    Date startDate = new GregorianCalendar(2020,
        Calendar.MAY, 10).getTime();
    Date endDate = new GregorianCalendar(2020,
        Calendar.MAY, 11).getTime();

    ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
        startDate, endDate);
    Assert.assertNotNull(lbk);
    Assert.assertEquals(0, lbk.size());
}

@Test
public void testGetBillofRoom6() {
}

```

```
//exception - 0 booked ci<co<=sd<ed
BookingDAO bd = new BookingDAO();
Date startDate = new GregorianCalendar(2020,
    Calendar.APRIL, 01).getTime();
Date endDate = new GregorianCalendar(2020,
    Calendar.APRIL, 29).getTime();

ArrayList<Booking> lbk = bd.getBookingOfRoom(1,
    startDate, endDate);
Assert.assertNotNull(lbk);
Assert.assertEquals(0, lbk.size());
}
```

### 9.5.3 Kiểm thử chức năng - Blackbox test

#### a. Lập kế hoạch test

Các trường hợp phải test cho 3 modul này:

TT	Chức năng	Các trường hợp cần kiểm thử
1	Sửa thông tin phòng	Sửa một phòng đã có trong CSDL
2		Sửa một phòng chưa có trong CSDL
3		Sửa liên tiếp hai lần cùng một phòng đã có trong CSDL
4	Đặt phòng	Có phòng trống và đã có khách hàng
5		Có phòng trống và chưa có khách hàng
6		Không có phòng trống
7		Đặt 2 lần liên tiếp cùng phòng, cùng ngày khi chỉ còn 1 phòng trống duy nhất
8	Thống kê phòng theo doanh thu	Có dữ liệu đặt phòng, thời gian ở nằm gọn trong khoảng thời gian thống kê
9		Có dữ liệu đặt phòng, thời gian ở không nằm trọn trong thời gian thống kê: checkin < start date < checkout
10		Có dữ liệu đặt phòng, thời gian ở không nằm trọn trong thời gian thống kê: checkin < end date < checkout
11		Không có dữ liệu đặt phòng trong thời gian thống kê

#### b. Các test case cho chức năng sửa thông tin phòng

+ Test case 1: sửa một phòng đã có trong CSDL (test case chuẩn)

CSDL trước khi test:

-tblUser:

<b>id</b>	<b>name</b>	<b>username</b>	<b>password</b>	<b>position</b>
1	Manager	manager	manager	manager
2	Administrator	admin	admin	administrator
3	Seller	seller	seller	seller
4	Receptionist	recept	recept	receptionist

- tblHotel:

<b>id</b>	<b>name</b>	<b>address</b>	<b>star</b>	<b>des</b>
1	Metropole	15 Ngõ Quyền, Hà Nội	5	

- tblRoom:

<b>id</b>	<b>idHotel</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>
1	1	101	single	800	garden view
2	1	102	double	1000	garden view
3	1	203	twink	1000	garden view

<b>Các bước thực hiện</b>	<b>Kết quả mong đợi</b>
1. Khởi tạo phần mềm	Giao diện đăng nhập hiện ra, có ô nhập username, password và nút đăng nhập
2. Nhập username = manager password = manager Click đăng nhập	Giao diện trang chủ nhân viên quản lý hiện ra. Có 3 nút chọn: - Quản lý khách sạn - Quản lý phòng - Xem thống kê
3. Click nút quản lý phòng	Giao diện quản lý phòng hiện ra. Có 3 nút chọn: - Thêm phòng - Sửa phòng - Xóa phòng
4. Click nút sửa phòng	Giao diện tìm phòng để sửa hiện lên. Có ô nhập từ khóa, nút tìm kiếm.
5. Nhập từ khóa = 2 và click vào	Kết quả có 2 phòng hiện lên:

nút tìm kiếm	<table border="1"> <thead> <tr> <th><b>id</b></th><th><b>name</b></th><th><b>type</b></th><th><b>price</b></th><th><b>des</b></th></tr> </thead> <tbody> <tr> <td>2</td><td>102</td><td>double</td><td>1000</td><td>garden view</td></tr> <tr> <td>3</td><td>203</td><td>twink</td><td>1000</td><td>garden view</td></tr> </tbody> </table>	<b>id</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>	2	102	double	1000	garden view	3	203	twink	1000	garden view
<b>id</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>												
2	102	double	1000	garden view												
3	203	twink	1000	garden view												
6. Click vào dòng của phòng 102	Giao diện hiện lên thông tin phòng 102: - id = 2 (không sửa được) - name = 102 - type = double - price = 1000 - description = "garden view" và nút Lưu															
7. Sửa thông tin giá phòng: - price = 1200 và click vào nút Lưu	Thông báo hiện lên: sửa phòng thành công!															
8. Click vào nút OK của thông báo	Quay về giao diện chính của quản lý thông tin phòng (bước 2)															

CSDL sau khi test:

- Bảng tblUser và tblHotel giữ nguyên, chỉ có thay đổi ở bảng tblRoom:

<b>id</b>	<b>idHotel</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>
1	1	101	single	800	garden view
2	1	102	double	1200	garden view
3	1	203	twink	1000	garden view

### b. Các test case cho chức năng đặt phòng

+ Test case 1: có phòng trống và có khách hàng trong CSDL (test case chuẩn, bỏ qua phần login)

CSDL trước khi test:

-tblUser:

<b>id</b>	<b>name</b>	<b>username</b>	<b>password</b>	<b>position</b>
1	Manager	manager	manager	manager
2	Administrator	admin	admin	administrator
3	Seller	seller	seller	seller
4	Receptionist	recept	recept	receptionist

- tblHotel:

<b>id</b>	<b>name</b>	<b>address</b>	<b>star</b>	<b>des</b>
1	Metropole	15 Ngõ Quyền, Hà Nội	5	

- tblRoom:

<b>id</b>	<b>idHotel</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>
1	1	101	single	800	garden view
2	1	102	double	1000	garden view
3	1	103	twink	1000	garden view
4	1	201	single	800	garden view
5	1	202	double	1000	garden view
6	1	203	twink	1000	garden view

-tblClient:

<b>id</b>	<b>name</b>	<b>idcard</b>	<b>address</b>	<b>email</b>	<b>tel</b>	<b>des</b>
1	Lê Văn An	1111	Hà Nội	an@gmail.com	123456	
2	Trần Tiến Anh	2222	Đà Nẵng	anh@gmail.com	234567	
3	Cao Thanh Thanh	3333	Sài Gòn	ctt@gmail.com	345678	
4	Hoàng Thị Luyến	4444	Càn Thơ	htl@gmail.com	456789	

- tblBooking:

<b>id</b>	<b>idCreator</b>	<b>idClient</b>	<b>bookingdate</b>	<b>saleoff</b>	<b>note</b>
1	3	1	13/02/2020	0	
2	4	3	13/02/2020	0	
3	3	2	21/04/2020	0	
4	4	4	25/04/2020	0	
5	3	3	26/04/2020	0	

- tblBookedRoom:

<b>id</b>	<b>idBooking</b>	<b>idRoom</b>	<b>checkin</b>	<b>checkout</b>	<b>price</b>	<b>saleoff</b>	<b>isCheckin</b>
1	1	2	14/02/2020	15/02/2020	1200	0	1
2	2	5	14/02/2020	15/02/2020	1200	0	1

3	3	2	29/04/2020	01/05/2020	1000	0	0
4	3	3	29/04/2020	01/05/2020	1000	0	0
5	4	1	28/04/2020	02/05/2020	800	0	0
6	4	6	28/04/2020	02/05/2020	1000	0	0
7	5	1	05/05/2020	10/05/2020	800	800	0

Các bước thực hiện	Kết quả mong đợi																												
1. Nhân viên tên Seller (id=3) đã đăng nhập. Chọn chức năng đặt phòng từ menu chính	Giao diện tìm phòng trống hiện ra. Có ô nhập: - ngày đến: - ngày đi: Nút tìm																												
2. Nhập : - Ngày đến = 30/04/2020 - Ngày đi = 01/05/2020 Và click vào nút Tìm	Kết quả có các phòng trống hiện lên: <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>type</th><th>price</th><th>des</th></tr> </thead> <tbody> <tr> <td>4</td><td>201</td><td>single</td><td>800</td><td>garden view</td></tr> <tr> <td>5</td><td>202</td><td>double</td><td>1000</td><td>garden view</td></tr> </tbody> </table>	id	name	type	price	des	4	201	single	800	garden view	5	202	double	1000	garden view													
id	name	type	price	des																									
4	201	single	800	garden view																									
5	202	double	1000	garden view																									
3. Click chọn phòng 202	Giao diện tìm khách hàng hiện lên. Có ô nhập tên, nút tìm																												
4. Nhập vào: - name = An Và click vào nút Search	Kết quả hiện ra thông tin khách hàng và nút thêm khách hàng mới: <table border="1"> <thead> <tr> <th>id</th><th>name</th><th>idcard</th><th>address</th><th>email</th><th>tel</th><th>des</th></tr> </thead> <tbody> <tr> <td>1</td><td>Lê Văn An</td><td>1111</td><td>Hà Nội</td><td>an@gmail.com</td><td>123456</td><td></td></tr> <tr> <td>2</td><td>Trần Tiến Anh</td><td>2222</td><td>Đà Nẵng</td><td>anh@gmail.com</td><td>234567</td><td></td></tr> <tr> <td>3</td><td>Cao Thanh Thanh</td><td>3333</td><td>Sài Gòn</td><td>ctt@gmail.com</td><td>345678</td><td></td></tr> </tbody> </table>	id	name	idcard	address	email	tel	des	1	Lê Văn An	1111	Hà Nội	an@gmail.com	123456		2	Trần Tiến Anh	2222	Đà Nẵng	anh@gmail.com	234567		3	Cao Thanh Thanh	3333	Sài Gòn	ctt@gmail.com	345678	
id	name	idcard	address	email	tel	des																							
1	Lê Văn An	1111	Hà Nội	an@gmail.com	123456																								
2	Trần Tiến Anh	2222	Đà Nẵng	anh@gmail.com	234567																								
3	Cao Thanh Thanh	3333	Sài Gòn	ctt@gmail.com	345678																								
5. Click vào dòng khách hàng tên An	Giao diện xác nhận hiện ra: - Khách hàng: - tên: Lê Văn An, - idcard: 1111 - địa chỉ: Hà Nội - email: an@gmail.com - tel: 123456 - Thông tin phòng đặt::: <table border="1"> <thead> <tr> <th>TT</th><th>name</th><th>type</th><th>price</th><th>des</th><th>checkin</th><th>checkout</th></tr> </thead> <tbody> <tr> <td>1</td><td>202</td><td>double</td><td>1000</td><td>garden</td><td>30/04/20</td><td>01/05/20</td></tr> </tbody> </table>	TT	name	type	price	des	checkin	checkout	1	202	double	1000	garden	30/04/20	01/05/20														
TT	name	type	price	des	checkin	checkout																							
1	202	double	1000	garden	30/04/20	01/05/20																							

					view	20	20
	<ul style="list-style-type: none"> <li>- Thanh toán khi trả phòng</li> <li>- Nút xác nhận và hủy bỏ</li> </ul>						
6. Click nút xác nhận	Thông báo: đặt phòng thành công!						
7. Click nút OK của thông báo	Quay trở về trang chủ của nhân viên bán hàng						

CSDL sau khi test:

- Các bảng khác không thay đổi, chỉ có bảng tblBooking và tblBookedRoom thay đổi:
- tblBooking:

<b>id</b>	<b>idCreator</b>	<b>idClient</b>	<b>bookingdate</b>	<b>saleoff</b>	<b>note</b>
1	3	1	13/02/2020	0	
2	4	3	13/02/2020	0	
3	3	2	21/04/2020	0	
4	4	4	25/04/2020	0	
5	3	3	26/04/2020	0	
6	3	1	27/04/2020	0	

- tblBookedRoom:

<b>id</b>	<b>idBooking</b>	<b>idRoom</b>	<b>checkin</b>	<b>checkout</b>	<b>price</b>	<b>saleoff</b>	<b>isCheckin</b>
1	1	2	14/02/2020	15/02/2020	1200	0	1
2	2	5	14/02/2020	15/02/2020	1200	0	1
3	3	2	29/04/2020	01/05/2020	1000	0	0
4	3	3	29/04/2020	01/05/2020	1000	0	0
5	4	1	28/04/2020	02/05/2020	800	0	0
6	4	6	28/04/2020	02/05/2020	1000	0	0
7	5	1	05/05/2020	10/05/2020	800	800	0
8	6	5	30/04/2020	01/05/2020	1000	0	0

### c. Các test case cho chức năng thống kê phòng theo doanh thu

- + Test case 1: có phòng trống và có khách hàng trong CSDL (test case chuẩn, bỏ qua phần login)

CSDL trước khi test:

-tblUser:

<b>id</b>	<b>name</b>	<b>username</b>	<b>password</b>	<b>position</b>
1	Manager	manager	manager	manager
2	Administrator	admin	admin	administrator
3	Seller	seller	seller	seller
4	Receptionist	recept	recept	receptionist

- tblHotel:

<b>id</b>	<b>name</b>	<b>address</b>	<b>star</b>	<b>des</b>
1	Metropole	15 Ngõ Quyền, Hà Nội	5	

- tblRoom:

<b>id</b>	<b>idHotel</b>	<b>name</b>	<b>type</b>	<b>price</b>	<b>des</b>
1	1	101	single	800	garden view
2	1	102	double	1000	garden view
3	1	103	twink	1000	garden view
4	1	201	single	800	garden view
5	1	202	double	1000	garden view
6	1	203	twink	1000	garden view

-tblClient:

<b>id</b>	<b>name</b>	<b>idcard</b>	<b>address</b>	<b>email</b>	<b>tel</b>	<b>des</b>
1	Lê Văn An	1111	Hà Nội	an@gmail.com	123456	
2	Trần Tiến Anh	2222	Đà Nẵng	anh@gmail.com	234567	
3	Cao Thanh Thanh	3333	Sài Gòn	ctt@gmail.com	345678	
4	Hoàng Thị Luyễn	4444	Cần Thơ	htl@gmail.com	456789	

- tblBooking:

<b>id</b>	<b>idCreator</b>	<b>idClient</b>	<b>bookingdate</b>	<b>saleoff</b>	<b>note</b>
1	3	1	13/02/2020	0	

2	4	3	13/02/2020	0	
3	3	2	21/04/2020	600	
4	4	4	25/04/2020	0	
5	3	3	26/04/2020	0	
6	3	4	25/05/2020	0	

- tblBookedRoom:

<b>id</b>	<b>idBooking</b>	<b>idRoom</b>	<b>checkin</b>	<b>checkout</b>	<b>price</b>	<b>saleoff</b>	<b>isCheckin</b>
1	1	2	14/02/2020	15/02/2020	1200	0	1
2	2	4	14/02/2020	15/02/2020	1200	0	1
3	3	2	29/04/2020	02/05/2020	1000	0	1
4	3	3	29/04/2020	02/05/2020	1000	0	1
5	4	1	28/04/2020	03/05/2020	800	0	1
6	4	6	28/04/2020	03/05/2020	1000	500	1
7	5	2	05/05/2020	10/05/2020	1000	1000	1
8	6	2	01/06/2020	05/06/2020	1000	0	0

Các bước thực hiện	Kết quả mong đợi
1. Nhân viên quản lý tên Manager (id=1) đã đăng nhập. Chọn chức năng xem thống kê từ menu chính	Giao diện chọn cấu hình thống kê hiện ra. Có các danh sách chọn: <ul style="list-style-type: none"> <li>- đối tượng thống kê: có 5 lựa chọn: khách sạn, phòng, dịch vụ, khách hàng, doanh thu. Mặc định đang chọn khách sạn.</li> <li>- kiểu thống kê: có các lựa chọn cho thống kê khách sạn: theo doanh thu, theo tỉ lệ lấp đầy.</li> </ul> Nút thống kê
2. Click chọn đối tượng thống kê là phòng.	Vẫn giao diện chọn cấu hình thống kê : <ul style="list-style-type: none"> <li>- đối tượng thống kê: có 5 lựa chọn: khách sạn, phòng, dịch vụ, khách hàng, doanh thu. Đang chọn phòng.</li> <li>- kiểu thống kê: có các lựa chọn cho thống kê phòng: theo doanh thu, theo số ngày có khách, theo tỉ lệ lấp đầy.</li> </ul> Nút thống kê
3. Click chọn loại thống kê là theo doanh thu và click xem thống kê	Giao diện thống kê phòng theo doanh thu hiện ra, có các ô nhập: <ul style="list-style-type: none"> <li>- ngày bắt đầu thống kê</li> <li>- ngày kết thúc thống kê</li> </ul> Nút xem
4. Nhập : - Ngày bắt đầu = 01/05/2020	Kết quả thống kê các phòng theo doanh thu hiện lên:

<p>- Ngày kết thúc = 30/05/2020 Và click vào nút Xem</p>	<table border="1"> <thead> <tr> <th>TT</th><th>name</th><th>type</th><th>des</th><th>total day</th><th>income</th></tr> </thead> <tbody> <tr><td>1</td><td>102</td><td>double</td><td>garden view</td><td>6</td><td>4900</td></tr> <tr><td>2</td><td>203</td><td>twink</td><td>garden view</td><td>2</td><td>1800</td></tr> <tr><td>3</td><td>101</td><td>single</td><td>garden view</td><td>2</td><td>1600</td></tr> <tr><td>4</td><td>103</td><td>twink</td><td>garden view</td><td>1</td><td>950</td></tr> <tr><td>5</td><td>201</td><td>single</td><td>garden view</td><td>0</td><td>0</td></tr> <tr><td>6</td><td>202</td><td>double</td><td>garden view</td><td>0</td><td>0</td></tr> </tbody> </table>	TT	name	type	des	total day	income	1	102	double	garden view	6	4900	2	203	twink	garden view	2	1800	3	101	single	garden view	2	1600	4	103	twink	garden view	1	950	5	201	single	garden view	0	0	6	202	double	garden view	0	0
TT	name	type	des	total day	income																																						
1	102	double	garden view	6	4900																																						
2	203	twink	garden view	2	1800																																						
3	101	single	garden view	2	1600																																						
4	103	twink	garden view	1	950																																						
5	201	single	garden view	0	0																																						
6	202	double	garden view	0	0																																						
<p>5. Click chọn xem chi tiết vào dòng của phòng 102</p>	<p>Giao diện thống kê chi tiết của phòng 102 từ 01-30/05/2020:</p> <table border="1"> <thead> <tr> <th>TT</th> <th>client</th> <th>start</th> <th>end</th> <th>price</th> <th>day</th> <th>income</th> </tr> </thead> <tbody> <tr><td>1</td><td>Anh</td><td>01/05/2020</td><td>02/05/2020</td><td>1000</td><td>1</td><td>900</td></tr> <tr><td>2</td><td>Thanh</td><td>05/05/2020</td><td>10/05/2020</td><td>1000</td><td>5</td><td>4000</td></tr> <tr> <td align="right" colspan="6"><b>Tổng</b></td><td><b>6</b></td></tr> <tr> <td align="right" colspan="6"><b>4900</b></td><td></td></tr> </tbody> </table> <p>- Nút OK.</p>	TT	client	start	end	price	day	income	1	Anh	01/05/2020	02/05/2020	1000	1	900	2	Thanh	05/05/2020	10/05/2020	1000	5	4000	<b>Tổng</b>						<b>6</b>	<b>4900</b>													
TT	client	start	end	price	day	income																																					
1	Anh	01/05/2020	02/05/2020	1000	1	900																																					
2	Thanh	05/05/2020	10/05/2020	1000	5	4000																																					
<b>Tổng</b>						<b>6</b>																																					
<b>4900</b>																																											
<p>6. Click nút OK</p>	<p>Quay trở về trang chủ của nhân viên quản lý</p>																																										

+ CSDL sau khi test: Không có thay đổi gì so với trước khi test.

