Hotel Management System

→ The hotel management system is a software used to manage all hotel activities efficiently & smoothly.

→ This help to track of room, customer & worker through a single portal.

→ System, search & book room

→ The guest is charged on duration the hotel room is booked.

Expectations from the interviewee

→ How will the system ensure that multiple users do not book the same room?
→ What type of users are allowed to book a room in the hotel?
→ Can users book a room in advance?
→ What payment methods can the customer use? (For eg → credit card / or cash)
→ How is the payment performed? Does the customer pay online or through a reception at the hotel?
→ Will the customer be able to pay in advance for room booking or is a just-in-time (JIT) payment system available?
→ How will the booking price be calculated? Which factors are involved
→ How does location & size depends on pricing?
→ How does the booking duration affect payment
→ Can user cancel the booking
→ Which type of users are allowed to cancel booking.

→ Design Pattern used → → Strategy design pattern
→ Singleton design pattern

Requirement Collection

R1: There can be four types of accounts in the system such as housekeeper, receptionist, guest or server.

R2: The rooms can be of different styles like standard, delux, family suits or bussiness suits.

R3: The system should allow the guest to search for any room & book any of the available room.

R4: During room booking, the user will enter the check-in date & duration to the stay. The user would also have to give some advance.

R5: The customer can cancel the booking & full-refund will be provided if the booking is canceled before 24 hours of the Check-in time.

R6: The system should send a notification to the customer's about the booking status or other information

R7: All Housekeeping tasks should be logged in & managed by the system.

R8: The system should allow the customer to add service of their own choice like food, or Kitchen service or amenity.

R9: Every room should have its own specific key, & there can be a master key that opens a specific set of rooms.

R10: A Hotel can have multiple branches of it.

Primary Actors

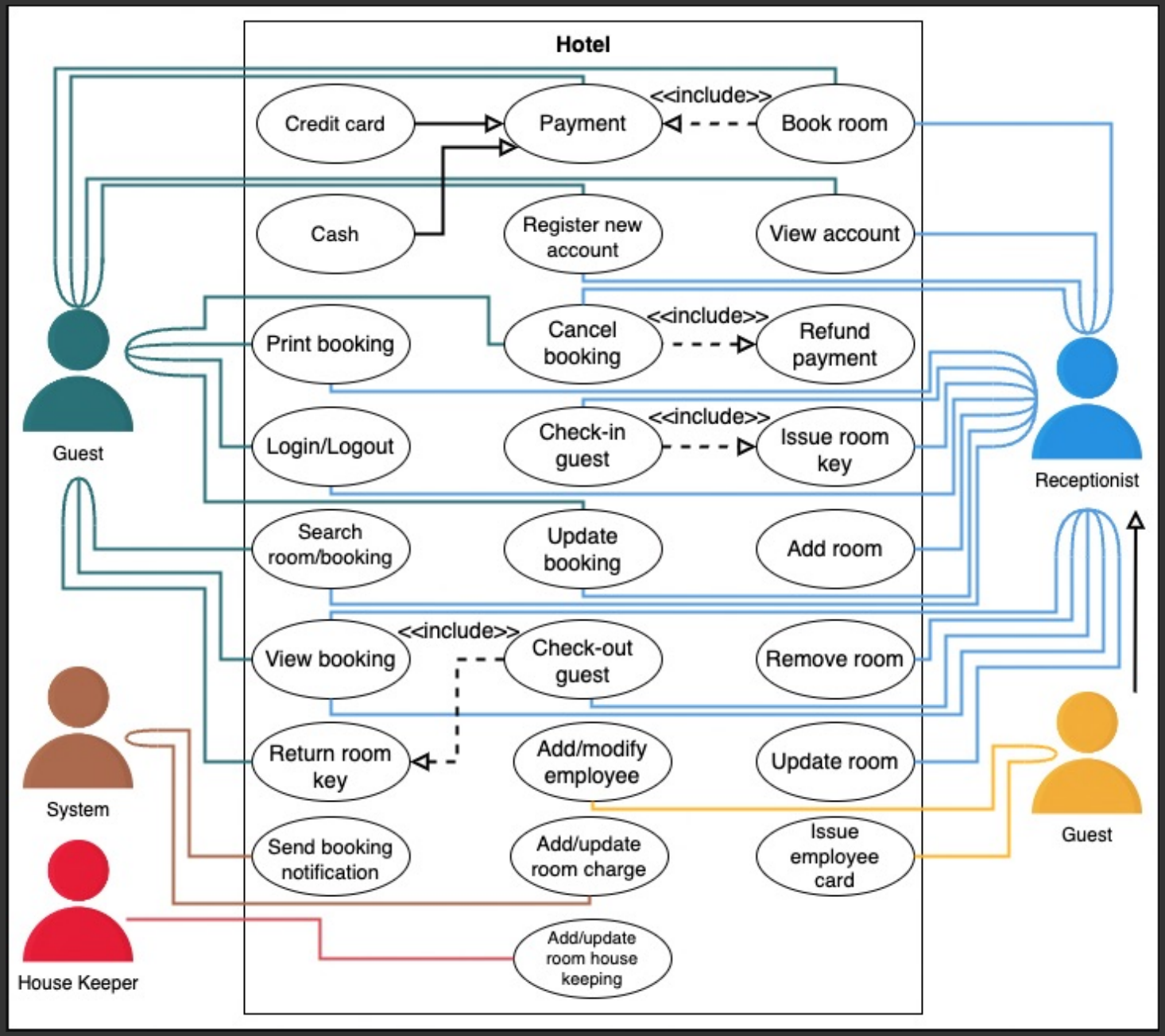
- Guest
- Receptionist
- Manager
- Housekeeping

Secondary Actors

- System
- Server.

Guest	Receptionist	Manager	System	Housekeeper
Book room	Book room	Issue employee card	Send booking notification	Add/update room housekeeping
Payment	View account	Add/modify employee	Add/update room charge	
View account	Register new account	Book room		
Register new account	Print booking	View account		
Print booking	Cancel booking	Register new account		
Cancel booking	Login/Logout	Print booking		
Login/Logout	Check in guest	Cancel booking		

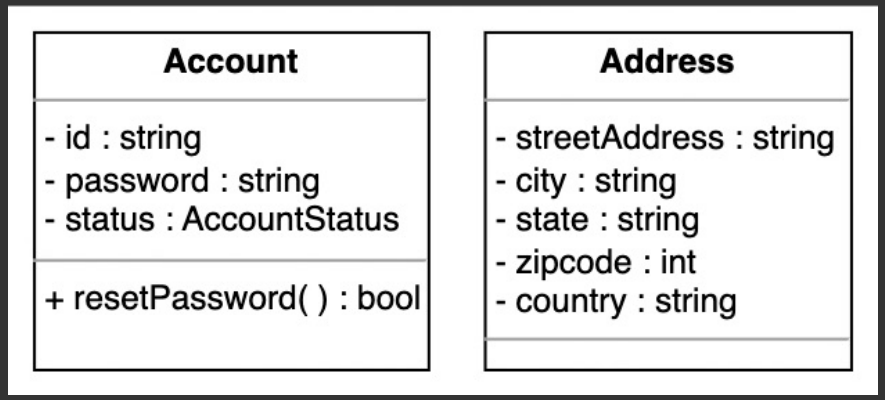
Guest	Receptionist	Manager	System	Housekeeper
Search room/booking	issue room key	Login/Logout		
Update booking	Search room/booking	Check-in guest		
View booking	Update booking	Issue room key		
Return room key	Check out guest	Search room/booking		
	View booking	Update booking		
	Add room	Check-out guest		
	Remove room	View booking		
	Update room	Add room		
		Remove room		
		Update room		



