# Class Diagram for the Hotel Management System

Learn to create a class diagram for the hotel management system problem using the bottom-up approach.

**We'll cover the following**

- Components of a hotel management system
  - Address and Account
  - Person
  - Service
  - Invoice
  - Room booking
  - Notification
  - Room, room key, and room housekeeping
  - Search interface and catalog
  - Bill transaction
  - Hotel and hotel branch
  - Enumerations
- Relationship between the classes
  - Association
    - One-way association
    - Two-way association
  - Aggregation
  - Composition
  - Inheritance
- Class diagram of the hotel management system
- Design pattern
- Additional requirements

Here, we are going to create the class diagram for our system on the basis of requirements that we gathered in one of the previous lessons. In the class diagram, we will first design and create the classes, abstract classes, and interfaces for the system, and then we'll identify the relationship between classes in accordance with all the requirements of the hotel management system.

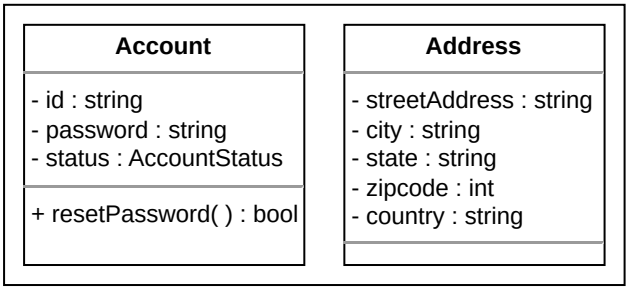
## Components of a hotel management system

In this section, we'll define the classes for a hotel management system. Since we are following the bottom-up approach to designing a class diagram, we will create the classes of small components first. Next, we will integrate these components and create the class diagram for the entire hotel management system.

### Address and Account

The `Address` is a class that is required to store any address. The `Address` is a custom data type that has attributes like a street address, city, etc. In the hotel management system, this class will be used to specify the address of the users and the hotel.

`Account` is a class that is used to store the account information of the user. This class has three members, i.e., account ID, password, and the status of the account. The class representation of `Address` and `Account` classes is as follows:




The class diagram of the Address and Account classes

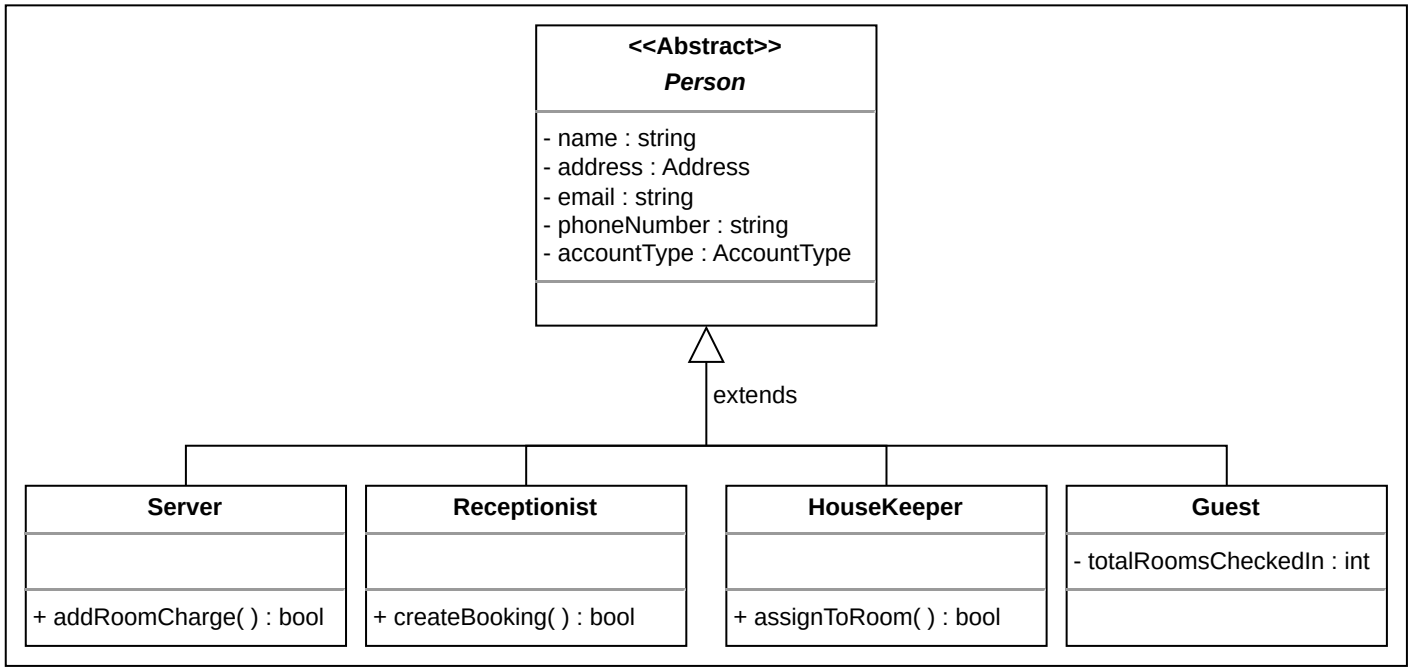
### Person

`Person` is an abstract class used to store information related to a person like a name, email, phone number, etc. In this class, there is an object of the `Address` type

to specify the person's address. The `Person` class specifies the accounts in the system. There can be four types of accounts in the system,i.e., housekeeper, receptionist, guest, and server.

There are multiple functions of the `Person` class's subclasses. First, the `Housekeeper` class will keep track of the housekeeping records of a room. Second, the `Receptionist` class represents the hotel receptionist. The methods in this class depict the actions that can be performed by the receptionist. Moreover, the `Guest` class describes the guests of the hotel. Guests are the customers of the hotel who can search for and book a room. Whereas, the `Server` class will handle the room service.

The relationship diagram for these classes is shown below:




The class diagram of Person and its derived classes

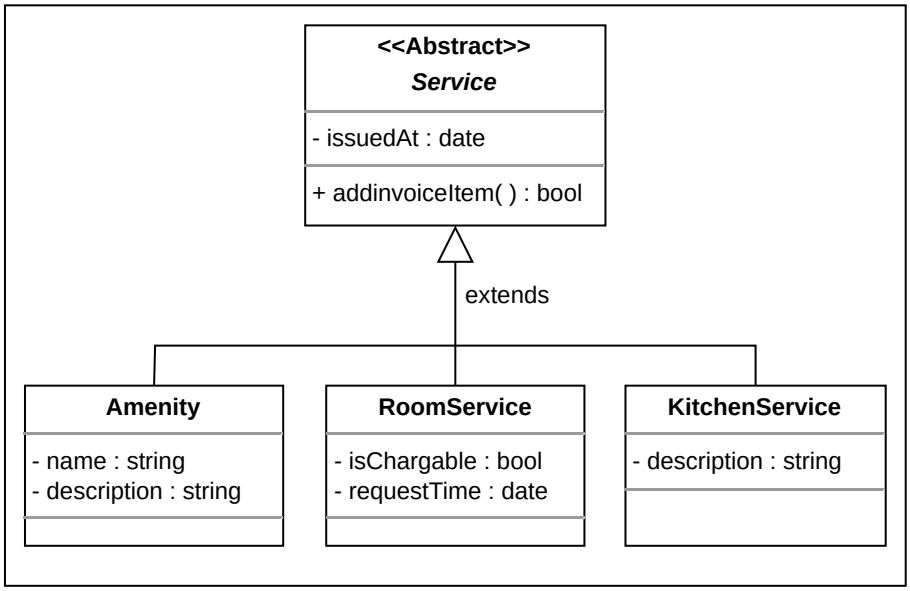
R1: Hotel Management System

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

## Service

`Service` is an abstract class that encapsulates the details of different types of services that guests have requested. There are three types of services provided—amenity, room service, and kitchen service.

The `Amenity` class is a subclass of `Service` having two members; name and description. Similarly, `RoomService` is also inherited from the `Service` class. This class stores information about room services whether these services are chargeable or not and what is the request time of the service. Furthermore, the last child class of the `Service` is the `KitchenService`. The relationship between these classes is shown in the illustration below.