## CHƯƠNG 10: PHA BẢO TRÌ

### 10.1 PHA BẢO TRÌ SAU KHI CHUYỂN GIAO

*Bát cứ thay đổi nào ở bát cứ thành phần nào của phần mềm (bao gồm tài liệu) sau khi phần mềm đó đã trải qua kiểm thử chấp nhận*

#### 10.1.1 Tại sao bảo trì sau khi chuyển giao là cần thiết

- Bảo trì sửa lỗi
  - Để sửa lỗi còn sót lại
    - Phân tích, thiết kế, cài đặt, tài liệu hoặc bát cứ các loại lỗi khác
- Bảo trì hoàn thiện (Perfective maintenance)
  - Các yêu cầu khách hàng thay đổi để cải thiện hiệu năng phần mềm
    - Thêm các chức năng
    - Làm cho phần mềm chạy nhanh hơn
    - Cải thiện việc bảo trì
- Bảo trì thích hợp
  - Đáp ứng những thay đổi với môi trường mà phần mềm hoạt động
    - Phần mềm được chuyển sang trình biên dịch mới, hệ điều hành hoặc/và phần cứng mới
    - Thay đổi mã thuê
    - Mã ZIP 9 số

#### 10.1.2 Người lập trình bảo trì sau khi chuyển giao yêu cầu những gì?

- Ít nhất 67 % tổng số chi phí của phần mềm được dồn lại trong suốt quá trình bảo trì sau khi chuyển giao
- Bảo trì là một nguồn thu nhập chính
- Tuy nhiên, ngày nay nhiều tổ chức chỉ định việc bảo trì cho
  - Những người bắt đầu không bị giám sát và (Unsupervised beginners)
  - Ít người lập trình thành thạo
- Bảo trì sau khi chuyển giao là một trong những khía cạnh khó của sản phẩm phần mềm bởi vì
  - Bảo trì sau khi chuyển giao kết hợp các khía cạnh của các luồng công việc khác
- Cho rằng một bản ghi khuyết điểm được chuyển giao cho người lập trình bảo trì
  - Nhớ là “khuyết điểm” là một thuật ngữ chung của lỗi, thất bại
- Nguyên nhân là gì?
  - Không có cái gì sai
  - Sô tay người dùng có thể bị sai, không phải ở mã lệnh
  - Tuy nhiên, thường có một lỗi nằm trong mã lệnh
- a- Bảo trì sửa lỗi
  - Công cụ nào mà người lập trình bảo trì phải dùng để tìm ra lỗi?
    - Bản ghi khuyết điểm được đưa ra bởi người dùng
    - Mã nguồn
    - Thường không còn gì khác
  - Do đó người lập trình bảo trì phải có kỹ năng gỡ lỗi xuất sắc
    - Lỗi có thể nằm ở bất cứ chỗ nào trong phần mềm

- Nguyên nhân đầu tiên của lỗi có thể nằm ở đặc tả không tồn tại hoặc tài liệu thiết kế
- Giả sử rằng người lập trình bảo trì đã định vị được lỗi
- Vấn đề:
  - Cách cố định lỗi mà không đưa ra lỗi hồi quy
- Cách cực tiểu lỗi hồi quy
  - Tham khảo tài liệu chi tiết của toàn bộ sản phẩm
  - Tham khảo tài liệu chi tiết của mỗi mô đun riêng lẻ
- Cái gì thường xuyên xảy ra
  - Không có tài liệu nào, hoặc
  - Tài liệu không hoàn thiện, hoặc
  - Tài liệu bị lỗi
- Người lập trình phải xem xét lại mã nguồn để tránh đưa ra lỗi hồi quy
- Người lập trình thay đổi mã nguồn
- Kiểm thử để thấy phần chỉnh sửa làm việc một cách chính xác
  - Đặc biệt, việc sử dụng những trường hợp kiểm thử cấu trúc
- *Người lập trình phải*
- Kiểm tra đổi với các lỗi hồi quy
- Sử dụng dữ liệu kiểm thử đã lưu trữ
- Thêm các trường hợp kiểm thử đã xây dựng một cách đặc biệt vào dữ liệu kiểm thử đã lưu trữ để cho việc kiểm thử hồi quy trong tương lai
- Viết tài liệu tất cả các thay đổi
  - Những kỹ năng chính được yêu cầu cho bảo trì sửa lỗi
    - Kỹ năng chuẩn đoán giỏi
    - Kỹ nung kiểm thử giỏi
    - Kỹ năng viết tài liệu giỏi

#### b- Bảo trì hoàn thiện và bảo trì thích ứng

- Người lập trình bảo trì phải đi xuyên suốt các luồng công việc
  - Xác định các yêu cầu
  - Viết các đặc tả
  - Thiết kế
  - Cài đặt và tích hợp
- Việc sử dụng các phần mềm có sẵn từ ban đầu
- Khi người lập trình đã phát triển
  - Các đặc tả được đưa ra bởi những chuyên gia phân tích
  - Thiết kế được đưa ra bởi các chuyên gia thiết kế
  - Mã nguồn được viết bởi các chuyên viên lập trình
  - Nhưng những người lập trình bảo trì phải là chuyên gia ở cả ba lĩnh vực trên và cả lĩnh vực kiểm thử và viết tài liệu

#### Kết luận

- Không có khuôn mẫu cho bảo trì
  - Có phải đó là một công việc cho những người bắt đầu không bị giám sát hoặc
  - Bảo trì nên được thực hiện bởi chuyên gia máy tính không có kỹ năng

c- Phần thưởng của bảo trì

- Bảo trì là một công việc bạc bẽo theo mọi cách
  - Người bảo trì thường lượng với những người dùng không hài lòng về phần mềm
  - Nếu người dùng vui, thì phần mềm sẽ không cần bảo trì
  - Vấn đề của người dùng thường bắt nguồn từ những cá nhân đã phát triển sản phẩm phần mềm, không phải người bảo trì
  - Bản thân mã lệnh có thể được viết rất tồi
  - Bảo trì sau khi chuyển giao bị nhiều người phát triển phần mềm xem thường
  - Trừ khi dịch vụ bảo trì tốt được đưa ra thì khách hàng sẽ thực hiện những giao dịch phát triển trong tương lai ở một nơi khác
  - Bảo trì sau khi chuyển giao là một khía cạnh thử thách nhất của phần mềm và bạc bẽo nhất
- Những người quản lý phải chỉ định công việc bảo trì cho những người lập trình giỏi nhất và
- Trả lương cho họ phù hợp

#### 10.1.3 Quản lý bảo trì sau khi chuyển giao

- Các vấn đề khác nhau về mặt quản lý bảo trì sau khi chuyển giao cần được xem xét
  - Trước tiên, người lập trình bảo trì nên xem xét tệp bản ghi khuyết điểm
  - Nó bao gồm
    - Tất cả các lỗi đã được ghi lại mà chưa sửa và
    - Những đề nghị về các công việc sẽ thực thi về những khuyết điểm đó
  - Trong một thế giới lý tưởng
    - Sửa tất cả mọi lỗi ngay lập tức
    - Sau đó chúng ta công bố phiên bản phần mềm mới ở mọi vị trí
  - Trong thế giới thực
    - Phân bổ các bản ghi lỗi ở tất cả các vị trí
    - Không có nhân viên để bảo trì ngay lập tức
    - Thực hiện nhiều thay đổi ở cùng một lúc sẽ rẻ hơn, đặc biệt nếu có nhiều vị trí cài đặt
- a- Các bản ghi khuyết điểm
- Cần một cơ chế đối với việc thay đổi sản phẩm phần mềm
  - Nếu sản phẩm phần mềm xuất hiện một chức năng không đúng, thì người dùng đưa ra một bản ghi khuyết điểm
    - Bản ghi đó phải đủ thông tin để cho phép người lập trình bảo trì tái tạo lại vấn đề
  - Theo lý tưởng, mỗi khuyết điểm nên được cố định ngay lập tức
    - Trong thực tế, tốt nhất chúng ta có thể làm là nghiên cứu sơ bộ ngay lập tức
  - Nếu khuyết điểm đã được ghi lại trước đó: Đưa thông tin trong tệp bản ghi khuyết điểm tới người dùng
  - Nếu nó là một khuyết điểm mới:
    - Người lập trình bảo trì nên cố gắng tìm
      - Nguyên nhân,

- Cách để sửa khuyết điểm đó, và
    - Cách làm việc xung quanh vấn đề đó
  - Khuyết điểm mới được ghi lại vào tệp tường trình phát hiện lỗi, cùng với tài liệu
    - Dánh sách (Listings)
    - Thiết kế (Designs)
    - Sổ tay (Manuals)
  - Tệp bản ghi khuyết điểm cũng nên chứa những yêu cầu của khách hàng để bảo trì thích hợp và hoàn thiện chức năng
    - Nội dung của tệp phải được định độ ưu tiên bởi khách hàng
    - Những chỉnh sửa tiếp theo là một trong những nội dung có độ ưu tiên cao nhất trong tệp
  - Các bản sao của bản ghi khuyết điểm phải lưu hành tới tất cả
    - Bao gồm: ước lượng khi nào khuyết điểm được sửa
  - Nếu cùng thất bại xảy ra ở một vị trí khác, người dùng có thể xác định
    - Khả năng làm việc xung quanh khuyết điểm
    - Thời gian mà khuyết điểm được sửa
- b- Cho phép thay đổi phần mềm
- Bảo trì sửa lỗi
    - Chỉ định một người lập trình bảo trì xác định lỗi và nguyên nhân của lỗi, sau đó sửa lỗi đó
    - Kiểm thử sửa chữa, kiểm thử toàn bộ phần mềm (kiểm thử hồi quy)
    - Cập nhật tài liệu để phản ánh những thay đổi đã thực hiện
    - Cập nhật những lời giải thích ban đầu để phản ánh
      - Những gì đã thay đổi,
      - Tại sao nó được thay đổi,
      - Ai thực hiện thay đổi, và
      - Khi nào
  - Bảo trì thích hợp và hoàn thiện
    - Giống với bảo trì sửa lỗi, ngoại trừ không có bản ghi khuyết điểm
    - Thay vì có thay đổi trong yêu cầu
  - Điều gì sẽ xảy ra nếu người lập trình không kiểm thử những lỗi đã được sửa một cách thích đáng?
    - Trước khi phần mềm được phân phối, thì phần mềm phải được kiểm thử bởi nhóm SQA
  - Bảo trì sau khi chuyển giao là cực kỳ khó
  - Kiểm thử là khó và tiêu tốn thời gian
    - Được thực hiện bởi nhóm SQA
  - Kỹ thuật phiên bản cơ sở và các bản sao riêng phải được sử dụng
  - Người lập trình thực hiện các thay đổi đối với các bản sao chép riêng của các mô đun mã và kiểm thử chúng
  - Người lập trình đóng băng phiên bản trước đó, và đưa ra phiên bản chính sửa cho nhóm SQA để kiểm thử
  - SQA thực hiện kiểm thử trên phiên bản cơ sở của tất cả các mô đun mã

c- Bảo đảm việc bảo trì

- Bảo trì không là sự cố gắng một lần (Maintenance is not a one-time effort)
- Phải lập kế hoạch cho bảo trì xuyên suốt toàn bộ vòng đời phần mềm
  - Luồng công việc thiết kế - sử dụng kỹ thuật ẩn dấu thông tin
  - Luồng cài đặt – lựa chọn đặt tên có ý nghĩa để thuận tiện cho những người lập trình trong tương lai
  - Tài liệu phải được hoàn thiện và chính xác và phản ánh đúng phiên bản hiện thời của mỗi mô đun mã
- Trong suốt quá trình bảo trì sau khi chuyển giao, bảo trì không phải giàn
  - Luôn luôn biết rõ việc bảo trì trong tương lai là không thể tránh được

d- Vấn đề của bảo trì lặp

- Việc thay đổi những yêu cầu phần mềm gây nhiều khó khăn cho đội phát triển
- Việc thay đổi thường xuyên thường gây bất lợi cho việc bảo trì phần mềm

*Thay đổi bài toán đích*

- The problem is exacerbated during postdelivery maintenance
- Càng nhiều thay đổi
  - Thị sản phẩm phần mềm càng khác xa so với thiết kế ban đầu
  - Việc thay đổi trong tương lai càng trở nên khó hơn
  - Tài liệu trở nên kém tin cậy hơn bình thường
  - Các file kiểm thử hồi quy không được cập nhật
  - Viết lại toàn bộ có thể cần thiết đối với bảo trì trong tương lai
- Giải pháp hiển nhiên
  - Đóng băng các đặc tả khi chúng được ký đến tận khi chuyển giao sản phẩm phần mềm
  - Sau mỗi yêu cầu của bảo trì hoàn thiện, đóng băng các đặc tả trong 3 tháng hoặc 1 năm
- Trong thực tế
  - Khách hàng có thể đưa ra những thay đổi ngay ngày hôm sau
  - Nếu băng lòng với giá cả, thì khách hàng có thể đưa ra những thay đổi hàng ngày
- “Ai trả tiền thì người ấy có tiền” (“He who pays the piper calls the tune”)

#### 10.1.4 Bảo trì sau khi chuyển giao với kỹ năng phát triển

- Những kỹ năng cần thiết cho bảo trì gồm
  - Khả năng xác định nguyên nhân gây ra lỗi của một phần mềm lớn
    - Cũng cần thiết trong suốt quá trình tích hợp và kiểm thử sản phẩm
  - Khả năng thực hiện chức năng có hiệu quả mà không cần có tài liệu chính xác
    - Tài liệu hiếm khi được hoàn thiện đến tận khi chuyển giao
  - Kỹ năng phân tích, thiết kế, cài đặt và kiểm thử
    - Tất cả bốn hoạt động được thực thi trong suốt quá trình phát triển
- Những kỹ năng cần thiết cho bảo trì sau khi chuyển giao giống với những kỹ năng của tất cả các luồng công việc khác
- Điểm chính
  - Người lập trình bảo trì không phải chỉ có kỹ năng rộng ở mọi lĩnh vực mà những kỹ năng đó phải ở trình độ cao

- Sự chuyên môn hóa không thể có ở những người lập trình bảo trì

#### 10.1.5 Kỹ nghệ ngược

- Khi nào tài liệu duy nhất đối với bảo trì sau khi chuyển giao là mã nguồn thì
  - Bắt đầu với mã
  - Tái tạo lại thiết kế
  - Tái tạo lại các đặc tả (cực kỳ khó)
  - Công cụ CASE có thể trợ giúp ((flowcharts, các mục đích trực quan khác))
- Kỹ nghệ lại
  - Kỹ nghệ ngược, được sinh ra bởi kỹ nghệ tiên tiến (Reverse engineering, followed by forward engineering)
  - Mức độ thấp hơn tới cao hơn tới thấp hơn của trừu tượng (Lower to higher to lower levels of abstraction)
- Xây dựng lại
  - Việc cải thiện sản phẩm phần mềm mà không có thay đổi chức năng phần mềm
  - Ví dụ:
    - Cải thiện việc bảo trì
    - Xây dựng lại(XP, agile processes)
- Điều gì sẽ xảy ra nếu chúng ta chỉ có mã thực thi?
  - Xem xét phần mềm như hộp đen
  - Suy luận những đặc tả từ hành vi của phần mềm hiện thời

#### 10.1.6 Công cụ CASE cho bảo trì sau khi chuyển giao

- Công cụ điều khiển cấu hình là cần thiết
  - Công cụ thương mại
    - CCC
  - Công cụ mã nguồn mở
    - CVS
- Các công cụ kỹ nghệ lại
  - Các công cụ thương mại
    - IBM Rational Rose, Together
  - Các công cụ mã nguồn mở
    - Doxygen
- Các công cụ theo dõi – phát hiện
  - Công cụ thương mại
    - IBM Rational ClearQuest
  - Công cụ mã nguồn mở
    - Bugzilla

#### 10.1.7 Thước đo của bảo trì sau khi chuyển giao

- Về bản chất, các hoạt động của bảo trì sau khi chuyển giao là những hoạt động của quá trình phát triển
  - Các thước đo đối với luồng công việc phát triển
- Các thước đo bản ghi khuyết điểm (Defect report metrics)
  - Sự phân loại khuyết điểm

- Trạng thái khuyết điểm (Defect status)

#### 10.1.8 Những thách thức của bảo trì sau khi chuyển giao

- Chương này miêu tả rất nhiều thách thức
- Thách thức khó nhất cần giải quyết
  - Bảo trì khó hơn phát triển, nhưng
  - Những người phát triển có xu hướng nhìn xuống (tend to look down) những người bảo trì và
  - Thường xuyên được trả tiền lương nhiều hơn

### 10.2 BẢO TRÌ HỆ PHẦN MỀM HƯỚNG ĐỐI TƯỢNG

- Bề ngoài, mô hình hướng đối tượng khuyến khích việc bảo trì theo bốn cách
  - Phần mềm bao gồm các đơn vị độc lập
  - Đóng gói (độc lập về mặt khái niệm)
  - Ân giấu thông tin (độc lập về mặt vật lý)
  - Truyền tham số là các giao tiếp duy nhất (Message-passing is the sole communication)
- Thực tế hơi khác (The reality is somewhat different)
- Ba cản trở

*Một là:* Cây phân cấp kế thừa có thể lớn

```

class UndirectedTreeClass
{
  ...
  void displayNode (Node a);
  ...