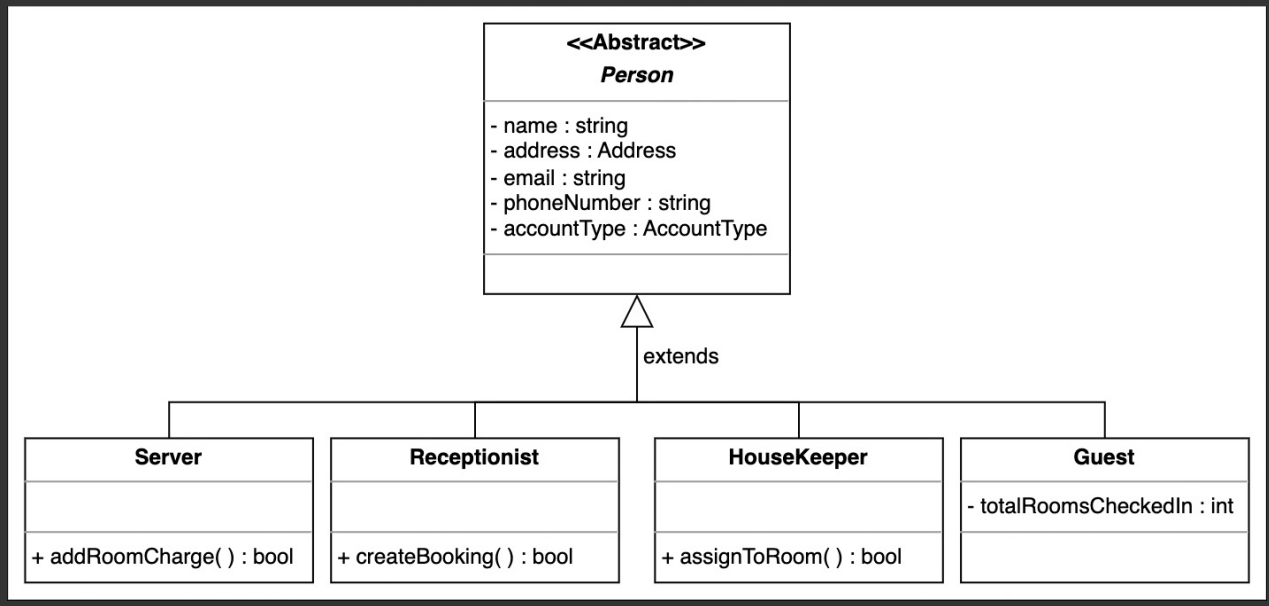
Use Case Diagram

Class Diagram of Hotel Management System

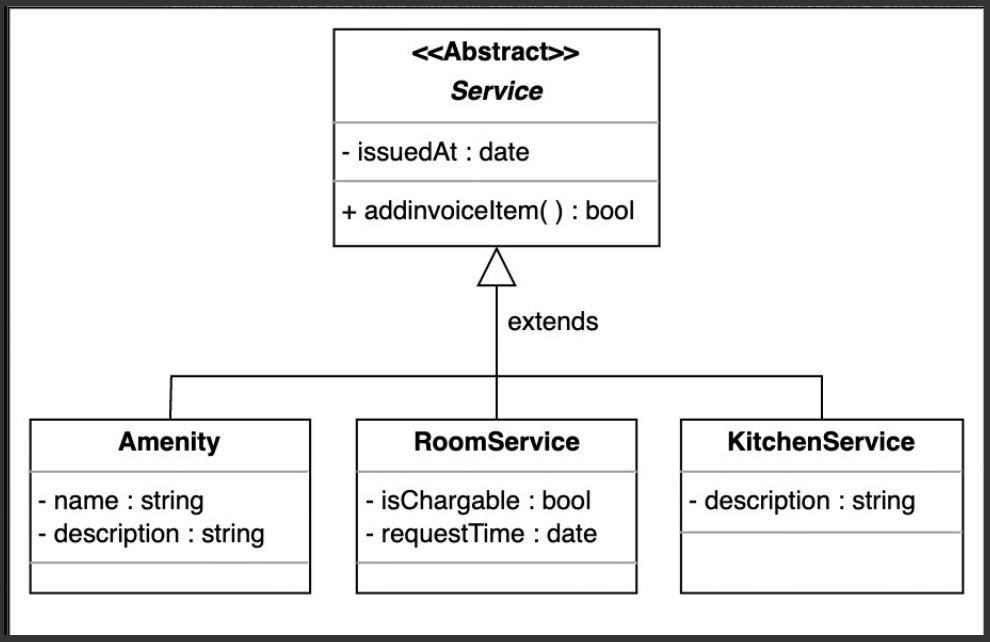
1. Address & Account



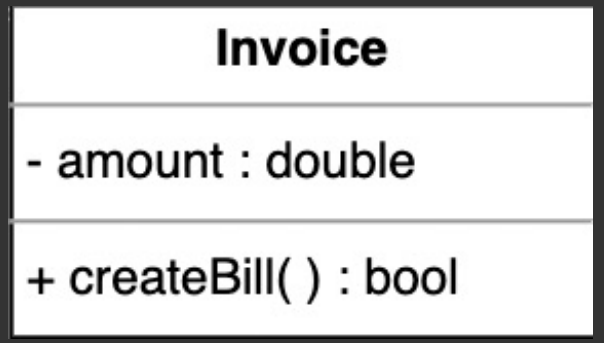
2. Person (A abstract class)