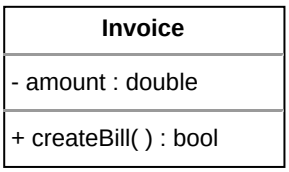

The class diagram of Service and its derived classes

R8: Hotel Management System

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

## Invoice

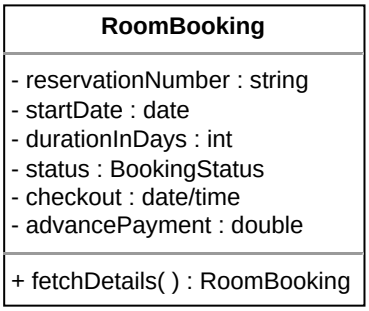
The `Invoice` class represents the billing system in the hotel management system. The UML diagram for both classes is presented below:




The class diagram of the Invoice class

## Room booking

The `RoomBooking` class is responsible for managing the bookings for a room. This class consists of attributes like reservation number, start date, duration, etc. Moreover, this class has a member of the `BookingStatus` type that is used to store the status of the room booking. The UML representation of this class is as follows:




The class diagram of the RoomBooking class

R4: Hotel Management System

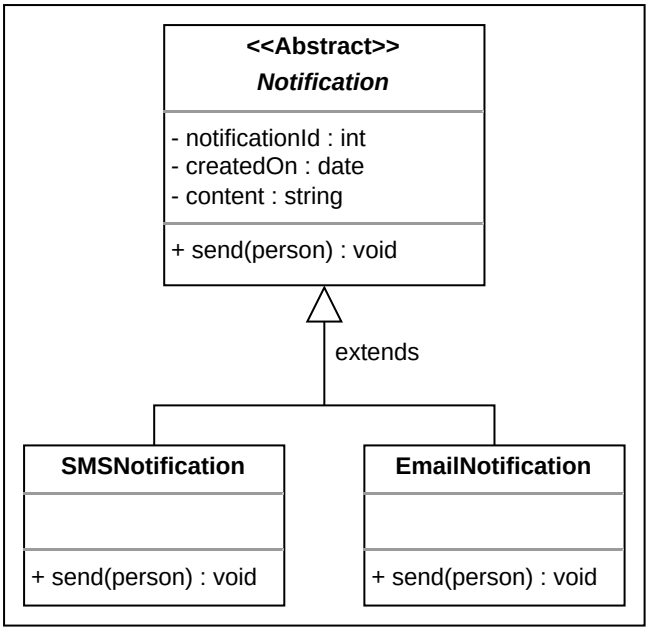
**R4:** During room booking, the user will enter the check-in date and duration of the stay. The user would also have to give some advance payment.

## Notification

`Notification` is an abstract class. This class is responsible for sending notifications to guests whenever the booking is nearing the check-in or check-out date. Every

notification has an ID, creation date, and content in it. The notification can either be an SMS notification or an email notification.

The `SMSNotification` class requires the phone number of the member to send a notification. On the other hand, the `EmailNotification` needs the email address of the member to send a notification. The relationship diagram of these classes is shown here:




The class diagram of Notification and its derived classes

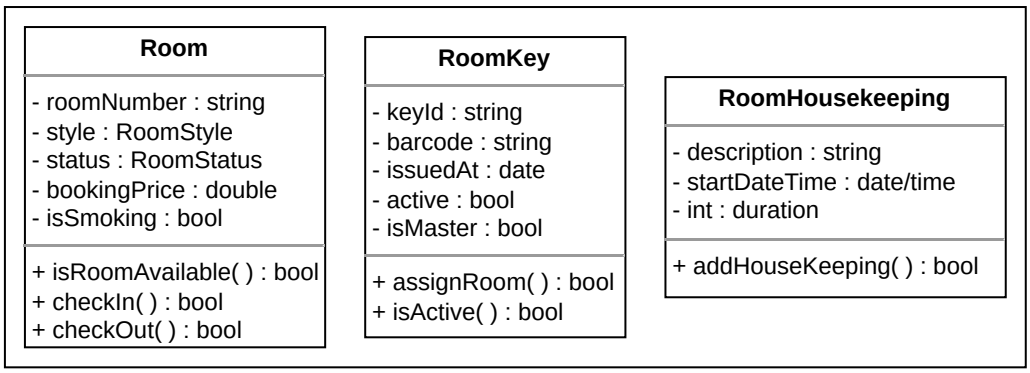
R6: Hotel Management System

**R6:** The system should send a notification to the customers about the booking status or other information.

## Room, room key, and room housekeeping

The `Room` class is the basic building block of the system. Every room has a room number and price associated with it. The `Room` class uses the `RoomStyle` and `RoomStatus` enums to specify the style and status of the rooms, respectively.