} // class UndirectedTreeClass

class DirectedTreeClass : public UndirectedTreeClass
{
  ...
} // class DirectedTreeClass

class RootedTreeClass : public DirectedTreeClass
{
  ...
  void displayNode (Node a);
  ...
} // class RootedTreeClass

class BinaryTreeClass : public RootedTreeClass
{
  ...
} // class BinaryTreeClass

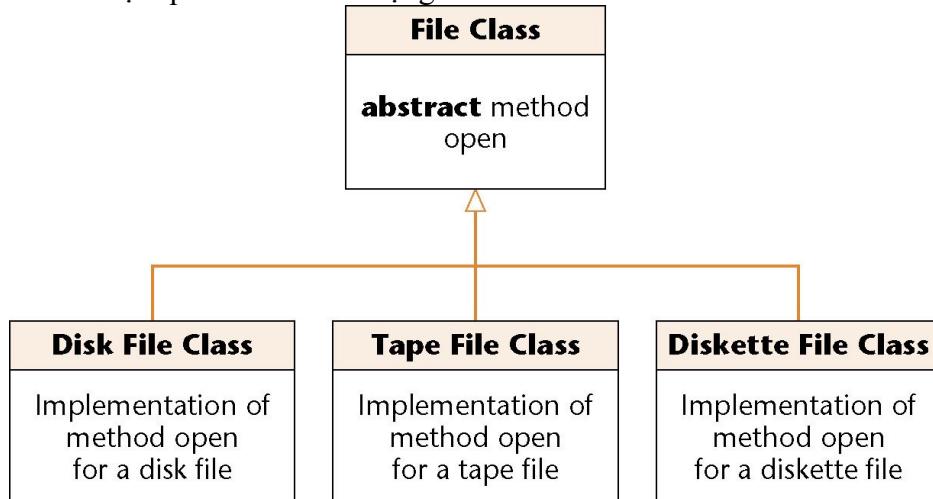
class BalancedBinaryTreeClass : public BinaryTreeClass
{
  Node      hhh;
  displayNode (hhh);
} // class BalancedBinaryTreeClass

```

- Để hình dung ra những gì `displayNode` làm trong `BalancedBinaryTreeClass`, chúng ta phải kiểm tra tất cả cây hoàn thiện
  - Cây hoàn thiện có thể trải rộng toàn bộ phần mềm
  - Khác xa “những đơn vị độc lập” (A far cry from “independent units” )

- Giải pháp
- Công cụ CASE có thể dàn mỏng cây kế thừa (A CASE tool can flatten the inheritance tree)

*Hai là:* Hậu quả của liên kết động và đa hình



- Hệ thống không hoạt động khi gọi myFile.open()
- Phiên bản nào của open có chứa lỗi?
  - Công cụ CASE không thể trợ giúp (công cụ tĩnh)
  - Chúng ta phải theo dõi (kiểm tra)
- Liên kết động và đa hình có thể có
  - Ảnh hưởng tích cực tới đội phát triển nhưng
  - Ảnh hưởng tiêu cực đối với bảo trì

*Ba là:* Hậu quả của kế thừa

- Tạo một lớp mới qua kế thừa
- Lớp con mới
  - Không ảnh hưởng tới lớp cha, và
  - Không ảnh hưởng tới bất cứ lớp con
  - Chỉnh sửa lớp con mới này
  - Một lần nữa, không ảnh hưởng
  - Chỉnh sửa lớp cha
  - Tất cả các lớp con kế thừa đều bị ảnh hưởng
  - “Fragile base class problem”
- Kế thừa có
  - Ảnh hưởng tích đối với người phát triển, nhưng
  - Ảnh hưởng tiêu cực đối với bảo trì

### 10.3 KIỂM THỬ PHA BẢO TRÌ

- Người bảo trì xem xét phần mềm như một tập các thành phần có liên quan lỏng lẻo
  - Chúng không liên quan đến sự phát triển phần mềm
- Kiểm thử hồi quy là cần thiết
  - Lưu giữ các trường hợp kiểm thử và đầu ra của chúng, chỉnh sửa nếu cần thiết

## CHƯƠNG 11: BÀI TẬP DỰ ÁN

### 11.1 HỆ THỐNG QUẢN LÍ BÁN VÉ VÀ LÊN LỊCH CHIẾU CỦA CHUỖI RẠP CHIẾU PHIM

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý bán vé cho một chuỗi rạp chiếu phim với mô tả như sau:

- Hàng có một chuỗi rạp chiếu phim (Mã rạp, tên rạp, địa chỉ, giới thiệu).
- Mỗi rạp chiếu phim có nhiều phòng chiếu khác nhau (Mã phòng chiếu, số lượng ghế, đặc điểm phòng chiếu)
- Mỗi phim (Mã phim, tên phim, loại phim, năm sản xuất, mô tả) có thể được chiếu tại nhiều phòng chiếu khác nhau vào nhiều thời điểm khác nhau
- Mỗi phòng chiếu có thể chiếu nhiều phim khác nhau tại nhiều thời điểm khác nhau
- Mỗi một thời điểm nhất định, trong một phòng chiếu chỉ có duy nhất một phim được chiếu, và bán với một giá vé xác định.
- Cùng một phim, chiếu tại cùng 1 phòng chiếu nhưng nếu ở các khung giờ và ngày khác nhau có thể có giá vé khác nhau.
- Nhân viên chỉ bán vé cho khách hàng khi phòng chiếu tại giờ chiếu mà khách hàng yêu cầu vẫn còn đủ số lượng ghế trống cho khách hàng.
- Khách hàng có thể trả lại vé sau khi đã mua, và có thể phải chịu tiền phạt: trả trước 48h thì miễn phí, trả trước 24h thì mất phí 20%, trả trước 12h thì mất phí 40%, trả trước 6h thì mất phí 60%, trả sau 6h thì mất phí 100%, tính từ giờ khởi chiếu.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý phim**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin phim với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý phim -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin phim -> giao diện tìm phim theo tên hiện ra -> QL nhập tên phim và click tìm kiếm -> danh sách các phim có tên chứa từ khóa hiện ra -> QL chọn sửa một phim -> giao diện sửa phim hiện ra với các thông tin của phim đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các chức năng quản lý nhân viên, quản lý khách hàng, quản lý rạp chiếu, quản lý phòng chiếu được thực hiện tương tự modul quản lý phim.
- Modul "**Lên lịch chiếu**" cho phép quản lý (QL) thực hiện lên lịch chiếu cho phim (phòng chiếu) với mô tả nghiệp vụ: QL chọn menu quản lý lịch chiếu → chọn lên lịch chiếu mới → giao diện lên lịch chiếu hiện ra → QL chọn lên phim từ danh sách sổ xuống + chọn

phòng chiếu từ danh sách sổ xuống + khung giờ chiếu và chọn giá vé từ danh sách sổ xuống + click thêm lịch chiếu → Hệ thống lưu vào CSDL và thông báo thêm thành công.

- Modul “**Sửa lịch chiếu**” cho phép quản lý (QL) sửa lịch chiếu cho phim (phòng chiếu) với mô tả nghiệp vụ: QL chọn menu quản lý lịch chiếu → chọn sửa lịch chiếu → giao diện sửa lịch chiếu hiện ra → QL nhập tên phim (hoặc tên phòng chiếu) + click tìm lịch chiếu theo tên phim (tên phòng chiếu) → Hệ thống hiện lên danh sách các lịch chiếu (thông tin phim + thông tin phòng + thông tin khung giờ + thông tin giá vé), mỗi lịch chiếu trên 1 dòng, cuối dòng có nút chọn sửa → QL click vào nút chọn sửa 1 lịch chiếu → Giao diện chọn sửa hiện ra với các ô chọn đã chọn sẵn như lịch cũ → QL chọn lên phim từ danh sách sổ xuống + chọn phòng chiếu từ danh sách sổ xuống + khung giờ chiếu và chọn giá vé từ danh sách sổ xuống + click thêm lịch chiếu → Hệ thống lưu vào CSDL và thông báo thêm thành công.
- Modul "**Bán vé xem phim**" cho phép nhân viên (NV) rạp thêm thông tin bán vé cho khách hàng với mô tả chi tiết nghiệp vụ: NV chọn menu bán vé -> trang bán vé hiện ra -> NV chọn phòng chiếu hoặc tên phim trong danh sách sổ ra (theo yêu cầu của khách) + chọn khung giờ chiếu -> NV nhập số lượng vé theo yêu cầu khách hàng -> nếu thỏa mãn các ràng buộc nêu trên thì thông báo thành công + in ra vé và hóa đơn cho khách hàng: Tên rạp, số hiệu phòng chiếu, ngày giờ chiếu, tên phim, số lượng vé, giá tiền cho mỗi vé+tổng số tiền của hóa đơn.
- Modul "**Trả vé xem phim**" cho phép nhân viên (NV) rạp cập nhật thông tin khách hàng trả vé với mô tả chi tiết nghiệp vụ: NV chọn menu trả vé -> trang trả vé hiện ra -> NV chọn phòng chiếu hoặc tên phim trong danh sách sổ ra (theo vé của khách) + chọn giờ chiếu -> NV nhập số ghế theo vé trả -> nếu thỏa mãn các ràng buộc nêu trên thì thông báo thành công + in ra hóa đơn phạt cho khách hàng: Tên rạp, số hiệu phòng chiếu, ngày giờ chiếu, tên phim, số lượng vé trả, giá tiền cho mỗi vé+tổng số tiền phạt của hóa đơn.
- Modul "**Thống kê doanh thu**" cho phép nhân viên (NV) rạp thống kê doanh thu bán vé theo phim (hoặc theo rạp) với mô tả chi tiết nghiệp vụ: NV chọn menu thống kê -> chọn thống kê doanh thu theo phim (hoặc theo rạp) -> nhập thời gian bắt đầu và kết thúc thống kê -> danh sách các phim (rạp) có doanh thu lớn nhất được hiện ra, mỗi dòng cho 1 phim: Mã, tên phim, tổng số lượng vé bán ra, tổng doanh thu thu được, sắp xếp theo chiều giảm dần tổng doanh thu.

#### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.

3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phim
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 rạp chiếu phim
  - Thêm, sửa, xóa thông tin 1 phòng chiếu của 1 rạp chiếu phim
  - Lên lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Sửa lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo phim (theo rạp, theo suất chiếu)
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phim
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 rạp chiếu phim
  - Thêm, sửa, xóa thông tin 1 phòng chiếu của 1 rạp chiếu phim
  - Lên lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Sửa lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo phim (theo rạp, theo suất chiếu)
6. Viết tất cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phim
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 rạp chiếu phim
  - Thêm, sửa, xóa thông tin 1 phòng chiếu của 1 rạp chiếu phim
  - Lên lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Sửa lịch chiếu cho 1 phòng chiếu (1 phim, 1 rạp)
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng

- Xem báo cáo doanh thu theo phim (theo rạp, theo suất chiếu)

## 11.2 HỆ THỐNG QUẢN LÝ ĐĂNG KÍ HỌC VÀ ĐIỂM TÍN CHỈ CỦA SINH VIÊN

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý học tín chỉ với mô tả như sau:

- Mỗi sinh viên (Mã SV, mật khẩu, tên, ngày sinh, khóa, quê quán, địa chỉ) được phép đăng kí tối thiểu 10 tín chỉ/học kì và tối đa 15 tín chỉ/học kì
- Mỗi sinh viên được đăng kí nhiều môn học (mã môn, tên môn, số tín chỉ)
- Mỗi môn học có thể có nhiều môn học yêu cầu sinh viên phải hoàn thành trước đó thì mới được đăng kí
- Mỗi môn học có nhiều giáo viên dạy (mã GV, mật khẩu, tên GV).
- Mỗi giáo viên có thể dạy tối đa 2 môn/học kì, trong mỗi môn có thể dạy nhiều lớp, nhưng tổng số tín chỉ mỗi giáo viên được dạy tối thiểu 5 tín chỉ/học kì và tối đa là 10 tín chỉ/học kì
- Mỗi môn học có thể có nhiều lớp học phần (mã lớp, tên lớp, số sv tối đa, phòng học, khung giờ học cố định trong tuần)
- Mỗi lớp học phần sẽ xác định duy nhất một môn học và một giáo viên dạy
- Sinh viên (và giáo viên) không được phép đăng kí học (dạy) hai lớp có trùng buổi học
- Với mỗi môn học, một sinh viên chỉ được đăng kí vào 1 lớp của 1 giáo viên xác định
- Kết quả của sinh viên (điểm thành phần số 1, số 2, số 3, điểm thi, điểm cuối cùng=10% số1+ 20% số2 + 10% số3 + 60% điểm thi) được lưu theo từng môn học
- Điểm trung bình của sinh viên trong học kì được tính bằng trung bình có trọng số là số tín chỉ từng môn học

*Các modul chức năng hoạt động như sau:*

- Modul “*Quản lý sinh viên*” cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin sinh viên với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý sinh viên -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin sinh viên -> giao diện tìm sinh viên theo tên hiện ra -> QL nhập tên sinh viên và click tìm kiếm -> danh sách các sinh viên có tên chứa từ khóa hiện ra -> QL chọn sửa một sinh viên -> giao diện sửa sinh viên hiện ra với các thông tin của sinh viên đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Modul quản lý giảng viên, quản lý môn học, quản lý lớp học phần, quản lý nhân viên được thực hiện theo nghiệp vụ tương tự chức năng quản lý sinh viên.
- Modul "*Lên lịch học cho lớp học phần*" với mô tả chi tiết nghiệp vụ: QL chọn menu lên lịch học cho lớp học phần → giao diện lên lịch học hiện ra với các ô số chọn môn học, lớp

học phần, phòng học, khung giờ, nút xác nhận → QL click chọn môn học từ danh sách sổ xuống → Danh sách lớp học phần của môn học học được cập nhật → QL click chọn lớp học phần của môn học từ danh sách sổ xuống + click chọn phòng học từ danh sách phòng học sổ xuống + click chọn khung giờ trong tuần từ danh sách khung giờ sổ xuống + click xác nhận → Hệ thống lưu lịch học vào CSDL và thông báo thành công.

- Modul "**Đăng kí học online**" với mô tả chi tiết nghiệp vụ: Học sinh đăng nhập -> chọn menu đăng kí tín chỉ cho học kì mới -> trang đăng kí hiện ra -> sinh viên chọn môn học trong danh sách môn học + chọn lớp trong danh sách các lớp (và giảng viên đi kèm) tương ứng với môn học -> nếu thỏa mãn các ràng buộc nêu trên thì thông báo thành công + in ra phiếu đăng kí cho sinh viên: mã SV, tên SV, khóa học, học kì+danh sách các môn học đã đăng kí, mỗi môn có: mã MH, tên MH, số tín chỉ, giờ học, giảng viên
- Modul "**Xem TKB của sinh viên**" với mô tả chi tiết nghiệp vụ: SV chọn menu xem TKB → Giao diện xem TKB hiện lên với phía trên là ô chọn các cách xem TKB theo: tuần, học kỳ → SV chọn xem theo tuần → Phía dưới cập nhật hiển thị thời khóa biểu theo tuần hiện tại của SV: 1 bảng có 7 cột tương ứng 7 ngày, 6 hàng tương ứng 6 kíp học cho mỗi ngày. Trong mỗi ô của bảng hiển thị tên môn học, nhóm môn học, và tên phòng học tương ứng với khung giờ đó.
- Modul giảng viên xem TKB cũng tương tự, chỉ khác là tên các lớp học phần mà giảng viên giảng dạy được hiện lên.
- Modul "**Đăng kí dạy online**" với mô tả chi tiết nghiệp vụ: Giáo viên đăng nhập -> chọn menu đăng kí dạy cho học kì mới -> trang đăng kí hiện ra -> giáo viên chọn môn học trong danh sách môn học + chọn lớp trong danh sách các lớp tương ứng với môn học -> nếu thỏa mãn các ràng buộc nêu trên thì thông báo thành công + in ra phiếu đăng kí cho giáo viên: mã GV, tên GV, học kì+danh sách các lớp học đã đăng kí, mỗi lớp có: tên lớp, tên MH, số tín chỉ, giờ học
- Modul "**Nhập điểm theo lớp học phần**" với mô tả chi tiết nghiệp vụ: giáo viên hoặc giáo vụ (GV) chọn chức năng nhập điểm → giao diện hiện ra danh sách các môn học do GV dạy → GV click chọn 1 môn học → giao diện hiện ra danh sách các lớp học phần của môn học đã chọn do GV dạy → GV click chọn 1 lớp học phần → Giao diện hiện lên danh sách các sinh viên trong lớp học phần, mỗi SV trên 1 dòng với các cột điểm thành phần và cột điểm thi → GV nhập đầy đủ các đầu điểm của các SV + click xác nhận → Hệ thống lưu vào CSDL và thông báo thành công.
- Modul "**Sửa điểm theo sinh viên**" với mô tả chi tiết nghiệp vụ: giáo viên hoặc giáo vụ (GV) chọn chức năng sửa điểm → Giao diện hiện lên ô tìm kiếm → GV nhập tên SV + click tìm → Giao diện hiện lên danh sách các môn học mà SV đã học, mỗi môn với đầy

đủ các cột điểm thành phần và điểm thi → GV click chọn sửa một môn học do mình phụ trách → giao diện hiện lên các ô điểm có sẵn điểm cũ của SV, GV thay đổi các đầu điểm muốn sửa và click lưu lại → Hệ thống cập nhật lại và thông báo thành công.

- Modul "**Thông kê sinh viên khá giỏi**" với mô tả chi tiết nghiệp vụ: quản lý đăng nhập -> chọn menu thống kê -> chọn thống kê sinh viên giỏi -> trang kết quả hiện ra danh sách SV: mã SV, tên SV, khóa học, học kì, tổng số tín chỉ đã học trong học kì, điểm trung bình môn cuối học kì, sắp xếp theo điểm trung bình cả học kì, từ cao đến thấp.