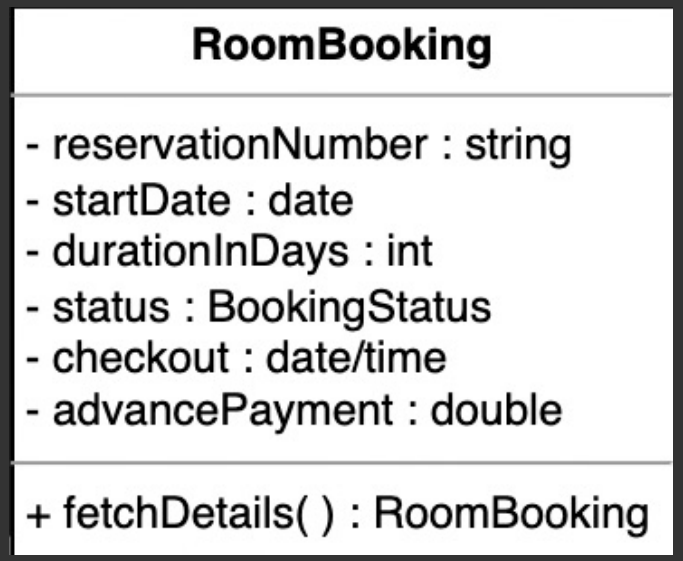
3. Service



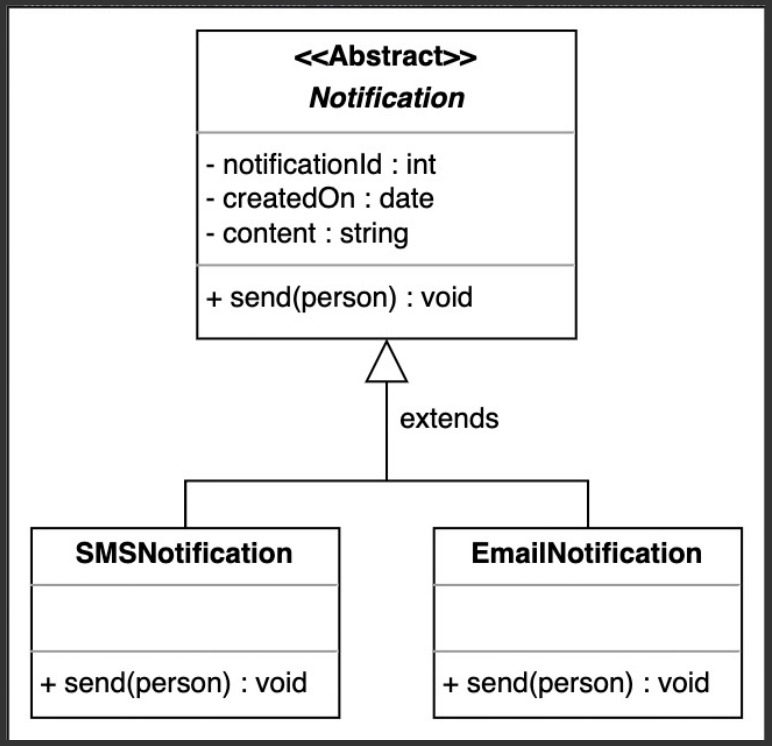
4. Invoice



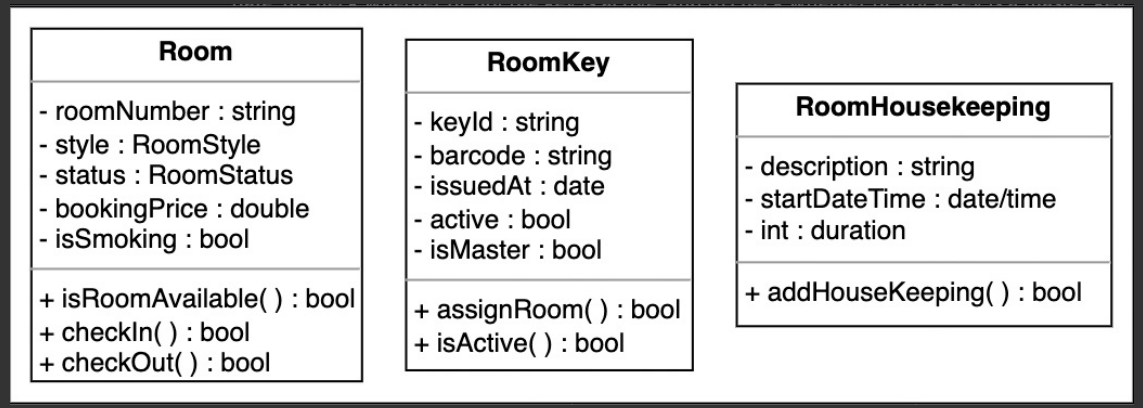
5. Room Booking



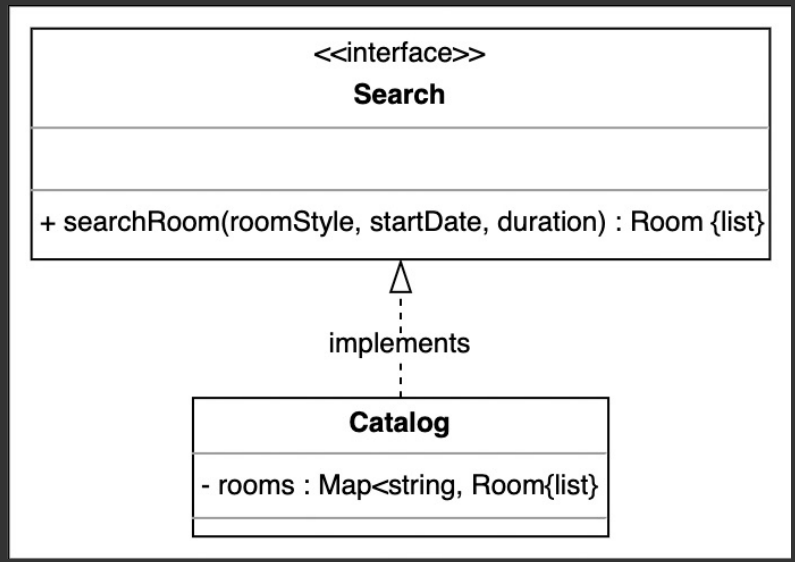
6. Notification



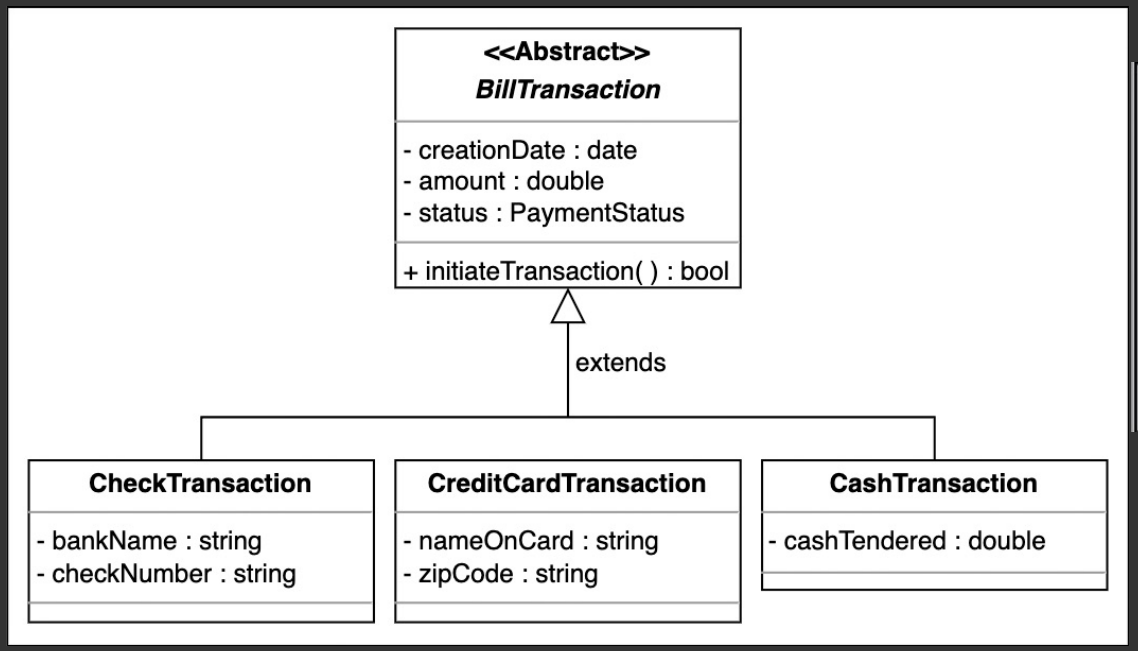
7. Room, RoomKey & Room House Keeping.



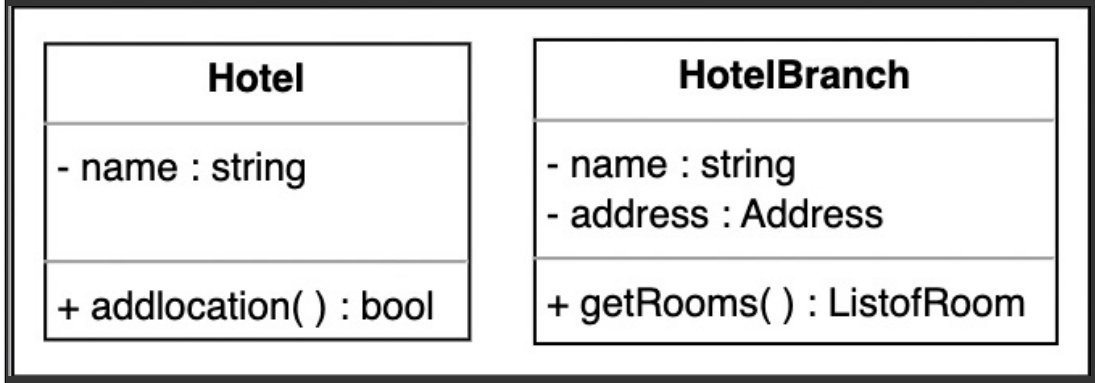
8. Search Interface & Catlog

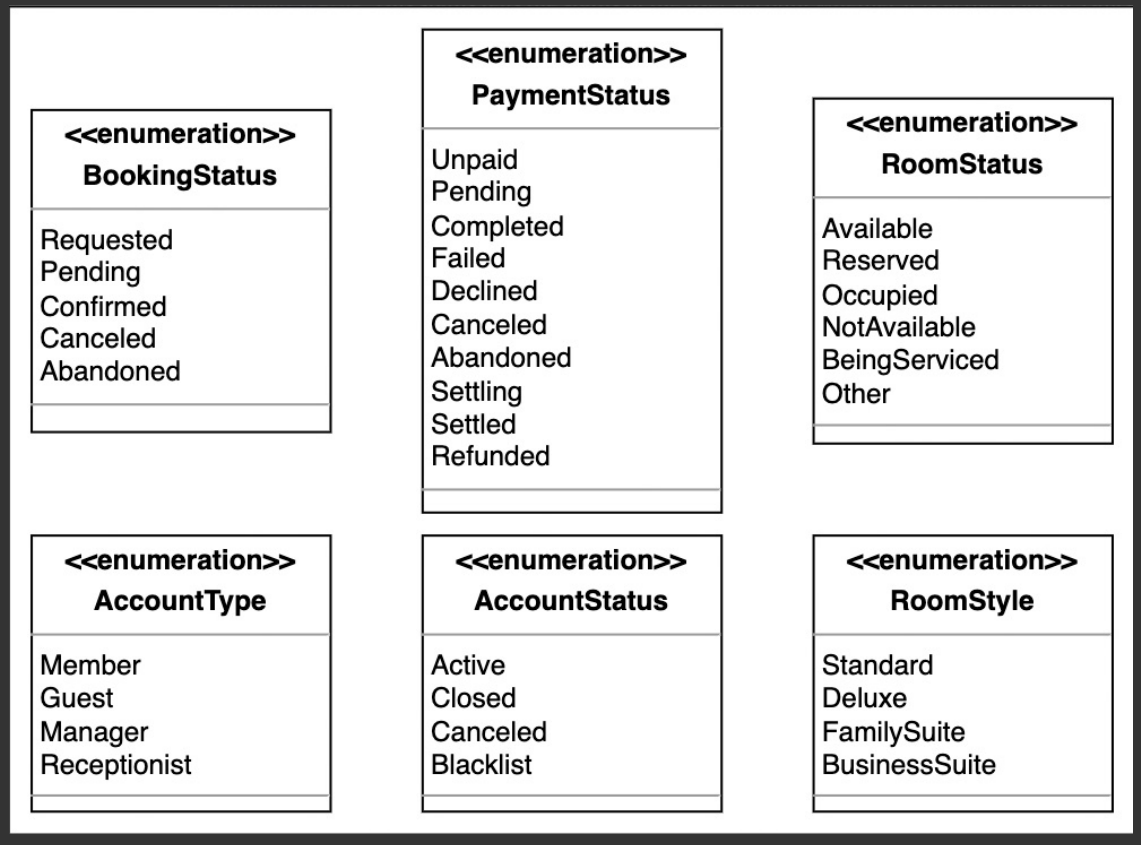


9. Bill Transaction



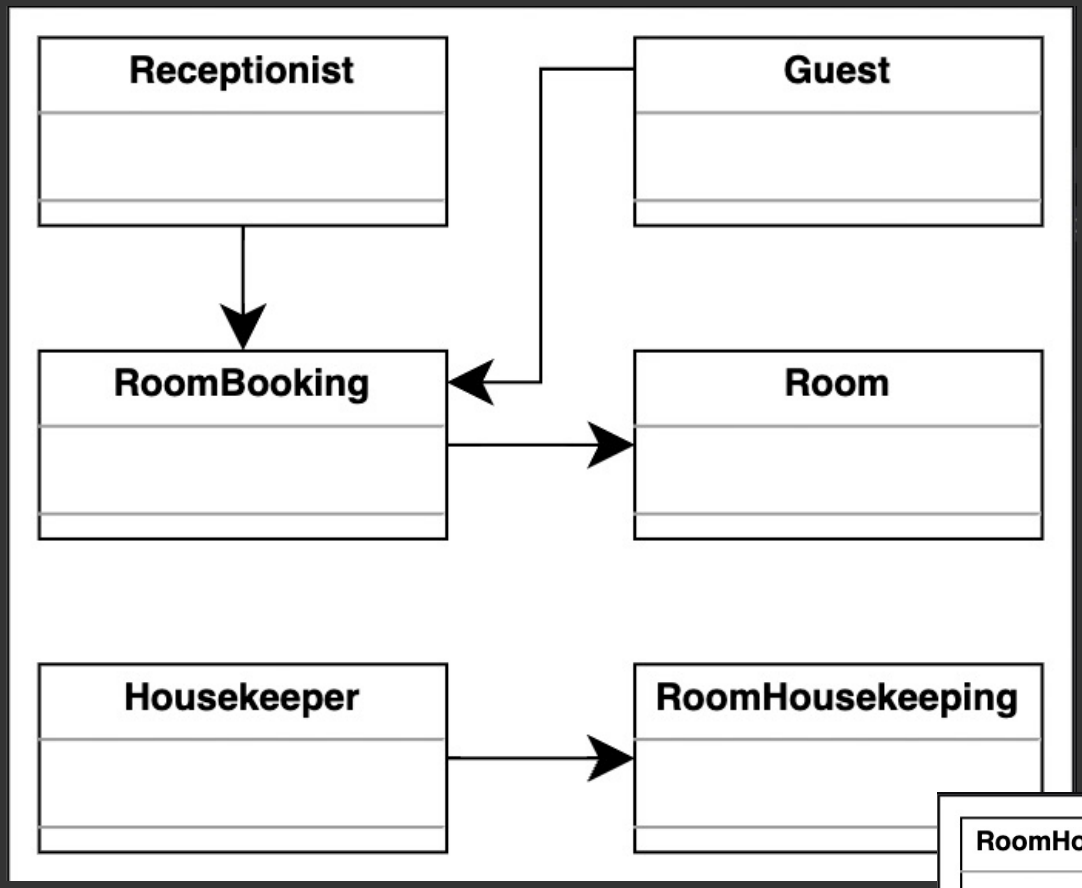
10. Hotel & Hotel branch



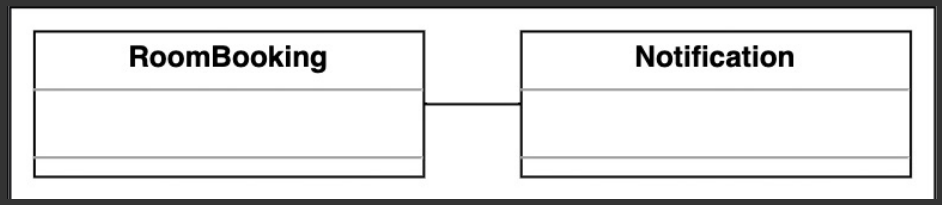


Enumerations

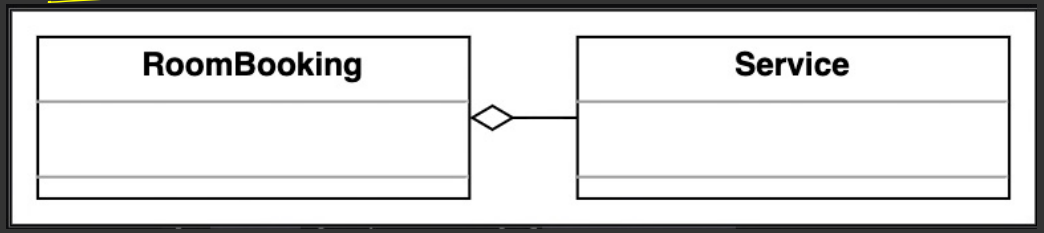
Association



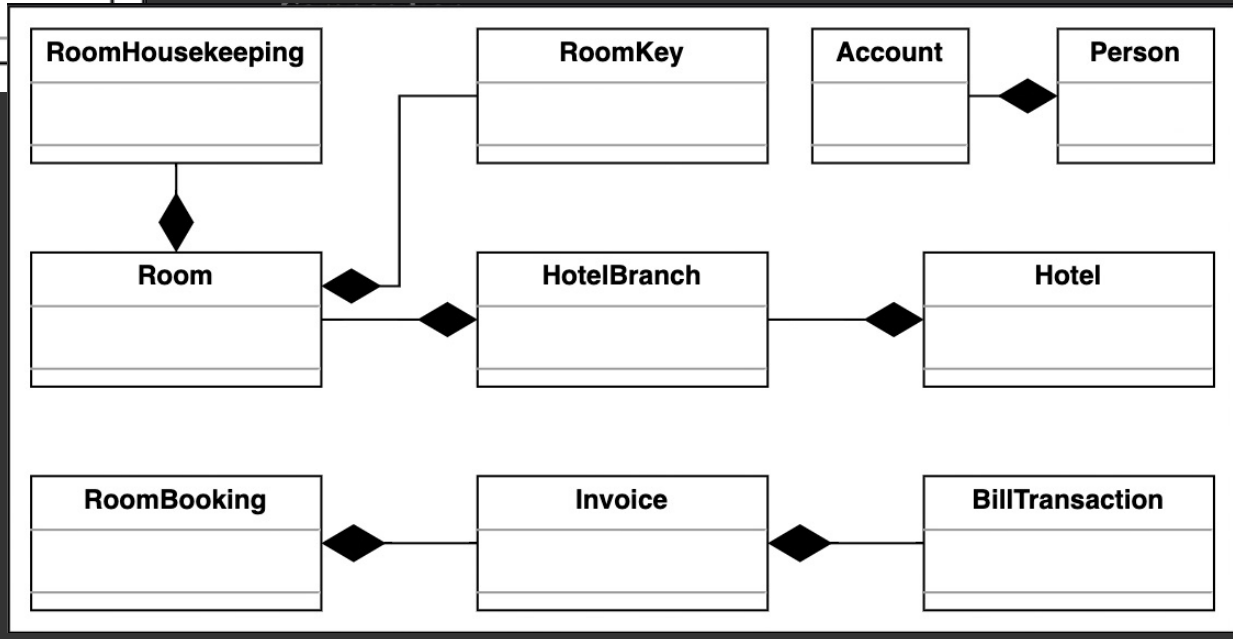
Two State Association

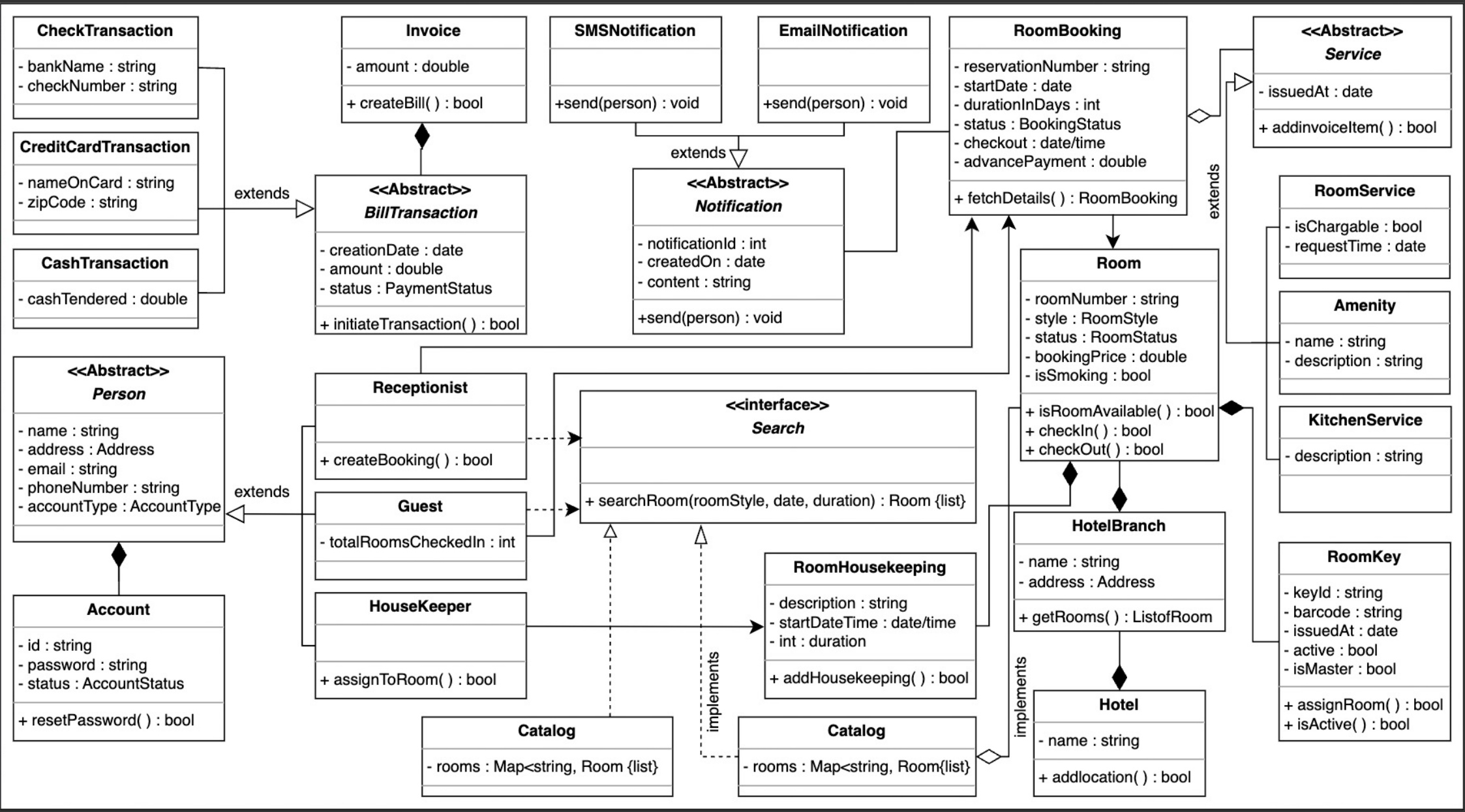


Aggregation

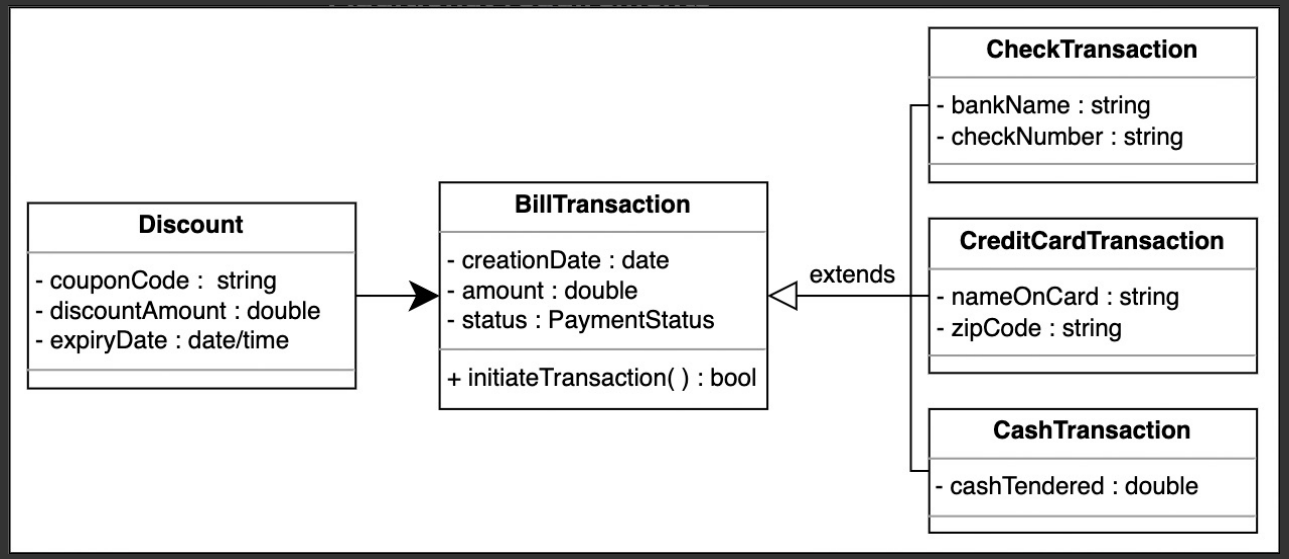


Composition

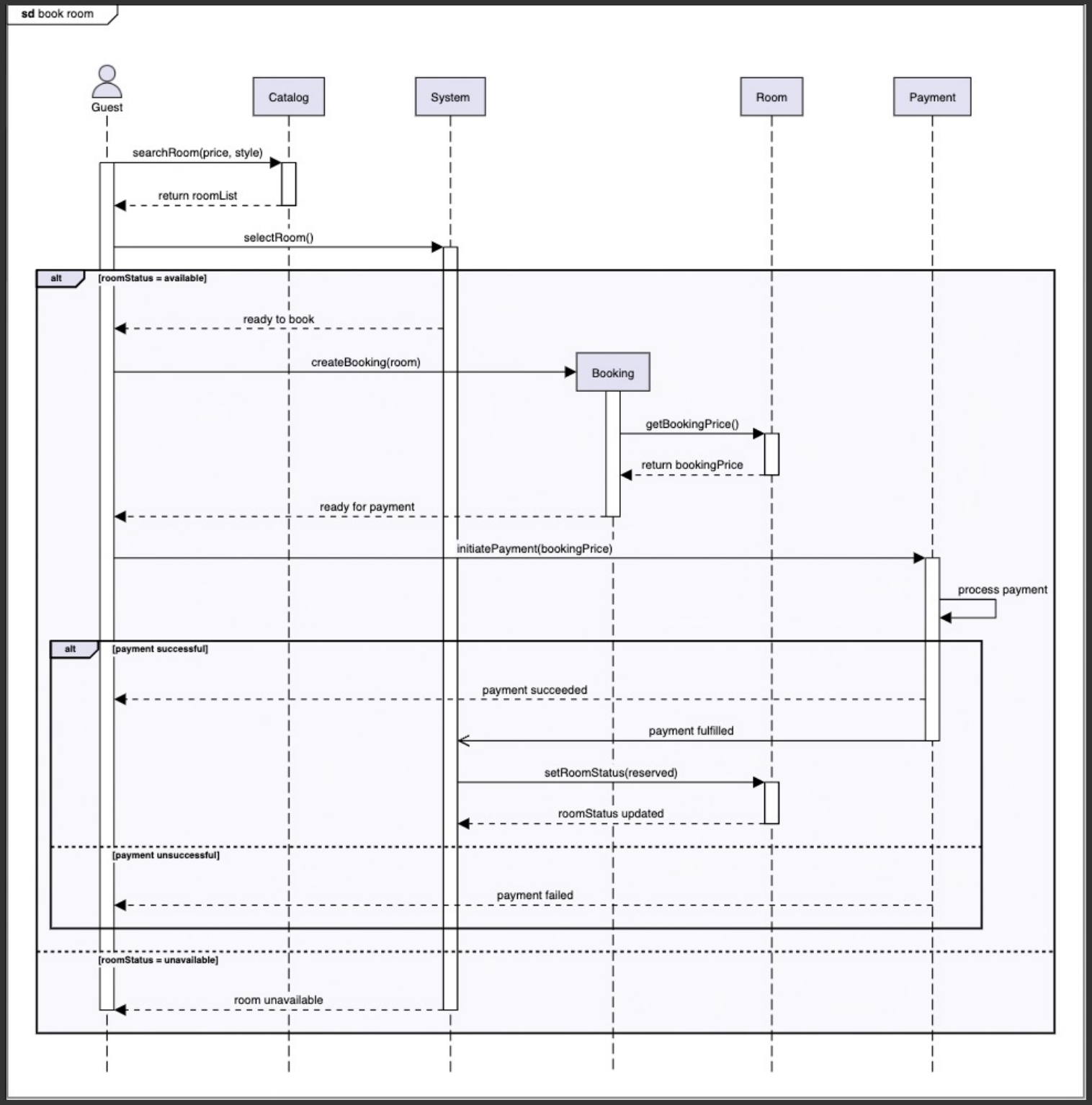


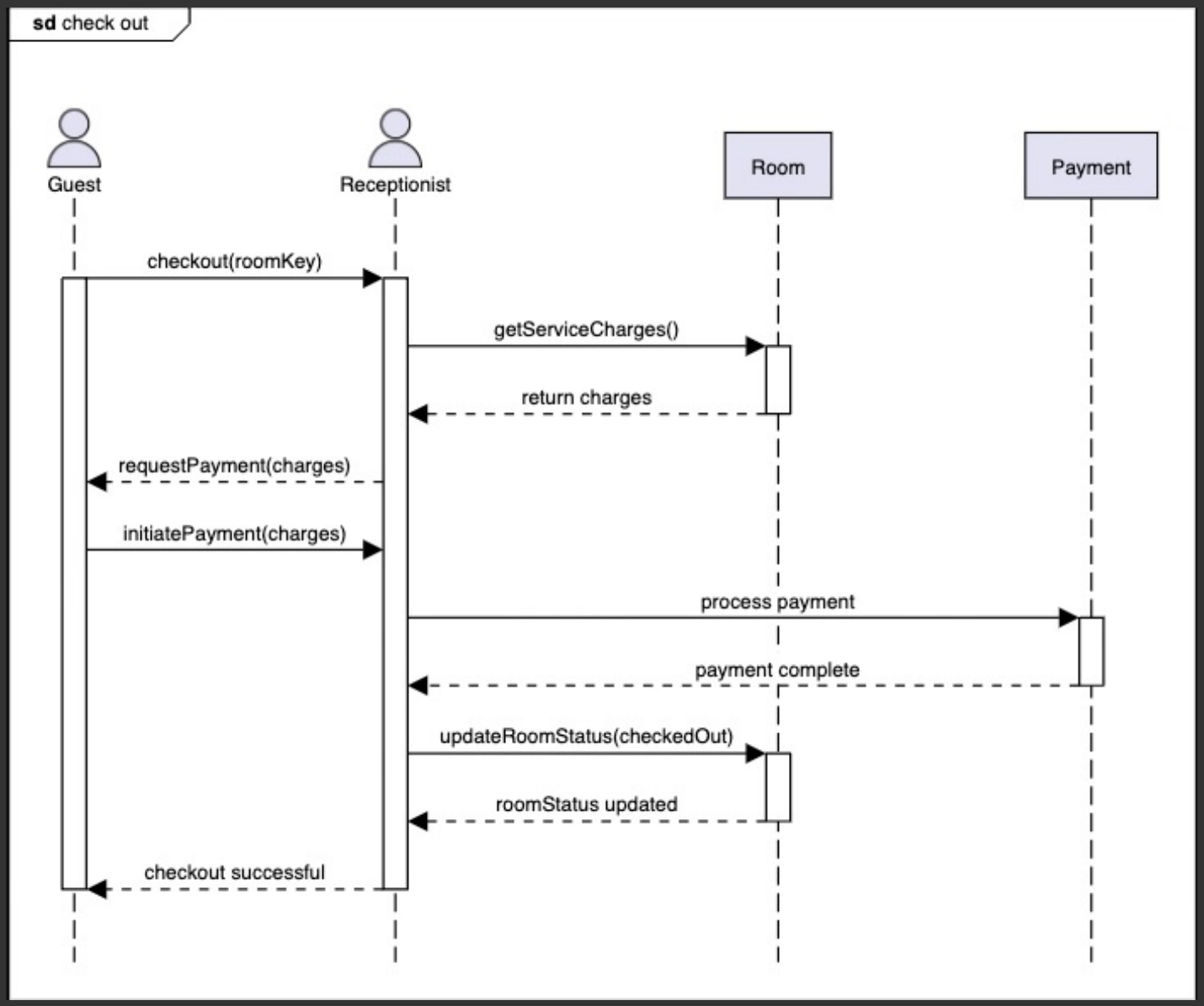

CheckTransaction
- bankName : string
- checkNumber : string