- Modul "**Thông kê môn học theo tỉ lệ sinh viên qua**" với mô tả chi tiết nghiệp vụ: quản lý đăng nhập -> chọn menu thống kê -> chọn thống kê môn học của từng giáo viên dạy theo tỉ lệ SV qua môn -> trang kết quả hiện ra danh sách môn học: mã MH, tên MH, số tín chỉ, giảng viên dạy, điểm trung bình của các SV trong môn học do GV đó dạy, tỉ lệ SV qua môn trong các nhóm do GV đó dạy (tính %). Kết quả được sắp xếp theo tỉ lệ SV qua môn học đó do từng GV dạy, từ cao đến thấp.

#### *Yêu cầu:*

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sinh viên
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 giảng viên
  - Thêm, sửa, xóa thông tin 1 môn học
  - Thêm, sửa, xóa thông tin 1 lớp học phần của 1 môn học
  - Thêm, sửa, xóa thông tin 1 phòng học
  - Lên lịch học cho 1 lớp học phần (ở phòng học nào, vào một khung giờ nào)
  - Sửa lịch học cho 1 lớp học phần
  - Sinh viên đăng ký 1 môn học
  - Sinh viên sửa đăng ký 1 môn học
  - Sinh viên xem TKB
  - Giảng viên đăng ký dạy 1 lớp học phần
  - Giảng viên sửa đăng ký dạy 1 lớp học phần
  - Giảng viên xem TKB
  - Giảng viên nhập điểm của 1 lớp học phần
  - Giảng viên sửa điểm của 1 sinh viên ở 1 môn học
  - Xem báo cáo môn học có tỉ lệ điểm cao

- Xem báo cáo sinh viên có điểm tổng kết cao
- Xem báo cáo sinh viên nợ tín chỉ nhiều
- Xem báo cáo tỉ lệ sinh viên qua theo môn học của giảng viên

4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:

- Thêm, sửa, xóa thông tin 1 sinh viên
- Thêm, sửa, xóa thông tin 1 nhân viên
- Thêm, sửa, xóa thông tin 1 giảng viên
- Thêm, sửa, xóa thông tin 1 môn học
- Thêm, sửa, xóa thông tin 1 lớp học phần của 1 môn học
- Thêm, sửa, xóa thông tin 1 phòng học
- Lên lịch học cho 1 lớp học phần (ở phòng học nào, vào một khung giờ nào)
- Sửa lịch học cho 1 lớp học phần
- Sinh viên đăng ký 1 môn học
- Sinh viên sửa đăng ký 1 môn học
- Sinh viên xem TKB
- Giảng viên đăng ký dạy 1 lớp học phần
- Giảng viên sửa đăng ký dạy 1 lớp học phần
- Giảng viên xem TKB
- Giảng viên nhập điểm của 1 lớp học phần
- Giảng viên sửa điểm của 1 sinh viên ở 1 môn học
- Xem báo cáo môn học có tỉ lệ điểm cao
- Xem báo cáo sinh viên có điểm tổng kết cao
- Xem báo cáo sinh viên nợ tín chỉ nhiều
- Xem báo cáo tỉ lệ sinh viên qua theo môn học của giảng viên

6. Viết tắt cả các test case cho các chức năng:

- Thêm, sửa, xóa thông tin 1 sinh viên
- Thêm, sửa, xóa thông tin 1 nhân viên
- Thêm, sửa, xóa thông tin 1 giảng viên
- Thêm, sửa, xóa thông tin 1 môn học
- Thêm, sửa, xóa thông tin 1 lớp học phần của 1 môn học
- Thêm, sửa, xóa thông tin 1 phòng học
- Lên lịch học cho 1 lớp học phần (ở phòng học nào, vào một khung giờ nào)
- Sửa lịch học cho 1 lớp học phần

- Sinh viên đăng kí 1 môn học
- Sinh viên sửa đăng kí 1 môn học
- Sinh viên xem TKB
- Giảng viên đăng kí dạy 1 lớp học phần
- Giảng viên sửa đăng kí dạy 1 lớp học phần
- Giảng viên xem TKB
- Giảng viên nhập điểm của 1 lớp học phần
- Giảng viên sửa điểm của 1 sinh viên ở 1 môn học
- Xem báo cáo môn học có tỉ lệ điểm cao
- Xem báo cáo sinh viên có điểm tổng kết cao
- Xem báo cáo sinh viên nợ tín chỉ nhiều
- Xem báo cáo tỉ lệ sinh viên qua theo môn học của giảng viên

### 11.3 HỆ THỐNG QUẢN LÍ MUỢN TRẢ SÁCH CỦA THƯ VIỆN QUỐC GIA

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý mượn trả sách của thư viện Quốc gia với mô tả như sau:

- Mỗi đầu sách (Mã, tên, tác giả, năm xuất bản, giá bìa, số lượng, mã vạch, mô tả) có thể được mượn nhiều lần khác nhau bởi nhiều bạn đọc khác nhau
- Mỗi bạn đọc có một thẻ bạn đọc chứa mã, tên, ngày sinh, địa chỉ, số điện thoại, mã vạch của bạn đọc đó
- Mỗi lần mượn được mượn tối đa 5 quyển sách, và tổng số sách đang mượn bởi một người cũng không được quá 5 quyển
- Thời gian tối đa mượn 1 quyển sách là 1 tháng kể từ ngày mượn quyển đó, nếu trả sau thời hạn này thì sẽ bị phạt 20% giá trị bìa sách.
- Mỗi lần trả sách có thể trả một phần hoặc toàn bộ số lượng sách đang mượn
- Khi mượn sách mới, thủ thư vẫn xem được danh sách các sách mà một độc giả đã mượn và trả rồi hoặc chưa trả trước đây.

**Các modul chức năng hoạt động như sau:**

- Modul “**Quản lí thông tin đầu sách**” cho phép quản lí (QL) thực hiện thêm, sửa, xóa thông tin sách với mô tả chi tiết nghiệp vụ: QL chọn menu quản lí sách -> trang quản lí hiện ra -> QL chọn chức năng sửa thông tin sách -> giao diện tìm sách theo tên hiện ra -> QL nhập tên sách và click tìm kiếm -> danh sách các sách có tên chứa từ khóa hiện ra -> QL chọn sửa một sách -> giao diện sửa sách hiện ra với các thông tin của sách đã chọn ->

QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Modul quản lí nhân viên, quản lí độc giả được thực hiện tương tự chức năng quản lí thông tin đầu sách.
- Modul "**Mượn sách**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu cho mượn sách -> quét thẻ độc giả để lấy thông tin độc giả -> thông tin chi tiết độc giả hiện lên + danh sách các sách mượn chưa trả + danh sách sách mượn đã trả -> nhân viên quét lần lượt các sách được chọn mượn -> danh sách sách mượn được bổ sung thêm cho đến khi hết sách chọn mượn (hoặc tối đa 5 quyển) thì submit -> in ra phiếu mượn chứa mã, tên, mã vạch độc giả, mã vạch phiếu mượn, và danh sách sách còn mượn, mỗi đầu sách trên một dòng: mã, tên sách, tác giả, mã vạch, ngày mượn, ngày phải trả và dòng cuối cùng ghi tổng số sách đang mượn.
- Modul "**Trả sách**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu trả sách -> quét thẻ độc giả để lấy thông tin độc giả -> thông tin chi tiết độc giả hiện lên + danh sách các sách mượn chưa trả + danh sách sách mượn đã trả -> nhân viên quét lần lượt các sách được trả -> danh sách sách đang mượn được rút ngắn cho đến khi hết sách mượn (hoặc hết số sách độc giả đem đến trả) thì submit -> in ra phiếu mượn (nếu còn sách mượn) chứa mã, tên, mã vạch độc giả, mã vạch phiếu mượn, và danh sách sách còn mượn, mỗi đầu sách trên một dòng: mã, tên sách, tác giả, mã vạch, ngày mượn, ngày phải trả và dòng cuối cùng ghi tổng số sách đang mượn + phiếu phạt (nếu bị phạt) chứa mã, tên, mã vạch độc giả, mã vạch phiếu mượn, và danh sách sách trả muộn bị phạt, mỗi đầu sách trên một dòng: mã, tên sách, tác giả, mã vạch, ngày mượn, ngày phải trả, ngày trả, số tiền phạt và dòng cuối cùng ghi tổng số tiền phạt.
- Modul "**Thông kê sách mượn nhiều**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu thống kê -> chọn thống kê sách mượn nhiều -> nhập khoảng thời gian (bắt đầu - kết thúc) -> danh sách sách mượn nhiều nhất được hiển thị theo thứ tự số lượt mượn từ nhiều đến ít, mỗi dòng chứa: mã, tên sách, tác giả, mã vạch, tổng số lượt mượn.
- Modul "**Thông kê độc giả mượn nhiều**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu thống kê -> chọn thống kê độc giả mượn nhiều -> nhập khoảng thời gian (bắt đầu - kết thúc) -> danh sách độc giả đã mượn nhiều nhất được hiển thị theo thứ tự số lượng sách mượn từ nhiều đến ít, mỗi dòng chứa: mã, tên, ngày sinh, địa chỉ độc giả, tổng số lượng sách đã mượn.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.

2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đầu sách
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 độc giả
  - Cho 1 độc giả mượn sách
  - Nhận trả sách từ 1 độc giả
  - Xem thông kê đầu sách được mượn nhiều
  - Xem thông kê độc giả mượn nhiều/nợ nhiều đầu sách
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đầu sách
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 độc giả
  - Cho 1 độc giả mượn sách
  - Nhận trả sách từ 1 độc giả
  - Xem thông kê đầu sách được mượn nhiều
  - Xem thông kê độc giả mượn nhiều/nợ nhiều đầu sách
6. Viết tắt cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đầu sách
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 độc giả
  - Cho 1 độc giả mượn sách
  - Nhận trả sách từ 1 độc giả
  - Xem thông kê đầu sách được mượn nhiều
  - Xem thông kê độc giả mượn nhiều/nợ nhiều đầu sách

#### **11.4 HỆ THỐNG QUẢN LÍ XUẤT NHẬP HÀNG TẠI MỘT ĐẠI LÝ TRUNG GIAN**

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lí xuất nhập hàng tại một đại lý trung gian với mô tả như sau:

- Mỗi hàng hóa (Mã hàng, tên, mô tả) có thể được nhập nhiều lần khác nhau, mỗi lần nhập có số lượng khác nhau và giá nhập khác nhau, đến từ một nhà cung cấp (mã NCC, tên NCC, địa chỉ, số ĐT) khác nhau
- Mỗi lần nhập hàng có thể nhập nhiều hàng hóa khác nhau
- Mỗi lần nhập có một phiếu nhập ghi danh sách các mặt hàng nhập vào, mỗi mặt hàng có đầy đủ thông tin: mã hàng, tên hàng, số lượng, đơn giá, nhà cung cấp, thành tiền (tự động tính) và dòng cuối cùng là tổng tiền của hóa đơn nhập
- Tương tự, mỗi hàng hóa có thể xuất đi nhiều lần khác nhau, mỗi lần cho các đại lí con (mã DL, tên DL, địa chỉ, số ĐT) khác nhau, với số lượng khác nhau và giá xuất khác nhau
- Mỗi lần xuất có thể xuất nhiều hàng khác nhau, miễn sao số lượng xuất không vượt quá số lượng hàng còn trong kho
- Mỗi lần xuất có một phiếu xuất ghi danh sách các mặt hàng xuất đi, mỗi mặt hàng có đầy đủ thông tin: mã hàng, tên hàng, số lượng, đơn giá, đại lí con, thành tiền (tự động tính) và dòng cuối cùng là tổng tiền của hóa đơn xuất.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin mặt hàng**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin mặt hàng với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý mặt hàng -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin mặt hàng -> giao diện tìm mặt hàng theo tên hiện ra -> QL nhập tên mặt hàng và click tìm kiếm -> danh sách các mặt hàng có tên chứa từ khóa hiện ra -> QL chọn sửa một mặt hàng -> giao diện sửa mặt hàng hiện ra với các thông tin của mặt hàng đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Modul "**Quản lý nhập hàng**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu nhập hàng -> trang nhập hàng hiện ra -> nhân viên chọn tên hàng trong danh sách hàng hóa có sẵn (nếu hàng mới thì chọn nhập mới) + nhập số lượng + chọn nhà cung cấp trong danh sách có sẵn (nếu nhà cung cấp mới thì nhập mới) đến khi hết các hàng nhập vào thì submit -> báo nhập thành công và in ra hóa đơn nhập như đã mô tả.
- Modul "**Quản lý xuất hàng**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu xuất hàng -> trang xuất hàng hiện ra -> nhân viên chọn tên hàng trong danh sách hàng hóa có sẵn + nhập số lượng + chọn đại lí tiêu thụ trong danh sách có sẵn (nếu là đại lí mới thì nhập mới) đến khi hết các hàng cần xuất vào thì submit -> nếu số lượng hợp lệ thì báo xuất thành công và in ra hóa đơn xuất như đã mô tả.
- Modul "**Thông kê sản phẩm bán chạy**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu thống kê -> chọn chức năng thống kê sản phẩm bán chạy -> nhập khoảng thời gian thống kê (bắt đầu - kết thúc) -> kết quả hiện ra danh sách các sản phẩm theo thứ tự

bán được tổng số lượng nhiều nhất đến ít dần trong khoảng thời gian đã chọn, mỗi dòng có các thông tin: mã hàng, tên hàng, số lượng đã bán được, tổng số tiền đã thu được từ sản phẩm ấy trong khoảng thời gian đã chọn.

- Modul "**Thống kê đại lí tiêu thụ mạnh**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn menu thống kê -> chọn chức năng thống kê đại lí tiêu thụ hàng đầu -> nhập khoảng thời gian thống kê (bắt đầu - kết thúc) -> kết quả hiện ra danh sách các đại lí tiêu thụ theo thứ tự bán được tổng doanh thu nhiều nhất đến ít dần trong khoảng thời gian đã chọn, mỗi dòng có các thông tin: mã đại lí, tên đại lí, tổng số tiền đã thu được từ đại lí ấy trong khoảng thời gian đã chọn.

### **Yêu cầu:**

- Vẽ sơ đồ use case cho toàn bộ hệ thống.
- Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
- Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sản phẩm
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 đại lí con
  - Thêm, sửa, xóa thông tin 1 nhà cung cấp
  - Xuất hàng cho 1 đại lí con
  - Nhập hàng từ 1 nhà cung cấp
  - Xem báo cáo mặt bằng bán chạy
  - Xem báo cáo đại lí con tiêu thụ mạnh
  - Xem báo cáo doanh thu theo tháng
- Thiết kế CSDL cho toàn hệ thống
- Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sản phẩm
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 đại lí con
  - Thêm, sửa, xóa thông tin 1 nhà cung cấp
  - Xuất hàng cho 1 đại lí con
  - Nhập hàng từ 1 nhà cung cấp
  - Xem báo cáo mặt bằng bán chạy
  - Xem báo cáo đại lí con tiêu thụ mạnh

- Xem báo cáo doanh thu theo tháng
6. Viết tất cả các test case cho các chức năng:
- Thêm, sửa, xóa thông tin 1 sản phẩm
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 đại lí con
  - Thêm, sửa, xóa thông tin 1 nhà cung cấp
  - Xuất hàng cho 1 đại lí con
  - Nhập hàng từ 1 nhà cung cấp
  - Xem báo cáo mặt bằng bán chạy
  - Xem báo cáo đại lí con tiêu thụ mạnh
  - Xem báo cáo doanh thu theo tháng

## 11.5 HỆ THỐNG QUẢN LÍ BÁN VÉ TÀU HỎA

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý bán vé tàu hỏa với mô tả như sau:

- Mỗi chuyến tàu (Mã tàu, tên, ga xuất phát, ga cuối cùng) có thể đi qua nhiều ga (Mã ga, địa chỉ, số ĐT, mô tả), và thời gian đến, thời gian đỗ tại mỗi ga khác nhau (chỉnh tính các ga mà chuyến tàu có dừng đón trả khách)
- Mỗi chuyến tàu có nhiều toa (Mã, tên, loại, mô tả). Mỗi toa có nhiều loại chỗ ngồi/năm (Mã, số ghế/giường, mô tả).
- Mỗi chỗ ngồi/năm, trên các chuyến tàu khác nhau, với mỗi chặng khác nhau, có giá vé khác nhau.
- Khách hàng có thể mua vé tương ứng với chỗ ngồi/năm từ một ga đi, đến một ga đến, nếu chỗ đó còn trống trên chuyến tàu trong suốt hành trình từ ga đi đến ga đến đã chọn.
- Cùng một chỗ ngồi/năm, trên cùng chuyến tàu, cùng chặng, thì giá vé khác nhau tại thời điểm bán khác nhau.
- Mỗi khách hàng có thể mua nhiều vé một lần, cho các chuyến tàu và chặng khác nhau, và có thể mua nhiều lần khác nhau.
- Khách hàng có thể trả vé, nếu trả trước giờ tàu chạy ở ga đi trên vé 48h thì phạt 10%, trước 24h phạt 20%, trước giờ tàu chạy phạt 50%, sau khi tàu chạy phạt 100% giá ghi trên vé.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin một ga dừng**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin nhà ga với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý nhà ga -> trang quản

lí hiện ra -> QL chọn chức năng sửa thông tin nhà ga -> giao diện tìm nhà ga theo tên hiện ra -> QL nhập tên nhà ga và click tìm kiếm -> danh sách các nhà ga có tên chứa từ khóa hiện ra -> QL chọn sửa một nhà ga -> giao diện sửa nhà ga hiện ra với các thông tin của nhà ga đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Các modul quản lý thông tin chuyến tàu, toa tàu, hạng ghế, quản lý thông tin khách hàng, quản lý thông tin nhân viên thực hiện tương tự.