CreditCardTransaction
- nameOnCard : string
- zipCode : string
CashTransaction
- cashTendered : double
extends
Invoice
- amount : double
+ createBill( ) : bool
<<Abstract>>
BillTransaction
- creationDate : date
- amount : double
- status : PaymentStatus
+ initiateTransaction( ) : bool
SMSNotification
+send(person) : void
EmailNotification
+send(person) : void
extends
<<Abstract>>
Notification
- notificationId : int
- createdOn : date
- content : string
+send(person) : void
RoomBooking
- reservationNumber : string
- startDate : date
- durationInDays : int
- status : BookingStatus
- checkout : date/time
- advancePayment : double
+ fetchDetails( ) : RoomBooking
<<Abstract>>
Service
- issuedAt : date
+ addinvoiceItem( ) : bool
extends
RoomService
- isChargable : bool
- requestTime : date
Amenity
- name : string
- description : string
KitchenService
- description : string
Room
- roomNumber : string
- style : RoomStyle
- status : RoomStatus
- bookingPrice : double
- isSmoking : bool
+ isRoomAvailable( ) : bool
+ checkIn( ) : bool
+ checkOut( ) : bool
<<Abstract>>
Person
- name : string
- address : Address
- email : string
- phoneNumber : string
- accountType : AccountType
extends
Receptionist
+ createBooking( ) : bool
Guest
- totalRoomsCheckedIn : int
HouseKeeper
+ assignToRoom( ) : bool
<<interface>>
Search
+ searchRoom(roomStyle, date, duration) : Room {list}
RoomHousekeeping
- description : string
- startDateTime : date/time
- int : duration
+ addHousekeeping( ) : bool
HotelBranch
- name : string
- address : Address
+ getRooms( ) : ListofRoom
RoomKey
- keyId : string
- barcode : string
- issuedAt : date
- active : bool
- isMaster : bool
+ assignRoom( ) : bool
+ isActive( ) : bool
Account
- id : string
- password : string
- status : AccountStatus
+ resetPassword( ) : bool
Catalog
- rooms : Map<string, Room {list}
implements
Catalog
- rooms : Map<string, Room{list}
implements
Hotel
- name : string
+ addlocation( ) : bool