- Modul "**Lên lịch chạy**" với mô tả chi tiết nghiệp vụ: QL chọn menu lên lịch chạy cho chuyến tàu → giao diện hiện lên danh sách số các chuyến tàu → QL chọn 1 chuyến tàu để lên lịch → giao diện hiện lên tất cả các ga trung gian từ điểm xuất phát đến điểm đích → QL chọn lần lượt từng ga mà chuyến tàu sẽ dừng lại, với mỗi ga giao diện hiện lên các ô nhập: thời gian vào đến ga, thời gian rời ga. Ga nào không được chọn thì không dừng lại → QL click nút xác nhận → Hệ thống hiển thị lại lịch trình của tàu qua các ga cụ thể với thời gian như thế nào và lưu vào CSDL.
- Modul "**Mua vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng mua vé theo yêu cầu của khách -> giao diện chọn ga đi, ga đến, ngày đi hiện ra -> NV chọn xong bấm tìm -> kết quả hiện ra gồm danh sách các chuyến tàu còn chỗ trống tương ứng với tiêu chí đã chọn, mỗi chuyến tàu liệt kê danh sách các loại ghế/giường còn trống + giá tương ứng tại thời điểm tìm -> NV chọn 1 vé theo lựa chọn của KH -> hóa đơn (vé) hiện ra chi tiết: tên chuyến tàu, ga đi, ga đến, giờ đi, giờ đến, số ghế/giường, loại ghế/giường, số toa, giá vé -> NV chọn thanh toán -> khách hàng thanh toán -> hệ thống lưu kết quả vào và in vé cho khách hàng.
- Modul "**Trả vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng trả vé theo yêu cầu của khách -> giao diện nhập mã vé hiện ra -> NV nhập mã -> kết quả hiện ra vé chi tiết: tên chuyến tàu, ga đi, ga đến, giờ đi, giờ đến, số ghế/giường, loại ghế/giường, số toa, giá vé cho khách hàng in ra -> NV chọn hủy vé -> hệ thống hiện hóa đơn phạt bao gồm thôn tin như trên vé + tiền phạt theo khung quy định -> NV nhấn Ok -> hệ thống lưu kết quả vào hệ thống, và nhân viên gửi lại phần tiền thừa cho khách hàng.
- Modul "**Thông kê chuyến tàu theo doanh thu**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê các chuyến tàu theo doanh thu -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các chuyến tàu chi tiết: mã, tên, ga đầu, ga cuối, số toa, trung bình số ghế/giường, trung bình số vé bán được (tính trên toàn chuyến), trung bình tỉ lệ lấp đầy ghế/giường, trung bình doanh thu. Sắp xếp theo trung bình doanh thu, xếp từ cao đến thấp.

- Modul "**Thống kê các chặng theo doanh thu**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê các chặng theo doanh thu -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các chặng chi tiết: mã, tên, ga đi, ga đến, tổng số lượt ghé/giường, tổng số lượt vé bán được (tính trên chặng), tổng doanh thu. Sắp xếp theo tổng doanh thu, xếp từ cao đến thấp.

**Yêu cầu:**

- Vẽ sơ đồ use case cho toàn bộ hệ thống.
- Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
- Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 ga dừng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 chuyến tàu
  - Lên lịch chạy cho 1 chuyến tàu
  - Sửa lịch chạy cho 1 chuyến tàu
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo chuyến tàu
  - Xem báo cáo doanh thu theo chặng tàu
  - Xem báo cáo doanh thu theo tháng
- Thiết kế CSDL cho toàn hệ thống
- Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 ga dừng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 chuyến tàu
  - Lên lịch chạy cho 1 chuyến tàu
  - Sửa lịch chạy cho 1 chuyến tàu
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo chuyến tàu

- Xem báo cáo doanh thu theo chặng tàu
  - Xem báo cáo doanh thu theo tháng
6. Viết tắt cả các test case cho các chức năng:
- Thêm, sửa, xóa thông tin 1 ga dừng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 chuyến tàu
  - Lên lịch chạy cho 1 chuyến tàu
  - Sửa lịch chạy cho 1 chuyến tàu
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo chuyến tàu
  - Xem báo cáo doanh thu theo chặng tàu
  - Xem báo cáo doanh thu theo tháng

## 11.6 HỆ THỐNG QUẢN LÝ BÁN VÉ MÁY BAY CỦA MỘT ĐẠI LÝ

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý bán vé máy bay với mô tả như sau:

- Mỗi chặng bay được xác định bởi sân bay đi và sân bay đến (Mã sân bay, tên, địa chỉ, thành phố, mô tả)
- Mỗi chặng bay có nhiều chuyến bay (Mã, giờ đi, giờ đến, loại máy bay, mô tả)
- Mỗi chuyến bay có nhiều hạng vé (Mã hạng, tên, mô tả). Mỗi hạng vé có nhiều chỗ ngồi khác nhau.
- Mỗi chuyến bay có lượng vé cho mỗi hạng vé xác định. Khi bán vé chỉ quản lý theo giới hạn này mà chưa gắn số ghế cụ thể cho từng vé. Việc gắn từng khách hàng vào ghế chỉ thực hiện khi khách hàng checkin tại sân bay.
- Mỗi hạng vé, trên các chuyến bay khác nhau, tại thời điểm mua khác nhau, có giá vé khác nhau.
- Mỗi khách hàng có thể mua nhiều vé một lần, cho các chuyến bay và chặng khác nhau, và có thể mua nhiều lần khác nhau. Nhưng mỗi vé phải cung cấp thông tin từng khách hàng cụ thể (Mã, tên, ngày sinh, giới tính, số ID, kiểu giấy tờ ID, quốc tịch).
- Khách hàng có thể trả vé, nếu trả trước giờ bay ghi trên vé 48h thì phạt 10%, trước 24h phạt 20%, trước giờ cất cánh phạt 50%, sau khi cất cánh phạt 100% giá ghi trên vé.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin một sân bay**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin sân bay với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý sân bay -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin sân bay -> giao diện tìm sân bay theo tên hiện ra -> QL nhập tên sân bay và click tìm kiếm -> danh sách các sân bay có tên chứa từ khóa hiện ra -> QL chọn sửa một sân bay -> giao diện sửa sân bay hiện ra với các thông tin của sân bay đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lý thông tin thành phố, quản lý thông tin khách hàng, quản lý thông tin nhân viên được thực hiện tương tự.
- Modul "**Lên lịch bay cho chuyến bay**" với mô tả chi tiết nghiệp vụ: Quản lý (QL) chọn menu lên lịch bay → giao diện hiện lên → QL nhập mã chuyến bay + chọn thành phố xuất phát từ danh sách thành phố sổ xuống + chọn sân bay đi từ danh sách sân bay cập nhật theo thành phố được chọn + chọn thành phố đích đến từ danh sách thành phố sổ xuống + chọn sân bay đến từ danh sách sân bay cập nhật theo thành phố được chọn + nhập giờ bay + chọn loại máy bay từ danh sách các máy bay sổ xuống + nhập giá vé cho chuyến bay theo 4 hạng ghế: thương gia, phổ thông, phổ thông tiết kiệm, siêu tiết kiệm + click thêm lịch bay → Hệ thống thông báo thành công và lưu vào CSDL.
- Modul "**Bán vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng bán vé theo yêu cầu của khách hàng -> giao diện chọn nơi đi - nơi đến hiện ra -> NV hỏi KH và nhập vào -> kết quả hiện ra gồm danh sách các chuyến bay chi tiết: mã, tên, giờ đi, giờ đến, mỗi chuyến bay có danh sách giá vé tương ứng với từng hạng vé còn ghế. -> KH chọn 1 chuyến bay và NV click chọn -> Yêu cầu nhập số lượng vé và thông tin KH -> Sau khi nhập xong, thông tin vé (hóa đơn) hiện ra từng vé một, mỗi vé có đầy đủ thông tin: mã vé riêng, tên KH, sân bay đi, sân bay đến, giờ đi, giờ đến, hạng vé, giá vé, và dòng cuối ghi tổng số tiền thanh toán cho cả hóa đơn -> NV click xác nhận -> hệ thống lưu lại, KH thanh toán cho NV, hệ thống in hóa đơn (vé) cho KH.
- Modul "**Trả vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng trả vé theo yêu cầu của khách -> giao diện nhập mã vé hiện ra -> NV nhập xong bấm thống kê -> kết quả hiện ra gồm danh sách các vé chi tiết: mã, sân bay đi, sân bay đến, giờ đi, giờ đến, máy bay, hạng vé, giá vé -> NV chọn hủy vé -> Giao diện hiện hóa đơn phát bao gồm các thông tin như trên vé + tổng số tiền phạt -> NV click xác nhận -> Hệ thống lưu thông tin và NV trả lại tiền thừa cho khách hàng đã mua.
- Modul "**Thống kê lượng khách theo khung giờ**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê lượng khách theo khung giờ -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra

gồm danh sách các khung giờ chi tiết: khung giờ từ máy giờ đến máy giờ trong ngày, trung bình số lượt ghế, trung bình số lượt vé bán được (tính trên chặng), trung bình doanh thu. Sắp xếp theo trung bình lượt khách, xếp từ cao đến thấp.

- Modul "**Thống kê các chặng bay theo doanh thu**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê các chặng theo doanh thu -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các chặng chi tiết: mã, tên, sân bay đi, sân bay đến, tổng số lượt ghế, tổng số lượt vé bán được (tính trên chặng), tổng doanh thu. Sắp xếp theo trung bình doanh thu, xếp từ cao đến thấp.

### **Yêu cầu:**

- Vẽ sơ đồ use case cho toàn bộ hệ thống.
- Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
- Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sân bay
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 thành phố
  - Lên lịch bay cho 1 chuyến bay
  - Sửa lịch bay cho 1 chuyến bay
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo khung giờ bay
  - Xem báo cáo doanh thu theo chặng bay
  - Xem báo cáo doanh thu theo tháng
- Thiết kế CSDL cho toàn hệ thống
- Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sân bay
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 thành phố
  - Lên lịch bay cho 1 chuyến bay
  - Sửa lịch bay cho 1 chuyến bay

- Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo khung giờ bay
  - Xem báo cáo doanh thu theo chặng bay
  - Xem báo cáo doanh thu theo tháng
6. Viết tất cả các test case cho các chức năng:
- Thêm, sửa, xóa thông tin 1 sân bay
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 thành phố
  - Lên lịch bay cho 1 chuyến bay
  - Sửa lịch bay cho 1 chuyến bay
  - Bán vé cho 1 khách hàng
  - Nhận trả vé từ một khách hàng
  - Xem báo cáo doanh thu theo khung giờ bay
  - Xem báo cáo doanh thu theo chặng bay
  - Xem báo cáo doanh thu theo tháng

## 11.7 HỆ THỐNG QUẢN LÍ ĐẶT BÀN VÀ GỌI MÓN TRONG MỘT NHÀ HÀNG

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý việc gọi món trong một nhà hàng, bao gồm các chức năng cơ bản:

- Nhà hàng có nhiều bàn (Mã bàn, tên, số lượng khách tối đa, mô tả). Nhiều bàn nhỏ có thể gộp lại thành một bàn lớn khi có yêu cầu từ đoàn khách có số lượng lớn.
- Mỗi bàn, có thể bị đặt nhiều lần khác nhau trong ngày, hoặc khác ngày.
- Mỗi khách hàng (Mã, tên, số ĐT, email, địa chỉ) có thể đặt bàn nhiều lần, mỗi lần có thể đặt nhiều bàn (trường hợp này sẽ bị gộp thành đặt 1 bàn)
- Khách hàng ở mỗi bàn có thể gọi nhiều món ăn (Mã, loại, tên, mô tả, giá hiện tại). Mỗi món ăn có có thể bị gọi với số lượng khác nhau.

**Các modul chức năng hoạt động như sau:**

- Modul "*Quản lý thông tin món ăn*" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin món ăn với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý món ăn -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin món ăn -> giao diện tìm món ăn theo tên hiện ra -> QL nhập tên món ăn và click tìm kiếm -> danh sách các món ăn có tên chứa từ khóa

hiện ra -> QL chọn sửa một món ăn -> giao diện sửa món ăn hiện ra với các thông tin của món ăn đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Các modul quản lý thông tin bàn, quản lý thông tin khách hàng, quản lý thông tin nhân viên được thực hiện tương tự.
- Modul "**Đặt bàn**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng đặt bàn khi khách hàng gọi đến -> giao diện tìm bàn trống hiện ra -> NV nhập ngày + giờ đặt + số lượng khách và bấm tìm -> kết quả hiện ra gồm danh sách các bàn còn trống vào ngày giờ đấy: mã, tên, số lượng khách tối đa, mô tả -> NV chọn 1 bàn theo yêu cầu của KH -> Giao diện nhập thông tin KH hiện ra -> NV hỏi khách hàng và nhập mã, tên, số ĐT, email, địa chỉ và click tìm -> Hệ thống hiện danh sách các khách hàng có cùng tên vừa nhập, mỗi khách hàng trên 1 dòng: mã, tên, số ĐT, email, địa chỉ → NV click vào dòng đúng với KH đang đặt (nếu không có thìlick thêm KH mới) → Hệ thống hiện lên giao diện xác nhận có đầy đủ thông tin bàn + thông tin KH + ngày giờ đặt → NV xác nhận với KH và click xác nhận → Hệ thống lưu thông tin vào CSDL.
- Modul "**Gọi món**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng gọi món -> giao diện bàn hiện ra với danh sách bàn và số hiệu sổ xuống -> NV chọn bàn đúng với KH đang gọi món → Giao diện nhập món được gọi hiện ra → NV hỏi KH và nhập vào tên món ăn + chọn tìm -> kết quả hiện ra gồm danh sách các món ăn chi tiết: mã, loại, tên, giá. -> NV chọn 1 món ăn đúng như KH gọi và NV click chọn -> Yêu cầu nhập số lượng -> NV nhập số lượng và click OK → Tên món ăn + số lượng + số tiền tạm tính được thêm vào danh sách các món ăn đã chọn phía dưới. NV lặp lại các bước chọn món ăn này cho đến khi nhập vào được hết các món mà khách hàng trong bàn đã gọi. NV đọc lại để xác nhận với KH -> NV click xác nhận -> hệ thống lưu lại.
- Modul "**Thống kê lượng khách theo khung giờ**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê lượng khách theo khung giờ -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các khung giờ chi tiết: khung giờ từ mấy giờ đến mấy giờ trong ngày, trung bình số lượng khách, trung bình doanh thu/đầu khách, tổng doanh thu của khung giờ. Sắp xếp theo tổng doanh thu, xếp từ cao đến thấp.
- Modul "**Thống kê doanh thu theo tháng**" cho phép quản lý (NV) nhà hàng thống kê doanh thu bán hàng với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê -> chọn thống kê doanh thu theo món ăn -> nhập thời gian bắt đầu và kết thúc thống kê -> danh sách các món ăn có doanh thu lớn nhất được hiện ra, mỗi dòng cho 1 món ăn: Mã, tên, tổng số

lượng khách đã dùng, tổng doanh thu thu được, sắp xếp theo chiều giảm dần tổng doanh thu.

- Modul "**Thống kê món ăn bán chạy**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê món ăn bán chạy -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các món ăn chi tiết: mã, loại, tên, tổng số lượt bán, tổng doanh thu. Sắp xếp theo tổng doanh thu, xếp từ cao đến thấp.

#### **Yêu cầu:**

- Vẽ sơ đồ use case cho toàn bộ hệ thống.
- Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
- Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 món ăn
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 bàn
  - Nhận đặt bàn từ 1 khách hàng
  - Sửa thông tin đặt bàn từ 1 khách hàng
  - Nhận gọi món từ 1 bàn
  - Thanh toán cho 1 bàn
  - Xem thống kê món ăn bán chạy
  - Xem thống kê số lượng khách theo khung giờ
  - Xem báo cáo doanh thu theo tháng
- Thiết kế CSDL cho toàn hệ thống
- Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 món ăn
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 bàn
  - Nhận đặt bàn từ 1 khách hàng
  - Sửa thông tin đặt bàn từ 1 khách hàng
  - Nhận gọi món từ 1 bàn
  - Thanh toán cho 1 bàn
  - Xem thống kê món ăn bán chạy

- Xem thống kê số lượng khách theo khung giờ
  - Xem báo cáo doanh thu theo tháng
6. Viết tắt cả các test case cho các chức năng:
- Thêm, sửa, xóa thông tin 1 món ăn
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Thêm, sửa, xóa thông tin 1 bàn
  - Nhận đặt bàn từ 1 khách hàng
  - Sửa thông tin đặt bàn từ 1 khách hàng
  - Nhận gọi món từ 1 bàn
  - Thanh toán cho 1 bàn
  - Xem thống kê món ăn bán chạy
  - Xem thống kê số lượng khách theo khung giờ
  - Xem báo cáo doanh thu theo tháng

## 11.8 HỆ THỐNG QUẢN LÍ TOUR DU LỊCH DẠNG OPEN TOUR

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý bán vé tour du lịch với mô tả như sau:

- Mỗi tour (Mã tour, tên, nơi xuất phát, nơi đến, mô tả) có thể xuất phát vào nhiều ngày khác nhau, tùy vào ngày xuất phát và số lượng người mua tour cho mỗi đoàn sẽ có giá khác nhau.
- Mỗi khách hàng (Mã, tên, số ID, loại thẻ ID, số ĐT, email, địa chỉ) có thể mua vé nhiều tour khác nhau. Mỗi tour có thể mua số lượng vé khác nhau. Mỗi lần mua có xuất hóa đơn ghi rõ thông tin tour, ngày xuất phát, giá tour, số lượng khách, tên khách hàng đại diện, tổng số tiền thanh toán.
- Cùng một khách hàng có thể đi cùng một tour nhiều lần, chỉ khác nhau ở ngày xuất phát và giá vé.
- Khách hàng có thể trả vé, nếu trả trước giờ xuất phát trước 7 ngày thì phạt 10%, trước 5 ngày phạt 20%, trước 3 ngày phạt 50%, trước ít hơn 3 ngày phạt 100% giá trên vé.

**Các modul chức năng hoạt động như sau:**

- Modul "*Quản lý thông tin tour*" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin tour với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý tour -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin tour -> giao diện tìm tour theo tên hiện ra -> QL nhập tên tour và click tìm kiếm -> danh sách các tour có tên chứa từ khóa hiện ra -> QL chọn

sửa một tour -> giao diện sửa tour hiện ra với các thông tin của tour đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Các modul quản lý thông tin khách hàng, quản lý thông tin nhân viên được thực hiện tương tự.
- Modul "**Mua vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng mua vé theo yêu cầu của khách -> giao diện tìm tour (theo tên nơi đến) -> NV nhập tên nơi đến và bấm tìm -> kết quả hiện ra gồm danh sách các tour còn chỗ trống tương ứng với tiêu chí đã chọn, mỗi tour hiển thị đầy đủ thông tin + ngày xuất phát + giá tương ứng tại thời điểm tìm -> NV chọn 1 tour theo lựa chọn của KH -> hóa đơn (vé) hiện ra chi tiết: tên tour, nơi đi, nơi đến, ngày đi, tên khách đại diện đoàn, số ID, kiểu ID, địa chỉ khách, số điện thoại, email, số lượng khách, giá vé -> NV chọn thanh toán -> khách hàng thanh toán -> hệ thống lưu kết quả vào và in vé cho khách hàng.
- Modul "**Trả vé**" với mô tả chi tiết nghiệp vụ: Nhân viên chọn chức năng trả vé theo yêu cầu của khách -> giao diện nhập mã vé hiện ra -> NV nhập mã -> kết quả hiện ra vé chi tiết: tên tour, nơi đi, nơi đến, ngày đi, tên khách đại diện đoàn, số ID, kiểu ID, địa chỉ khách, số điện thoại, email, số lượng khách, giá vé -> NV chọn hủy vé -> hệ thống hiện hóa đơn phạt bao gồm thông tin như trên vé + tiền phạt theo khung quy định -> NV nhấn Ok -> hệ thống lưu kết quả vào hệ thống, và nhân viên gửi lại phần tiền thừa cho khách hàng.
- Modul "**Thống kê tour theo doanh thu**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê các tour theo doanh thu -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các tour chi tiết: mã, tên, tên, nơi xuất phát, nơi đến, trung bình số khách/tour, tổng doanh thu. Sắp xếp theo tổng doanh thu, xếp từ cao đến thấp.
- Modul "**Thống kê doanh thu theo địa điểm**" với mô tả chi tiết nghiệp vụ: Quản lý chọn chức năng thống kê doanh thu theo địa điểm du lịch -> giao diện chọn thời gian thống kê (ngày bắt đầu - kết thúc) hiện ra -> quản lý chọn xong bấm thống kê -> kết quả hiện ra gồm danh sách các địa điểm chi tiết: tên, số lượng tour đến địa điểm đấy, tổng số lượng khách đến địa điểm đấy, tổng doanh thu. Sắp xếp theo tổng doanh thu, xếp từ cao đến thấp.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.

3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 tour
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Bán vé tour cho khách
  - Sửa thông tin đặt tour của khách
  - Xem thông kê tour theo doanh thu
  - Xem thống kê doanh thu theo địa điểm
  - Xem báo cáo doanh thu theo tháng
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 tour
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Bán vé tour cho khách
  - Sửa thông tin đặt tour của khách
  - Xem thông kê tour theo doanh thu
  - Xem thống kê doanh thu theo địa điểm
  - Xem báo cáo doanh thu theo tháng
6. Viết tất cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 tour
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Bán vé tour cho khách
  - Sửa thông tin đặt tour của khách
  - Xem thông kê tour theo doanh thu
  - Xem thống kê doanh thu theo địa điểm
  - Xem báo cáo doanh thu theo tháng

## 11.9 HỆ THỐNG QUẢN LÍ GIẢI VÔ ĐỊCH BÓNG ĐÁ QUỐC GIA

Liên đoàn bóng đá Quốc gia đặt hàng anh/chị phát triển một phần mềm quản lý kết quả thi đấu của Giải ngoại hạng với mô tả như sau:

- Giải có nhiều đội bóng tham dự (Mã đội, tên, trang phục truyền thống, địa chỉ, mô tả).
- Giải có nhiều vòng đấu, mỗi vòng đấu các đội thi đấu theo cặp.
- Mỗi đội có nhiều cầu thủ (mã, tên, năm sinh, vị trí, chân thuận, chiều cao, cân nặng, đặc điểm). Trước mỗi trận đấu diễn ra các đội phải đăng ký các cầu thủ cho trận đấu tương ứng
- Trong khi trận đấu đang diễn ra, người quản lý có thể bổ sung các thông tin chi tiết cho trận đấu đó: các cầu thủ nào tham gia, mỗi cầu thủ vào sân từ phút bao nhiêu, thay ra ở phút bao nhiêu. Cầu thủ nào bị thẻ đỏ, thẻ vàng và ở phút bao nhiêu. Cầu thủ nào kiến tạo, cầu thủ nào ghi bàn, ở phút bao nhiêu. Cầu thủ nào bị thổi phạt lỗi, ở phút bao nhiêu.
- Sau trận đấu, thứ hạng các đội trong bảng đấu được tự động cập nhật. Thắng 3 điểm, hòa 1 điểm, thua 0 điểm. Thứ hạng các đội xếp theo tiêu chí tổng điểm, hiệu số thắng thua giữa các đội cùng điểm số, số bàn thắng giữa các đội cùng điểm số, hiệu số bàn thắng của đội trong bảng, tổng bàn thắng của đội trong bảng. Chỉ có 2 đội đầu mỗi bảng được đi tiếp vào vòng sau.

#### **Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin cầu thủ**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin cầu thủ với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý cầu thủ -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin cầu thủ -> giao diện tìm cầu thủ theo tên hiện ra -> QL nhập tên cầu thủ và click tìm kiếm -> danh sách các cầu thủ có tên chứa từ khóa hiện ra -> QL chọn sửa một cầu thủ -> giao diện sửa cầu thủ hiện ra với các thông tin của cầu thủ đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lý thông tin đội bóng, quản lý thông tin trọng tài, quản lý thông tin sân vận động, quản lý thông tin nhân viên, quản lý thông tin khách hàng mua vé được thực hiện tương tự.
- Modul "**Lên lịch thi đấu**" cho phép ban tổ chức (BTC) lên lịch thi đấu cho từng trận đấu với mô tả chi tiết nghiệp vụ: BTC chọn menu lên lịch → trang chọn vòng đấu hiện ra → BTC chọn vòng đấu trong danh sách sổ ra + click Next → Giao diện lên lịch trận đấu hiện ra → BTC chọn đội 1 và đội thứ 2 trong 2 danh sách sổ ra + chọn ngày giờ và sân đấu + click Lên lịch → Hệ thống thông báo lịch cặp đấu giữa đội . Và đội 1 và đội 2 ở sân vận động đã chọn, vào ngày giờ đã chọn, đã được thêm vào hệ thống
- Modul "**Đăng ký danh sách cầu thủ cho trận đấu**" cho phép ban tổ chức (BTC) đăng ký danh sách cầu thủ thi đấu cho từng trận đấu với mô tả chi tiết nghiệp vụ: BTC chọn menu đăng ký cầu thủ → trang chọn vòng đấu hiện ra → BTC chọn vòng đấu trong danh sách sổ ra + click Next → Giao diện lên lịch trận đấu hiện ra → BTC chọn 1 trong 2 đội bóng

trong danh sách số ra + click Danh sách cầu thủ → Danh sách cầu thủ toàn đội hiện ra → BTC click chọn các cầu thủ đăng kí + click đang kí → Hệ thống thông báo danh sách đăng kí 16 cầu thủ của đội đã được thêm vào hệ thống

- Modul "**Cập nhật kết quả thi đấu**" cho phép ban tổ chức (BTC) cập nhật kết quả cho từng trận đấu với mô tả chi tiết nghiệp vụ: BTC chọn menu cập nhật kết quả → trang chọn vòng đấu hiện ra → BTC chọn vòng đấu trong danh sách số ra + chọn trận đấu trong danh sách số ra + click Next → Giao diện cập nhật kết quả trận đấu hiện ra với danh sách đăng kí của 2 đội ở 2 cột → khi có sự kiện bàn thắng, thẻ phạt, thay người xảy ra, BTC chọn tên cầu thủ tương ứng + click Cập nhật → Giao diện cập nhật sự kiện cho cầu thủ hiện ra → BTC click chọn bàn thắng/thẻ đỏ(vàng)/lỗi, thay + click Thêm vào → Hệ thống thông báo thêm sự kiện tương ứng thành công cho cầu thủ đó.
- Modul "**Xem BXH đội bóng**" cho phép Ban tổ chức (BTC) thống kê và sắp xếp các đội bóng theo thành tích tại giải với mô tả chi tiết nghiệp vụ: BTC chọn menu thống kê → chọn thống kê đội bóng theo thành tích → danh sách các đội bóng hiện ra, mỗi dòng cho 1 đội bóng: Mã, tên, thứ hạng chung cuộc của đội bóng, tổng số điểm, tổng số bàn thắng, tổng số bàn thua, hiệu số thắng thua, tổng số trận thi đấu. Sắp xếp theo thứ tự lần lượt chiều tăng dần thứ hạng, giảm dần tổng điểm, giảm dần tổng số bàn thắng, tăng dần của tổng bàn thua, giảm dần tổng số trận thi đấu.
- Modul "**Xem BXH các cầu thủ ghi nhiều bàn thắng**" cho phép Ban tổ chức (BTC) thống kê và sắp xếp các cầu thủ theo số bàn thắng để trao giải Goldshoes với mô tả chi tiết nghiệp vụ: BTC chọn menu thống kê → chọn thống kê bàn thắng theo cầu thủ → danh sách các cầu thủ ghi bàn hiện ra, mỗi dòng cho 1 cầu thủ: Mã, tên cầu thủ, số áo, tên đội, tổng số bàn thắng, tổng số pha kiến tạo thành bàn, thứ hạng chung cuộc của đội bóng, tổng số thời gian thi đấu. Sắp xếp theo thứ tự lần lượt chiều giảm dần tổng số bàn thắng, giảm dần của tổng số pha kiến tạo, tăng dần của thứ hạng đội bóng, tăng dần tổng thời gian thi đấu. Cầu thủ xếp đầu bảng sẽ được giải Goldshoes.
- Modul "**Thống kê thẻ phạt các đội bóng**" cho phép Ban tổ chức (BTC) thống kê và sắp xếp các đội theo số thẻ phạt để trao giải Fairplay với mô tả chi tiết nghiệp vụ: BTC chọn menu thống kê → chọn thống kê thẻ phạt theo đội → danh sách các đội bị thẻ phạt hiện ra, mỗi dòng cho 1 đội: Mã, tên đội, tổng số thẻ đỏ, tổng số thẻ vàng, tổng số lỗi bị thổi phạt. Sắp xếp theo chiều giảm dần tổng số thẻ đỏ, nếu số thẻ đỏ bằng nhau thì sắp xếp theo chiều giảm dần của tổng số thẻ vàng, nếu số thẻ vàng bằng nhau thì xếp theo chiều giảm dần tổng số lỗi. Đội xếp cuối bảng sẽ được giải Fairplay.

**Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đội bóng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 cầu thủ của đội bóng
  - Thêm, sửa, xóa thông tin 1 trọng tài
  - Lên lịch thi đấu cho 1 vòng đấu
  - Sửa lịch thi đấu cho 1 vòng đấu
  - Đăng ký cầu thủ thi đấu cho 1 đội bóng trước 1 trận đấu
  - Cập nhật các sự kiện trong 1 trận đấu: bị phạt, phạt thẻ, ghi bàn, thay người
  - Xem BXH các đội bóng sau 1 vòng đấu
  - Xem BXH các cầu thủ ghi nhiều bàn thắng
  - Xem BXH các đội chơi xấu/chơi đẹp
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đội bóng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 cầu thủ của đội bóng
  - Thêm, sửa, xóa thông tin 1 trọng tài
  - Lên lịch thi đấu cho 1 vòng đấu
  - Sửa lịch thi đấu cho 1 vòng đấu
  - Đăng ký cầu thủ thi đấu cho 1 đội bóng trước 1 trận đấu
  - Cập nhật các sự kiện trong 1 trận đấu: bị phạt, phạt thẻ, ghi bàn, thay người
  - Xem BXH các đội bóng sau 1 vòng đấu
  - Xem BXH các cầu thủ ghi nhiều bàn thắng
  - Xem BXH các đội chơi xấu/chơi đẹp
6. Viết tất cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đội bóng
  - Thêm, sửa, xóa thông tin 1 nhân viên
  - Thêm, sửa, xóa thông tin 1 cầu thủ của đội bóng
  - Thêm, sửa, xóa thông tin 1 trọng tài
  - Lên lịch thi đấu cho 1 vòng đấu

- Sửa lịch thi đấu cho 1 vòng đấu
- Đăng ký cầu thủ thi đấu cho 1 đội bóng trước 1 trận đấu
- Cập nhật các sự kiện trong 1 trận đấu: bị phạt, phạt thẻ, ghi bàn, thay người
- Xem BXH các đội bóng sau 1 vòng đấu
- Xem BXH các cầu thủ ghi nhiều bàn thắng
- Xem BXH các đội chơi xấu/choi đẹp

## 11.10 HỆ THỐNG QUẢN LÝ LỊCH THI ĐẤU VÀ KẾT QUẢ GIẢI VÔ ĐỊCH CỜ VUA THẾ GIỚI

Liên đoàn cờ vua thế giới (FIDE) yêu cầu anh/chị phát triển một phần mềm quản lý Giải đấu vô địch thế giới với mô tả như sau:

- Mỗi giải đấu (Mã, tên, năm, lần tổ chức, địa điểm, mô tả) cho phép nhiều cờ thủ (mã, tên, năm sinh, quốc tịch, hệ số Elo, ghi chú) tham gia.
- Có thể có hàng trăm cờ thủ tham gia, nhưng mỗi cờ thủ phải thi đấu 11 trận theo hệ Thụy Sỹ
- Ở ván thứ nhất, các cờ thủ được xếp hạng theo thứ tự hệ số Elo từ cao đến thấp. Sau đó đi từ trên xuống dưới bảng xếp hạng, hai cờ thủ đứng kề nhau sẽ tạo thành một cặp đấu cho vòng 1.
- Ở mỗi vòng đấu, thắng được 1 điểm, hòa được 0.5 điểm, thua được 0 điểm. Sau mỗi vòng đấu, kết quả từng trận được cập nhật theo các cặp đấu đã lên lịch trước đó. Đồng thời hệ số Elo tăng hay giảm sau mỗi vòng đấu cũng được cập nhật (Tính theo công thức của FIDE, chỉ cần nhập kết quả vào).
- Bắt đầu từ ván thứ 2, bảng xếp hạng tạm thời sau vòng đấu trước đó được xếp theo thứ tự các tiêu chí: tổng điểm (giảm dần), tổng điểm của các đối thủ đã gặp (giảm dần), hệ số Elo (giảm dần). Và cặp đấu được xác định như sau, đi từ đầu đến cuối bảng xếp hạng tạm thời, với mỗi cờ thủ chưa có cặp, đối thủ cả cờ thủ đó là cờ thủ đầu tiên gặp phải và thỏa mãn: chưa có cặp, và chưa gặp cờ thủ đang xét.
- Sau 11 vòng đấu như vậy, cờ thủ đứng đầu bảng xếp hạng sẽ là nhà vô địch.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin kí thủ**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin kí thủ với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý kí thủ -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin kí thủ -> giao diện tìm kí thủ theo tên hiện ra -> QL nhập tên kí thủ và click tìm kiếm -> danh sách các kí thủ có tên chứa từ khóa hiện ra -> QL chọn sửa một kí thủ -> giao diện sửa kí thủ hiện ra với các thông tin của kí thủ đã

chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Modul "**Cập nhật kết quả**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn menu cập nhật kết quả → trang cập nhật kết quả hiện ra → BTC chọn vòng đấu từ danh sách số ra + chọn cặp đấu từ danh sách số ra theo vòng đấu + nhập số điểm và điểm Elo cho 2 cờ thủ của trận đấu + click Cập nhật → Hệ thống thông báo lưu thành công kết quả trận đấu và quay về trang chọn vòng đấu + trận đấu.
- Modul "**Xem bảng xếp hạng**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn menu thống kê → chọn chức năng xem bảng xếp hạng sau từng vòng đấu → chọn vòng đấu trong danh sách số ra → kết quả hiện ra danh sách các cờ thủ, mỗi người có đầy đủ thông tin: id, tên, năm sinh, quốc tịch, tổng điểm, tổng điểm đối thủ đã gấp, hệ số Elo tức thời. Sắp xếp theo thứ tự đã mô tả ở trên.
- Modul "**Xếp cặp thi đấu**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn menu xếp cặp thi đấu → trang xếp cặp thi đấu hiện ra → BTC chọn vòng đấu trước đó trong danh sách số xuống → hệ thống hiện bảng xếp hạng hiện tại sau vòng đấu trước đó + nút Xếp lịch → BTC click nút Xếp lịch → Hệ thống tự động xếp cặp cho các cờ thủ theo luật mô tả ở trên, và hiện danh sách các bàn đấu theo đúng thứ tự các cặp đấu → BTC click Lưu → Hệ thống lưu lịch thi đấu của vòng mới vào CSDL
- Modul "**Thống kê thay đổi Elo**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn menu thống kê → chọn chức năng thống kê thay đổi Elo của các cờ thủ sau giải → kết quả hiện ra danh sách các cờ thủ, mỗi cờ thủ được hiện đầy đủ thông tin: mã, tên, năm sinh, quốc tịch, hệ số Elo cũ, hệ số Elo mới, hệ số Elo đã tăng/giảm. Sắp xếp theo thứ tự giảm dần của mức tăng giảm hệ số Elo của các kì thi, tiếp đến là giảm dần của hệ số Elo mới, sau giải.

### *Yêu cầu:*

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 kì thi
  - Xem bảng xếp hạng sau 1 vòng đấu
  - Phân cặp thi đấu cho lượt đấu tiếp theo
  - Cập nhật kết quả của 1 vòng đấu
  - Xem thống kê thay đổi hệ số Elo sau giải

4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 kì thi
  - Xem bảng xếp hạng sau 1 vòng đấu
  - Phân cặp thi đấu cho lượt đấu tiếp theo
  - Cập nhật kết quả của 1 vòng đấu
  - Xem thống kê thay đổi hệ số Elo sau giải
6. Viết tắt cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 kì thi
  - Xem bảng xếp hạng sau 1 vòng đấu
  - Phân cặp thi đấu cho lượt đấu tiếp theo
  - Cập nhật kết quả của 1 vòng đấu
  - Xem thống kê thay đổi hệ số Elo sau giải

## 11.11 HỆ THỐNG QUẢN LÍ LỊCH THI ĐUA VÀ KẾT QUẢ GIẢI ĐUA CÔNG THỨC 1 F1

Ban tổ chức đua xe công thức 1 (F1) đặt hàng anh/chị phát triển một phần mềm quản lý kết quả giải đua với mô tả như sau:

- Mỗi năm có một giải. Một giải bao gồm nhiều chặng đua diễn ra trên khắp thế giới (Mã chặng, tên, số vòng đua, địa điểm, thời gian, mô tả).