Suppose an interviewer is having discount on transaction

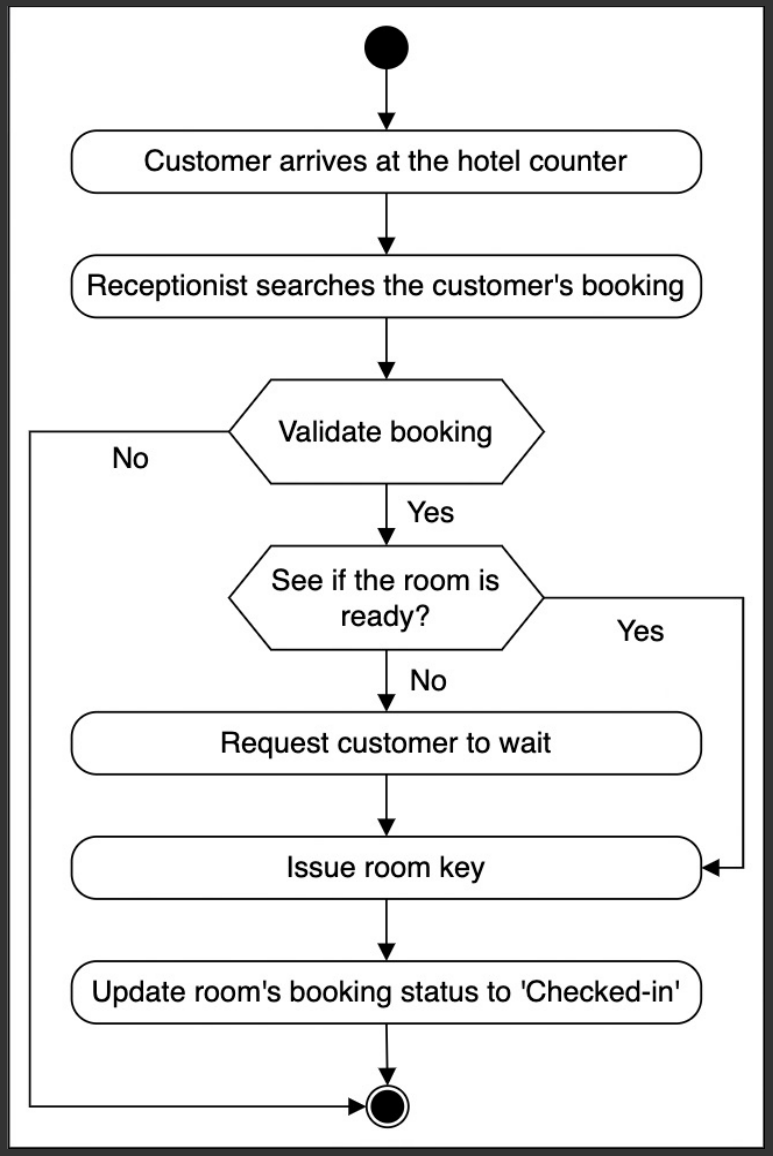


Sequence Diagram

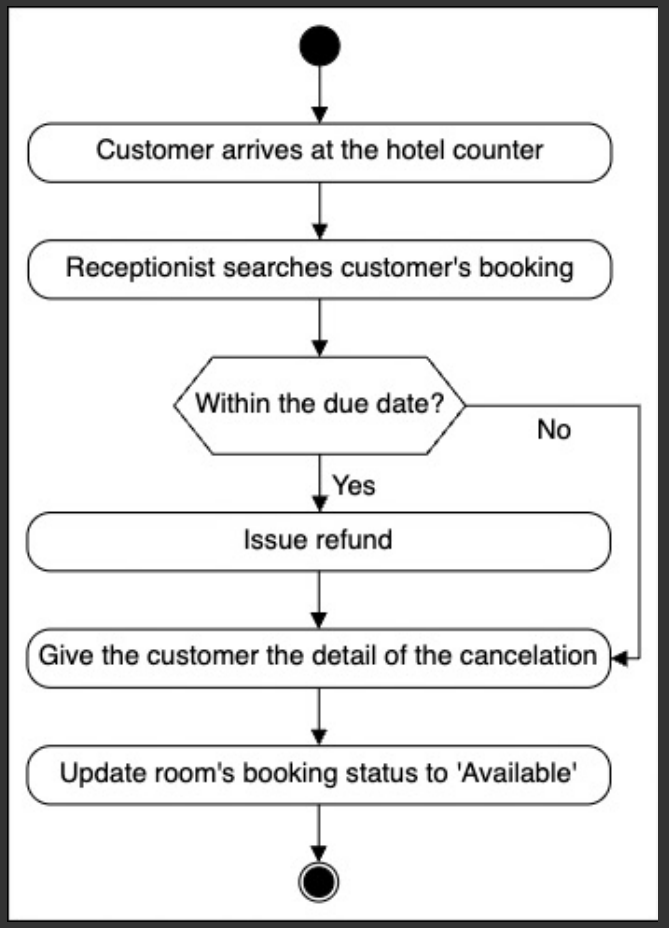



sd check out
Guest
Receptionist
Room
Payment
checkout(roomKey)
getServiceCharges()
return charges
requestPayment(charges)
initiatePayment(charges)
process payment
payment complete
updateRoomStatus(checkedOut)
roomStatus updated
checkout successful

checkout



Activity diagram for hotel check-in



Checkout

Code For Hotel Management System

Enumeration

```
// definition of enumerations used in hotel management system
enum RoomStyle {
    STANDARD,
    DELUXE,
    FAMILY_SUITE,
    BUSINESS_SUITE
}

enum RoomStatus {
    AVAILABLE,
    RESERVED,
    OCCUPIED,
    NOT_AVAILABLE,
    BEING_SERVICED,
    OTHER
}

enum BookingStatus {
    REQUESTED,
    PENDING,
    CONFIRMED,
    CANCELLED,
    ABANDONED
}

enum AccountStatus {
    ACTIVE,
    CLOSED,
    CANCELED,
    BLACKLISTED,
    BLOCKED
}

enum AccountType {
    MEMBER,
    GUEST,
    MANAGER,
    RECEPTIONIST
}

enum PaymentStatus {
    UNPAID,
    PENDING,
    COMPLETED,
    FILLED,
    DECLINED,
    CANCELLED,
    ABANDONED,
    SETTLING,
    SETTLED,
    REFUNDED
}
```

1. Address & Account

```
public class Address {
    private String streetAddress;
    private String city;
    private String state;
    private int zipCode;
    private String country;
}

public class Account {
    private String id;
    private String password;
    private AccountStatus status;

    public boolean resetPassword();
}
```

2. Person

```
public abstract class Person {
    private String name;
    private Address address;
    private String email;
    private String phone;
    private Account account;
}


public class Guest extends Person {
    private int totalRoomsCheckedIn;

    public List<RoomBooking> getBookings();
}

public class Receptionist extends Person {
    public List<Member> searchMember(String
name);
    public boolean createBooking();
}

public class Housekeeper extends Person {
    public boolean assignToRoom();
}
```

3. Service

```
public abstract class Service {
    private Date issueAt;

    public boolean addInvoiceItem(Invoice invoice);
}

public class Amenity extends Service {
    private String name;
    private String description;
}

public class RoomService extends Service {
    private boolean isChargeable;
    private Date requestTime;
}

public class KitchenService extends Service {
    private String description;
}
```

4. Invoice

```
public class Invoice {
    private double amount;

    public boolean createBill();
}
```

5. Room Booking

```
public class RoomBooking {
    private String reservationNumber;
    private Date startDate;
    private int durationInDays;
    private BookingStatus status;
    private Date checkin;
    private Date checkout;

    private int guestId;
    private Room room;
    private Invoice invoice;
    private List<Notification> notifications;

    public static RoomBooking fectchDetails(String reservationNumber);
}
```

6. Bill Transaction

```
// BillTransaction is an abstract class
public abstract class BillTransaction {
    private Date creationDate;
    private double amount;
    private PaymentStatus status;

    public abstract void
initiateTransaction();
}

class CheckTransaction extends BillTransaction
{
    private String bankName;
    private String checkNumber;

    public void initiateTransaction() {
        // functionality
    }
}

class CreditCardTransaction extends
BillTransaction {
    private String nameOnCard;
    private int zipcode;

    public void initiateTransaction() {
        // functionality
    }
}

class CashTransaction extends BillTransaction
{
    private double cashTendered;

    public void initiateTransaction() {
        // functionality
    }
}
```

7. Notification

```
// Notification is an abstract class
public abstract class Notification {
    private int notificationId;
    // The Date data type represents and
deals with both date and time.
    private Date createdOn;
    private String content;

    public abstract void
sendNotification(Person person);
}

class SMSNotification extends Notification {

    public void sendNotification(Person
person) {
        // functionality
    }
}

class EmailNotification extends Notification
{

    public void sendNotification(Person
person) {
        // functionality
    }
}
```

8. Room, RoomKey & Housekeeping

```
public class Room {
    private String roomNumber;
    private RoomStyle style;
    private RoomStatus status;
    private double bookingPrice;
    private boolean isSmoking;
    private List<RoomKey> keys;
    private List<RoomHousekeeping> housekeepingLog;

    public boolean isRoomAvailable();
    public boolean checkin();
    public boolean checkout();
}

public class RoomKey {
    private String keyId;
    private String barcode;
    private Date issuedAt;
    private boolean isActive;
    private boolean isMaster;

    public boolean assignRoom(Room room);
}

public class RoomHousekeeping
{
    private String description;
    private Date startDatetime;
    private int duration;
    private Housekeeper housekeeper;

    public boolean addHousekeeping(Room room);
}
```

9. Search & Catalog

```
public interface Search {
    public static List<Room> search(RoomStyle
style, Date date, int duration);
}

public class Catalog implements Search {
    private List<Room> rooms;

    public List<Room> search(RoomStyle style,
Date date, int duration);
}
```

10. Hotel & Hotel Branch

```
public class HotelBranch {
  private String name;
  private Address location;

  public List<Room> getRooms();
}

public class Hotel {
  private String name;
  private List<HotelBranch> locations;

  public boolean addLocation(HotelBranch
location);
}
```