- Mỗi giải có nhiều đội đua tham gia (Mã, tên, hăng, mô tả).
- Mỗi đội đua có nhiều tay đua (mã, tên, ngày sinh, quốc tịch, tiểu sử). Nhưng ở mỗi chặng đua, mỗi đội chỉ được phép cho tối đa 2 tay đua tham dự.
- Mỗi chặng đua, kết quả xếp theo thứ tự về đích (thời gian) và điểm số chỉ được tính cho top 10 người về đích sớm nhất, lần lượt theo các thứ tự về đích là 25, 18, 15, 12, 10, 8, 6, 4, 2, 1.
- Nếu tay đua nằm trong top 10 nhưng không về đích do bỏ cuộc hoặc tai nạn thì 0 điểm.
- Điểm số và thời gian của từng tay đua sẽ được cộng dồn giữa các chặng để quyết định giải cá nhân và giải đồng đội của mùa giải.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý tay đua**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin tay đua với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý tay đua -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin tay đua -> giao diện tìm tay đua theo tên hiện ra -> QL

nhập tên tay đua và click tìm kiếm -> danh sách các tay đua có tên chứa từ khóa hiện ra -> QL chọn sửa một tay đua -> giao diện sửa tay đua hiện ra với các thông tin của tay đua đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.

- Các modul quản lý hàng đua, quản lý đội đua, quản lý người dùng hệ thống được thực hiện tương tự.
- Modul "**Đăng ký thi đấu**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn chức năng đăng ký tay đua → giao diện đăng ký tay đua cho mỗi chặng đấu hiện ra → BTC chọn chặng đua từ danh sách sổ xuống + chọn đội đua từ danh sách sổ xuống → danh sách các tay đua của đội đua đã chọn hiện ra, xếp theo abc của họ tên → BTC tích chọn đúng 2 tay đua theo yêu cầu của đội + click Đăng ký → Hệ thống lưu thông tin và thông báo thành công.
- Modul "**Cập nhật kết quả**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn chức năng nhập kết quả chặng đua → giao diện nhập kết quả hiện ra → BTC chọn tên chặng đua từ danh sách sổ xuống → Danh sách các tay đua đã đăng ký thi đấu cho chặng đua hiện ra dưới dạng bảng, mỗi dòng chứa các ô trống nhập thời gian về đích, số vòng đua hoàn thành → BTC nhập đầy đủ kết quả tất cả các tay đua và click Lưu → Hệ thống lưu kết quả vào CSDL và thông báo thành công.
- Modul "**Xem BXH các tay đua**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn chức năng thống kê → Chọn xem bảng xếp hạng các tay đua hiện tại → Hệ thống hiện lên danh sách các tay đua theo dạng bảng, mỗi dòng chứa: Tên tay đua, quốc tịch, tên đội đua, tổng điểm sau các chặng, tổng thời gian sau các chặng. Kết quả sắp xếp theo thứ tự giảm dần của tổng điểm, sau đó là thứ tự tăng dần tổng thời gian.
- Modul "**Xem BXH các đội đua**" với mô tả chi tiết nghiệp vụ: Ban tổ chức (BTC) chọn chức năng thống kê → Chọn xem bảng xếp hạng các đội đua hiện tại → Hệ thống hiện lên danh sách các đội đua theo dạng bảng, mỗi dòng chứa: Tên đội đua, hảng, tổng điểm các tay đua của đội sau các chặng, tổng thời gian sau các chặng. Kết quả sắp xếp theo thứ tự giảm dần của tổng điểm, sau đó là thứ tự tăng dần tổng thời gian.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 tay đua

- Thêm, sửa, xóa thông tin 1 đội đua
  - Đăng ký 2 tay đua cho 1 đội đua trước 1 chặng đua
  - Cập nhật kết quả 1 chặng đua
  - Xem BXH các tay đua sau 1 chặng đua
  - Xem BXH các đội đua sau 1 chặng đua
4. Thiết kế CSDL cho toàn hệ thống
  5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
    - Thêm, sửa, xóa thông tin 1 tay đua
    - Thêm, sửa, xóa thông tin 1 đội đua
    - Đăng ký 2 tay đua cho 1 đội đua trước 1 chặng đua
    - Cập nhật kết quả 1 chặng đua
    - Xem BXH các tay đua sau 1 chặng đua
    - Xem BXH các đội đua sau 1 chặng đua
  6. Viết tắt cả các test case cho các chức năng:
    - Thêm, sửa, xóa thông tin 1 tay đua
    - Thêm, sửa, xóa thông tin 1 đội đua
    - Đăng ký 2 tay đua cho 1 đội đua trước 1 chặng đua
    - Cập nhật kết quả 1 chặng đua
    - Xem BXH các tay đua sau 1 chặng đua
    - Xem BXH các đội đua sau 1 chặng đua

## 11.12 HỆ THỐNG QUẢN LÍ CHO THUÊ TRUYỆN CỦA CỬA HÀNG CHO THUÊ TRUYỆN

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý cho thuê truyện ở một cửa hàng chuyên cho thuê truyện với mô tả như sau:

- Cửa hàng có nhiều đầu truyện khác nhau. Mỗi đầu truyện có số lượng khác nhau và giá thuê khác nhau (giá thuê theo ngày).
- Mỗi đầu truyện có thể được mượn bởi nhiều khách hàng khác nhau. Mỗi khách hàng mỗi lần mượn được mượn nhiều đầu truyện khác nhau.
- Mỗi lần mượn, khách hàng được nhận một phiếu mượn. Trong đó, dòng đầu ghi tên khách hàng và ngày mượn. Thông tin mỗi đầu truyện mượn được ghi trên một dòng: tên, tác giả, nhà xuất bản, năm xuất bản, giá thuê. Dòng cuối cùng ghi số lượng đầu truyện mượn.

- Khi trả truyện, khách hàng được nhận hóa đơn trả. Trong đó, dòng đầu ghi tên khách hàng và ngày thanh toán. Thông tin mỗi đầu truyện trả được ghi trên một dòng: tên, tác giả, nhà xuất bản, năm xuất bản, ngày mượn, ngày trả, giá thuê, thành tiền. Dòng cuối cùng ghi tổng số tiền thanh toán.

#### **Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý đầu truyện**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin đầu truyện với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý đầu truyện -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin đầu truyện -> giao diện tìm đầu truyện theo tên hiện ra -> QL nhập tên đầu truyện và click tìm kiếm -> danh sách các đầu truyện có tên chứa từ khóa hiện ra -> QL chọn sửa một đầu truyện -> giao diện sửa đầu truyện hiện ra với các thông tin của đầu truyện đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lý thông tin khách hàng được thực hiện tương tự.
- Modul "**Cho thuê truyện**" với mô tả chi tiết nghiệp vụ: Sau khi chọn được các truyện để thuê mượn, khách hàng (KH) cầm đến quầy nhân viên (NV) thu ngân làm phiếu mượn. NV nhập tên KH và tìm kiếm → Hệ thống trả về danh sách các KH có tên vừa nhập → NV click chọn tên KH trong danh sách (nếu KH mượn lần đầu thì nhập mới) → Hệ thống hiện giao diện thêm truyện mượn vào phiếu: Với mỗi đầu truyện, NV click chọn tìm truyện theo tên → nhập tên truyện + click tìm → hệ thống hiện lên danh sách các đầu truyện có tên vừa nhập → NV click chọn dòng đúng với quyền truyện do KH chọn thuê → Hệ thống thêm 1 ông tương ứng với đầu truyện đó vào phiếu thuê mượn như mô tả. Khi hết các đầu truyện do KH chọn thuê, NV click tạo phiếu mượn → Hệ thống lưu vào CSDL và hiển thị phiếu mượn lên màn hình → NV click in ra → Hệ thống in phiếu mượn ra cho KH.
- Modul "**Khách hàng trả truyện và thanh toán**" với mô tả chi tiết nghiệp vụ: Khi KH đem truyện đến trả, NV chọn menu tìm danh truyện mượn theo tên KH → nhập tên KH+click tìm kiếm → hệ thống hiển thị danh sách các KH có tên vừa nhập → NV chọn tên KH đúng với thông tin KH hiện tại → hệ thống hiện lên danh sách các đầu truyện mà KH đó đang mượn, mỗi đầu truyện trên một dòng với đầy đủ thông tin về đầu truyện, ngày mượn, giá mượn, và số tiền thuê tính đến ngày đang trả, cột cuối cùng là ô tích chọn trả → NV click vào nút chọn trả cho các đầu truyện mà KH đem trả (có thể không trả hết 1 lần), cuối cùng click nút thanh toán → hệ thống hiện hóa đơn đầy đủ thông tin khách hàng + 1 bảng danh sách các đầu truyện trả như mô tả trên + dòng cuối là tổng số tiền trả → NV click xác nhận → hệ thống cập nhật vào CSDL.

- Modul "**Thống kê truyện được mượn nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê đầu truyện được mượn nhiều → Hệ thống hiển thị danh sách các đầu truyện được mượn nhiều theo dạng bảng, mỗi dòng tương ứng với một đầu truyện với đầy đủ thông tin: mã, tên, tác giả, NXB, năm XB, cột tổng số lượt được mượn, cột tổng số tiền thu được. Xếp theo thứ tự giảm dần của cột tổng số lượt mượn, tiếp theo là giảm dần của cột tổng số tiền thu được.
- Modul "**Thống kê khách hàng mượn nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê khách hàng mượn nhiều → hệ thống hiển thị danh sách KH mượn nhiều theo dạng bảng, mỗi dòng tương ứng với một KH với đầy đủ thông tin: mã, tên, số CMT, số đt, địa chỉ, tiếp theo là cột tổng số lượt mượn, cột tổng số tiền đã trả. Xếp theo chiều giảm dần của tổng số lượt mượn, tiếp theo là chiều giảm dần của tổng số tiền trả.
- Modul "**Thống kê doanh thu**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê doanh thu theo thời gian (tháng, quý, năm) → hệ thống hiện ô chọn thống kê theo tháng, quý, hoặc năm → QL click chọn theo tháng → hệ thống hiện lên thống kê doanh thu theo tháng dưới dạng bảng, mỗi dòng tương ứng với 1 tháng (tương ứng là quý, năm): tên tháng, tổng doanh thu. Sắp xếp theo chiều thời gian tháng (tương ứng là quý, năm) gần nhất đến tháng (tương ứng là quý, năm) cũ nhất.

#### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đầu truyện
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Cho khách hàng mượn một số truyện
  - Khách hàng trả truyện và thanh toán
  - Xem thống kê các đầu truyện được mượn nhiều
  - Xem thống kê các khách hàng mượn nhiều
  - Xem thống kê doanh thu theo thời gian
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 đầu truyện
  - Thêm, sửa, xóa thông tin 1 khách hàng

- Cho khách hàng mượn một số truyện
  - Khách hàng trả truyện và thanh toán
  - Xem thống kê các đầu truyện được mượn nhiều
  - Xem thống kê các khách hàng mượn nhiều
  - Xem thống kê doanh thu theo thời gian
6. Viết tất cả các test case cho các chức năng:
- Thêm, sửa, xóa thông tin 1 đầu truyện
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Cho khách hàng mượn một số truyện
  - Khách hàng trả truyện và thanh toán
  - Xem thống kê các đầu truyện được mượn nhiều
  - Xem thống kê các khách hàng mượn nhiều
  - Xem thống kê doanh thu theo thời gian

### **11.13 HỆ THỐNG QUẢN LÍ CHO THUÊ SÂN BÓNG MINI CỦA CHỦ SÂN BÓNG**

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý cho thuê sân bóng mini của một chủ sân bóng với mô tả như sau:

- Sân bóng có nhiều sân con mini cho thuê. Tùy yêu cầu khách hàng mà có thể ghép 2 hay 4 sân bé liền nhau thành 1 sân lớn cho thuê.
- Mỗi sân có thể cho nhiều khách hàng thuê tại nhiều khung giờ khác nhau. Mỗi khách hàng có thể thuê nhiều sân khác nhau.
- Khách hàng có thể thuê sân theo buổi trong tuần hoặc thuê theo tháng (vào một hoặc một số buổi cố định trong tuần, trong vòng mấy tháng cụ thể).
- Khi làm hợp đồng thuê sân, khách hàng nhận được phiếu thuê sân. Trong đó, dòng đầu ghi ngày làm hợp đồng, thông tin chủ sân, thông tin của khách hàng. Các dòng tiếp theo, mỗi dòng ghi một sân mini với đầy đủ thông tin về sân, giá thuê một buổi, khung giờ thuê trong tuần, ngày bắt đầu, ngày kết thúc đợt thuê, tổng tiền thuê dự kiến. Dòng cuối cùng ghi tổng số tiền thuê sân dự kiến
- Khi đặt sân, khách hàng phải đặt cọc trước cho chủ sân ít nhất 10% tổng tiền thuê dự kiến. Và thông tin số tiền đặt cọc này cũng được ghi rõ trong phiếu đặt sân là đã thanh toán bao nhiêu tiền, vào ngày nào.

- Khi khách hàng đến đá bóng tại sân, chủ sân có thể phục vụ nước uống giải khát và đồ ăn nhẹ. Mỗi buổi khách hàng dùng các loại mặt hàng nào, mỗi loại bao nhiêu chai (gói), hết tổng tiền bao nhiêu đều được cập nhật vào hệ thống. Khách hàng sẽ thanh toán luôn khoản chi phí phát sinh này vào cuối đợt thuê sân.
- Khi thanh toán tiền thuê sân, khách hành nhận được một hóa đơn ghi chi tiết thông tin thuê sân và chi phí thuê sân giống như phiếu đặt sân. Có thể có thêm một số buổi phát sinh hoặc phải đổi lịch theo yêu cầu khách hàng. Ngoài ra, phần dưới hóa đơn ghi rõ đồ ăn uống phát sinh theo từng buổi, mỗi buổi được liệt kê thành một bảng, trong đó mỗi dòng của bảng mô tả một mặt hàng: mã, tên, giá, số lượng dùng, thành tiền. Tổng số tiền từng buổi và tổng số tiền cho cả đợt đặt sân.

#### **Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin sân bóng**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin sân bóng với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý sân bóng -> trang quản lý hiện ra -> QL chọn chức năng sửa thông tin sân bóng -> giao diện tìm sân bóng theo tên hiện ra -> QL nhập tên sân bóng và click tìm kiếm -> danh sách các sân bóng có tên chứa từ khóa hiện ra -> QL chọn sửa một sân bóng -> giao diện sửa sân bóng hiện ra với các thông tin của sân bóng đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lý thông tin khách hàng được thực hiện tương tự.
- Modul "**Đặt sân**" với mô tả chi tiết nghiệp vụ: Khách hàng (KH) đến yêu cầu đặt sân → Nhân viên (NV) chọn chức năng đặt sân → hệ thống hiện giao diện tìm sân trống theo khung giờ → NV nhập khung giờ + chọn loại sân theo yêu cầu KH + click tìm → hệ thống hiện lên danh sách sân còn trống theo khung giờ đã chọn → NV click chọn 1 sân → hệ thống hiện giao diện điền thông tin KH → NV nhập tên và tìm → hệ thống hiện lên danh sách các KH có tên vừa nhập → NV click chọn tên KH đúng với KH hiện tại (nếu KH lần đầu đến đặt sân thì phải thêm mới) → hệ thống hiện giao diện nhập khoảng thời gian ngày bắt đầu, ngày kết thúc đợt đặt sân (ưu tiên đặt theo quý) → NV click chọn và click xác nhận → hệ thống hiện phiếu đặt sân với đầy đủ thông tin KH, thông tin sân đặt, giá sân đặt, khung giờ đặt, tổng số buổi theo thời gian đã chọn, tổng số tiền ước tính và số tiền phải đặt cọc (10%) → NV click xác nhận → hệ thống in phiếu đặt sân và cập nhật vào CSDL.
- Modul "**Khách hàng thanh toán**" với mô tả chi tiết nghiệp vụ: Khi KH đến yêu cầu thanh toán, NV chọn menu tìm phiếu đặt sân theo tên KH → nhập tên KH+click tìm kiếm → hệ thống hiển thị danh sách các KH có tên vừa nhập → NV chọn tên KH đúng với thông tin

KH hiện tại → hệ thống hiện lên danh sách các phiếu đặt mà KH đó đang đặt → NV click vào nút chọn thanh toán cho 1 phiếu đặt sân → hệ thống hiện hóa đơn đầy đủ thông tin khách hàng + 1 bảng danh sách các sản phẩm ăn uống mà KH đã sử dụng trong suốt các buổi thuê sân như mô tả trên + dòng cuối là tổng số tiền trả → NV click xác nhận → hệ thống cập nhật vào CSDL.

- Modul "**Thống kê khung giờ được thuê nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê khung giờ được thuê nhiều → Hệ thống hiển thị danh sách các khung giờ được thuê nhiều theo dạng bảng, mỗi dòng tương ứng với một khung giờ với đầy đủ thông tin: khung giờ, ngày, cột tổng số lượt được thuê, cột tổng số tiền thu được. Xếp theo thứ tự giảm dần của cột tổng số lượt thuê, tiếp theo là giảm dần của cột tổng số tiền thu được.
- Modul "**Thống kê khách hàng**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê khách hàng thuê sân nhiều → hệ thống hiển thị danh sách KH thuê sân nhiều theo dạng bảng, mỗi dòng tương ứng với một KH với đầy đủ thông tin: mã, tên, số CMND, số đt, địa chỉ, tiếp theo là cột tổng số lượt thuê, cột tổng số tiền đã trả. Xếp theo chiều giảm dần của tổng số lượt thuê, tiếp theo là chiều giảm dần của tổng số tiền trả.
- Modul "**Thống kê doanh thu**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê doanh thu theo thời gian (tháng, quý, năm) → hệ thống hiện ô chọn thống kê theo tháng, quý, hoặc năm → QL click chọn theo tháng → hệ thống hiện lên thống kê doanh thu theo tháng dưới dạng bảng, mỗi dòng tương ứng với 1 tháng (tương ứng là quý, năm): tên tháng, tổng doanh thu. Sắp xếp theo chiều thời gian tháng (tương ứng là quý, năm) gần nhất đến tháng (tương ứng là quý, năm) cũ nhất.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sân bóng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Cho khách hàng đặt sân
  - Khách hàng thanh toán
  - Xem thống kê các khung giờ được thuê nhiều
  - Xem thống kê các khách hàng thuê nhiều
  - Xem thống kê doanh thu theo thời gian

4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sân bóng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Cho khách hàng đặt sân
  - Khách hàng thanh toán
  - Xem thống kê các khung giờ được thuê nhiều
  - Xem thống kê các khách hàng thuê nhiều
  - Xem thống kê doanh thu theo thời gian
6. Viết tắt cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 sân bóng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Cho khách hàng đặt sân
  - Khách hàng thanh toán
  - Xem thống kê các khung giờ được thuê nhiều
  - Xem thống kê các khách hàng thuê nhiều
  - Xem thống kê doanh thu theo thời gian

## 11.14 HỆ THỐNG QUẢN LÝ CHO THUÊ PHÒNG TRỌ CỦA CHỦ NHÀ TRỌ

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý cho thuê phòng trọ của một chủ nhà trọ với mô tả như sau:

- Chủ nhà trọ có nhiều khu nhà trọ. Mỗi khu có nhiều phòng trọ. Mỗi phòng có mức giá thuê khác nhau do chủ trọ ấn định.
- Một khách hàng có thể thuê nhiều phòng trọ ở nhiều thời điểm khác nhau. Một phòng trọ có thể được thuê bởi nhiều khách hàng ở nhiều thời điểm khác nhau.
- Mỗi khách hàng, khi thuê một phòng trọ, hàng tháng phải trả với một mức giá xác định.
- Khi làm hợp đồng, khách hàng phải đặt cọc tiền thuê nhà đúng bằng tiền thuê 1 tháng tại thời điểm làm hợp đồng.
- Hàng tháng, khi thoanh toán tiền phòng, khách hàng phải thanh toán đồng thời tiền các dịch vụ đi kèm: điện (tính theo công tơ điện, chót theo tháng), nước (tính theo công tơ nước, chốt theo tháng), tiền vệ sinh (tính theo đầu người), tiền gửi xe máy xe đạp (tính theo đầu xe).

- Hàng tháng, chủ trọ sẽ đi chót số điện nước của từng phòng vào ngày cuối tháng và về lập hóa đơn tiền nhà cho từng phòng để đưa đến cho từng phòng.
- Các khách hàng phải thanh toán cho chủ trọ trong thời gian chậm nhất 1 tuần từ ngày nhận hóa đơn.
- Khi thanh toán, chủ trọ cập nhật lại trạng thái hóa đơn đã thanh toán, và kí xác nhận thanh toán vào hóa đơn đã đưa cho khách hàng trước đó.
- Nếu quá hạn mà chưa đóng tiền hàng tháng, khách hàng phải chịu lãi trên tổng số tiền nợ theo tỉ lệ % (theo tháng).

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin phòng**" cho phép quản lý (QL) thực hiện thêm, sửa, xóa thông tin phòng với mô tả chi tiết nghiệp vụ: QL chọn menu quản lý phòng → trang quản lý hiện ra → QL chọn chức năng sửa thông tin phòng → giao diện tìm phòng theo tên hiện ra → QL nhập tên phòng và click tìm kiếm → danh sách các phòng có tên chứa từ khóa hiện ra → QL chọn sửa một phòng → giao diện sửa phòng hiện ra với các thông tin của phòng đã chọn → QL nhập một số thông tin thay đổi và click cập nhật → hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lý thông tin khách hàng, quản lý thông tin khu trọ được thực hiện tương tự.
- Modul "**Làm hợp đồng cho thuê phòng**" với mô tả chi tiết nghiệp vụ: Khi có khách đến thuê nhà, quản lý chọn chức năng tìm phòng còn trống → hệ thống hiện giao diện tìm phòng trống → QL nhập ngày muốn bắt đầu ở và mức giá chấp nhận của KH → hệ thống hiện lên danh sách các phòng còn trống theo yêu cầu khách hàng → QL click chọn 1 phòng theo KH chọn → hệ thống hiện giao diện nhập thông tin KH: tên, ngày tháng năm sinh, số CMT, điện thoại, quê quán → hệ thống hiện lên mẫu hợp đồng thuê nhà với đầy đủ thông tin bên A (chủ nhà), bên B (Khách hàng), thông nhất ở phòng nào, tần ngày nào, giá thuê khởi điểm là bao nhiêu, tiền đặt cọc, tiền điện, nước, vệ sinh, xe máy, xe đạp hàng tháng bao nhiêu → KH duyệt và chấp nhận thì QL click xác nhận → hệ thống in hợp đồng và lưu vào hệ thống.
- Modul "**Lên hóa đơn tháng cho khách hàng**" với mô tả chi tiết nghiệp vụ: QL chọn chức năng lên hóa đơn hàng tháng → hệ thống hiện danh sách các phòng chưa lên hóa đơn tháng này → QL click chọn 1 phòng → hệ thống hiện giao diện nhập số điện, số nước ghi được của tháng này → QL nhập → hệ thống hiển thị hóa đơn tháng của phòng đã chọn với đầy đủ thông tin: thông tin đầy đủ của phòng, thông tin đầy đủ của khách hàng, các khoản thu được liệt kê mỗi khoản trên một dòng: tiền phòng, tiền điện, tiền nước, tiền vệ

sinh, xe máy, xe đạp. Mỗi dòng có số lượng, đơn giá, thành tiền. 2 dòng tiếp theo ghi số nợ tồn dư đến tháng trước (nếu không có thì ghi 0), và tổng tiền tháng này. Dòng cuối cùng ghi tổng số tiền mà khách hàng phải trả → QL click xác nhận → hệ thống in ra hóa đơn cho phòng tương ứng. Quá trình này được lặp lại cho đến khi hết các phòng của chủ nhà.

- Modul "**Thanh toán hàng tháng cho khách hàng**" với mô tả chi tiết nghiệp vụ: Khi KH cầm hóa đơn đến thanh toán, QL chọn chức năng thanh toán → hệ thống hiện giao diện nhập mã hóa đơn (hoặc tên KH, tên phòng) → QL nhập và click tìm → hệ thống hiện thông tin hóa đơn tương ứng → QL nhận tiền và click nút xác nhận đã thanh toán → hệ thống thông báo cập nhật thành công và QL kí xác nhận đã thanh toán vào hóa đơn cho KH.
- Modul "**Thông kê khách hàng còn nợ nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê khách hàng còn nợ nhiều → hệ thống hiển thị danh sách KH còn nợ nhiều theo dạng bảng, mỗi dòng tương ứng với một KH với đầy đủ thông tin: mã, tên, số CMT, số đt, thông tin phòng đang thuê, tiếp theo là cột tổng số tiền phòng gốc còn nợ, cột tổng số tiền lãi phát sinh, và cột tổng số tiền hiện phải trả. Xếp theo chiều giảm dần của tổng số tổng số tiền hiện phải trả.
- Modul "**Thông kê doanh thu**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê doanh thu theo thời gian (tháng, quý, năm) → hệ thống hiện ô chọn thống kê theo tháng, quý, hoặc năm → QL click chọn theo tháng → hệ thống hiện lên thống kê doanh thu theo tháng dưới dạng bảng, mỗi dòng tương ứng với 1 tháng (tương ứng là quý, năm): tên tháng, tổng doanh thu. Sắp xếp theo chiều thời gian tháng (tương ứng là quý, năm) gần nhất đến tháng (tương ứng là quý, năm) cũ nhất.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.
3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phòng trọ
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Làm hợp đồng cho khách hàng thuê
  - Lênh hóa đơn hàng tháng cho khách hàng
  - Nhận thanh toán hàng tháng từ khách hàng
  - Xem thống kê các khách hàng còn nợ nhiều

- Xem thống kê doanh thu theo thời gian
4. Thiết kế CSDL cho toàn hệ thống
  5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
    - Thêm, sửa, xóa thông tin 1 phòng trọ
    - Thêm, sửa, xóa thông tin 1 khách hàng
    - Làm hợp đồng cho khách hàng thuê
    - Lên hóa đơn hàng tháng cho khách hàng
    - Nhận thanh toán hàng tháng từ khách hàng
    - Xem thống kê các khách hàng còn nợ nhiều
    - Xem thống kê doanh thu theo thời gian
  6. Viết tất cả các test case cho các chức năng:
    - Thêm, sửa, xóa thông tin 1 phòng trọ
    - Thêm, sửa, xóa thông tin 1 khách hàng
    - Làm hợp đồng cho khách hàng thuê
    - Lên hóa đơn hàng tháng cho khách hàng
    - Nhận thanh toán hàng tháng từ khách hàng
    - Xem thống kê các khách hàng còn nợ nhiều
    - Xem thống kê doanh thu theo thời gian

## 11.15 HỆ THỐNG QUẢN LÝ CHO THUÊ GIƯỜNG KÍ TÚC XÁ

Khách hàng yêu cầu anh/chị phát triển một phần mềm quản lý cho thuê giường trong kí túc xá với mô tả như sau:

- Tổ hợp có nhiều phòng. Mỗi phòng thuộc một hạng phòng khác nhau và có giá thuê giường trong phòng đó khác nhau: phòng điều hòa/không điều hòa, phòng có 4 giường/8 giường.
- Mỗi phòng có nhiều giường. Mỗi giường thuộc một kiểu giường khác nhau và có giá thuê khác nhau: giường tầng trên/tầng dưới trong phòng có 4 giường/8 giường có hay không có điều hòa (tổng cộng có 8 loại giường)
- Một sinh viên có thể thuê nhiều giường ở nhiều thời điểm khác nhau. Một giường có thể được thuê bởi nhiều sinh viên ở nhiều thời điểm khác nhau.
- Mỗi sinh viên, khi thuê một phòng trọ, hàng tháng phải trả với một mức giá xác định.

- Khi làm hợp đồng, sinh viên phải đặt cọc tiền thuê nhà đúng bằng tiền thuê 1 tháng tại thời điểm làm hợp đồng.
- Hàng tháng, khi thoanh toán tiền phòng, sinh viên phải thanh toán đồng thời tiền các dịch vụ đi kèm: điện (tính theo công tơ điện, chốt theo tháng, chia trung bình theo phòng), nước (tính theo công tơ nước, chốt theo tháng, chia trung bình theo phòng), tiền vệ sinh (tính theo đầu người), tiền gửi xe máy xe đạp (tính theo đầu xe).
- Hàng tháng, nhân viên sẽ đi chốt số điện nước của từng phòng vào ngày cuối tháng và về lập hóa đơn cho từng sinh viên (giường) để đưa đến cho từng phòng.
- Sinh viên phải thanh toán trong thời gian chậm nhất 1 tuần từ ngày nhận hóa đơn.
- Khi thanh toán, nhân viên cập nhật lại trạng thái hóa đơn đã thanh toán, và kí xác nhận thanh toán vào hóa đơn đã đưa cho sinh viên trước đó.

**Các modul chức năng hoạt động như sau:**

- Modul "**Quản lý thông tin phòng**" cho phép quản lí (QL) thực hiện thêm, sửa, xóa thông tin phòng với mô tả chi tiết nghiệp vụ: QL chọn menu quản lí phòng -> trang quản lí hiện ra -> QL chọn chức năng sửa thông tin phòng -> giao diện tìm phòng theo tên hiện ra -> QL nhập tên phòng và click tìm kiếm -> danh sách các phòng có tên chứa từ khóa hiện ra -> QL chọn sửa một phòng -> giao diện sửa phòng hiện ra với các thông tin của phòng đã chọn -> QL nhập một số thông tin thay đổi và click cập nhật -> hệ thống lưu thông tin vào CSDL và thông báo thành công.
- Các modul quản lí thông tin giường của một phòng, quản lí thông tin khách hàng được thực hiện tương tự.
- Modul "**Làm hợp đồng cho thuê với khách hàng**" với mô tả chi tiết nghiệp vụ: Khi có khách đến thuê nhà, quản lí chọn chức năng tìm phòng còn trống → hệ thống hiện giao diện tìm phòng trống → QL nhập ngày muốn bắt đầu ở và mức giá chấp nhận của KH → hệ thống hiện lên danh sách các phòng còn trống theo yêu cầu khách hàng → QL click chọn 1 phòng theo KH chọn → hệ thống hiện giao diện nhập thông tin KH: tên, ngày tháng năm sinh, số CMT, điện thoại, quê quán → hệ thống hiện lên mẫu hợp đồng thuê nhà với đầy đủ thông tin bên A (chủ nhà), bên B (Khách hàng), thông nhất ở phòng nào, từ ngày nào, giá thuê khởi điểm là bao nhiêu, tiền đặt cọc, tiền điện, nước, vệ sinh, xe máy, xe đạp hàng tháng bao nhiêu → KH duyệt và chấp nhận thì QL click xác nhận → hệ thống in hợp đồng và lưu vào hệ thống.
- Modul "**Lên hóa đơn hàng tháng cho khách hàng**" với mô tả chi tiết nghiệp vụ: QL chọn chức năng lên hóa đơn hàng tháng → hệ thống hiện danh sách các giường chưa lên hóa đơn tháng này → QL click chọn 1 giường → hệ thống hiển thị hóa đơn tháng của

giường đã chọn với đầy đủ thông tin: thông tin đầy đủ của giường (+ phòng), thông tin đầy đủ của sinh viên, các khoản thu được liệt kê mỗi khoản trên một dòng: tiền giường, tiền điện, tiền nước, tiền vệ sinh, xe máy, xe đạp. Mỗi dòng có số lượng, đơn giá, thành tiền. 2 dòng tiếp theo ghi số nợ tồn dư đến tháng trước (nếu không có thì ghi 0), và tổng tiền tháng này. Dòng cuối cùng ghi tổng số tiền mà giường phải trả → QL click xác nhận → hệ thống in ra hóa đơn cho giường tương ứng. Quá trình này được lặp lại cho đến khi hết các giường của các phòng trong khu KTX.

- Modul "**Thanh toán hàng tháng cho sinh viên**" với mô tả chi tiết nghiệp vụ: Khi SV cầm hóa đơn đến thanh toán, QL chọn chức năng thanh toán → hệ thống hiện giao diện nhập mã hóa đơn (hoặc tên SV, tên phòng-giường) → QL nhập và click tìm → hệ thống hiện thông tin hóa đơn tương ứng → QL nhận tiền và click nút xác nhận đã thanh toán → hệ thống thông báo cập nhật thành công và QL kí xác nhận đã thanh toán vào hóa đơn cho SV.
- Modul "**Thông kê các loại giường được thuê nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê loại giường được thuê nhiều → Hệ thống hiển thị danh sách các loại được thuê nhiều theo dạng bảng, mỗi dòng tương ứng với một loại giường với đầy đủ thông tin: kiểu phòng có/không có điều hòa, phòng 4 giường/8 giường, giường tầng 1 hay 2, cột tổng số lượt được thuê, cột tổng số tiền thu được. Xếp theo thứ tự giảm dần của cột tổng số lượt thuê, tiếp theo là giảm dần của cột tổng số tiền thu được.
- Modul "**Thông kê sinh viên còn nợ nhiều**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê sinh viên còn nợ nhiều → hệ thống hiển thị danh sách SV thuê sân nhiều theo dạng bảng, mỗi dòng tương ứng với một SV với đầy đủ thông tin: mã, tên, số CMT, số đt, trường, khoa, khóa, tên phòng, kiểu phòng, mã giường, loại giường, tiếp theo là cột tổng số tiền còn nợ chưa trả. Xếp theo chiều giảm dần của tổng số tiền còn nợ chưa trả.
- Modul "**Thông kê doanh thu**" với mô tả chi tiết nghiệp vụ: QL chọn menu thống kê doanh thu theo thời gian (tháng, quý, năm) → hệ thống hiện ô chọn thống kê theo tháng, quý, hoặc năm → QL click chọn theo tháng → hệ thống hiện lên thống kê doanh thu theo tháng dưới dạng bảng, mỗi dòng tương ứng với 1 tháng (tương ứng là quý, năm): tên tháng, tổng doanh thu. Sắp xếp theo chiều thời gian tháng (tương ứng là quý, năm) gần nhất đến tháng (tương ứng là quý, năm) cũ nhất.

### **Yêu cầu:**

1. Vẽ sơ đồ use case cho toàn bộ hệ thống.
2. Trích lớp thực thể và vẽ sơ đồ lớp thực thể cho toàn bộ hệ thống.

3. Viết scenario (chuẩn và ngoại lệ) + trích các lớp theo mô hình MVC + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phòng
  - Thêm, sửa, xóa thông tin 1 giường trong phòng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Làm hợp đồng cho khách hàng thuê
  - Lên hóa đơn hàng tháng cho khách hàng
  - Nhận thanh toán hàng tháng từ khách hàng
  - Xem thống kê các loại giường được thuê nhiều
  - Xem thống kê các khách hàng còn nợ nhiều
  - Xem thống kê doanh thu theo thời gian
4. Thiết kế CSDL cho toàn hệ thống
5. Thiết kế sơ đồ lớp chi tiết + thiết kế sơ đồ hoạt động + vẽ sơ đồ tuần tự cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phòng
  - Thêm, sửa, xóa thông tin 1 giường trong phòng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Làm hợp đồng cho khách hàng thuê
  - Lên hóa đơn hàng tháng cho khách hàng
  - Nhận thanh toán hàng tháng từ khách hàng
  - Xem thống kê các loại giường được thuê nhiều
  - Xem thống kê các khách hàng còn nợ nhiều
  - Xem thống kê doanh thu theo thời gian
6. Viết tất cả các test case cho các chức năng:
  - Thêm, sửa, xóa thông tin 1 phòng
  - Thêm, sửa, xóa thông tin 1 giường trong phòng
  - Thêm, sửa, xóa thông tin 1 khách hàng
  - Làm hợp đồng cho khách hàng thuê
  - Lên hóa đơn hàng tháng cho khách hàng
  - Nhận thanh toán hàng tháng từ khách hàng
  - Xem thống kê các loại giường được thuê nhiều
  - Xem thống kê các khách hàng còn nợ nhiều
  - Xem thống kê doanh thu theo thời gian