Each room has an electronic key card associated with it. The `RoomKey` class expresses the electronic key card. Each card has its own unique ID and barcode on it. The `RoomKey` class also has members to store the issue date, to check whether or not the key is active, and to check whether or not a key is a master key. Whereas, `RoomHousekeeping` is a class used to keep track of all housekeeping records for the rooms. The UML representation of these classes is as follows:




The class diagram of the Room, RoomKey, and RoomHousekeeping classes

R7, and R9: Hotel Management System

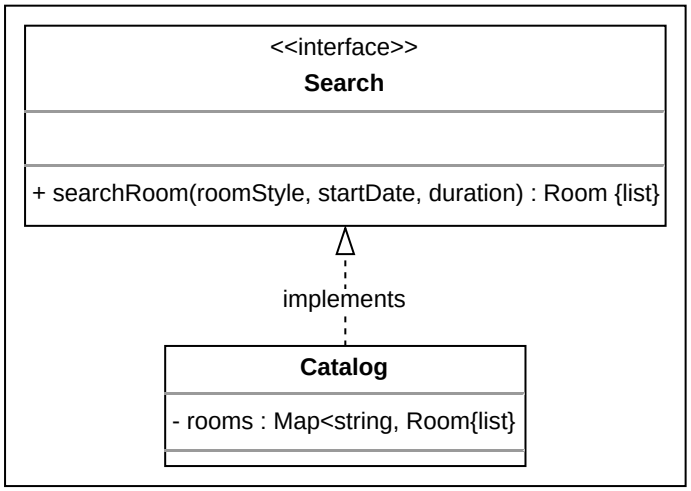
**R9:** Every room should have its own specific key, and there can be a master key that opens a specific set of rooms.

**R7:** All the housekeeping tasks should be logged in and managed by the system.

## Search interface and catalog

`Search` is one of the most important components of the hotel management system. In the diagram below, `Search` is the interface that allows the guest to search for any room of their choice and pay range. The receptionist can also use this interface to search for any room. The `Catalog` class contains a list of all rooms and implements the `Search` interface.

The following UML diagram shows this relationship:



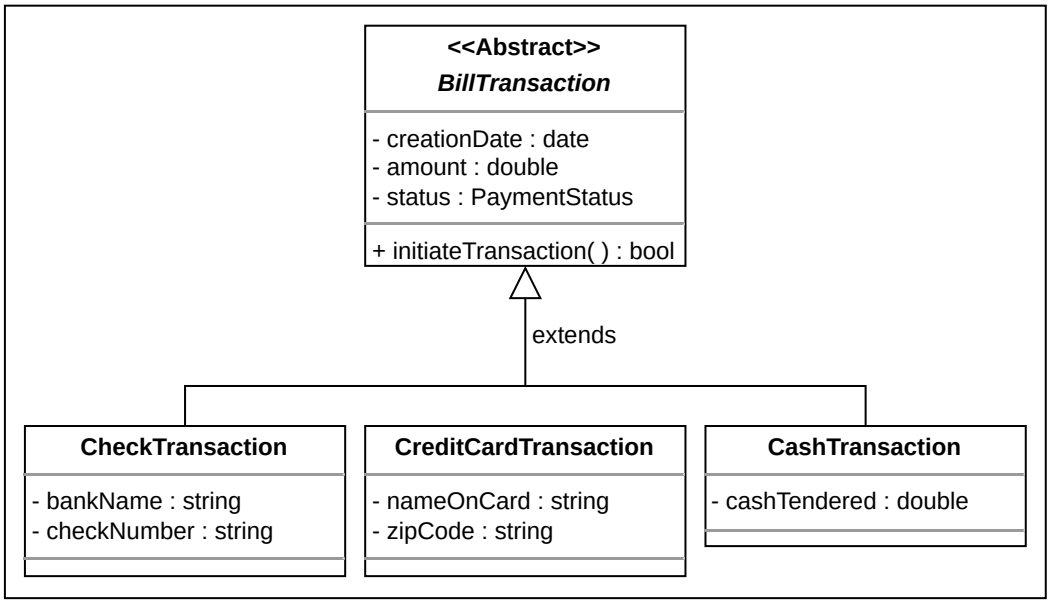

The class diagram of the Search interface and the Catalog class

R3: Hotel Management System

**R3:** The system should allow the guests to search for any room and book any of the available rooms.

## Bill transaction

After generating an invoice, a customer needs to pay the bill to confirm the booking of the room. A `BillTransaction` class is required to store the information of bill payment. Three ways to pay the bill are check transaction, cash transaction, and credit card transaction. We can define the bill payment functionality through any payment method using the diagram below:

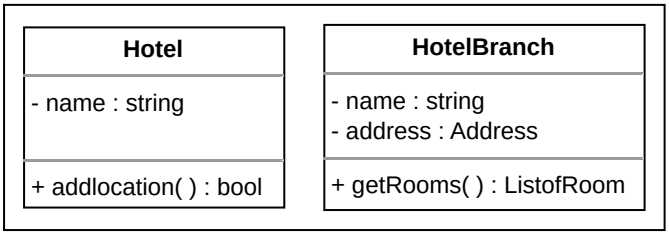



The class diagram of BillTransaction and its derived classes

## Hotel and hotel branch

In this section, we'll look at the `Hotel` and `HotelBranch` classes. According to the requirements, there can be multiple branches of the hotel. `HotelBranch` is a class used to represent the location of the hotel branch. This class consists of two members: `name` and `Address`. The string type `name` is used to store the name of the hotel branch, while the complex object `Address` is used to store the complete address of a branch.

The `Hotel` class is the base class of the system which is used to represent the hotel. The visual representation of these classes is as follows:




The class diagram of the Hotel and HotelBranch classes

R10: Hotel Management System

**R10:** A hotel can have multiple branches of it.

## Enumerations

Here is the list of enumerations required in the hotel management system:

`BookingStatus`: This status describes the status of the booking whether the booking is requested, pending, confirmed, canceled, or abandoned.

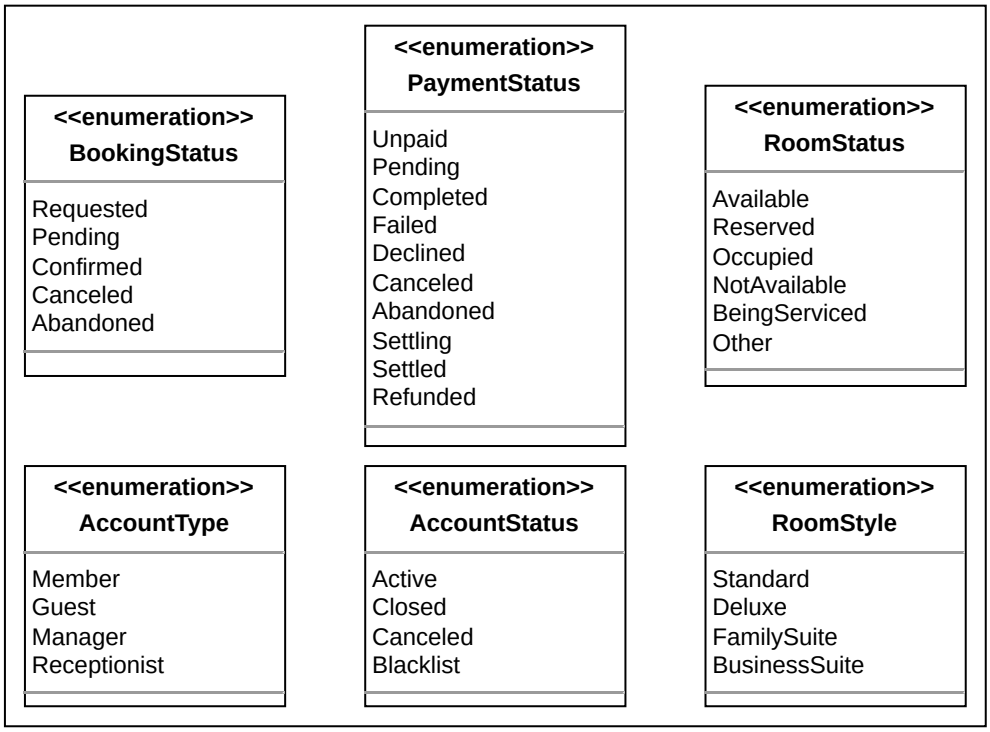
`PaymentStatus`: This status describes the status of the booking's payment, whether it is unpaid, pending, completed, failed, declined, canceled, abandoned, setting, settled, or refunded.

`RoomStatus`: This status describes the status of the room, whether it is available, reserved, occupied, not available, being serviced, or any other possibility.

`RoomStyle`: This describes the style of the room that the user wants to book. The style could be standard, deluxe, family suite, or business suite.

`AccountStatus`: This status tells the status of the user account whether it is active, closed, canceled, or blocklisted.

`AccountType`: The account type tells the type of the account of the user, whether it is a member, guest, manager, or receptionist.




Enums in the hotel management system

R2: Hotel Management System

**R2:** The rooms can be of different styles like standard, deluxe, family suite, or business suite.

## Relationship between the classes

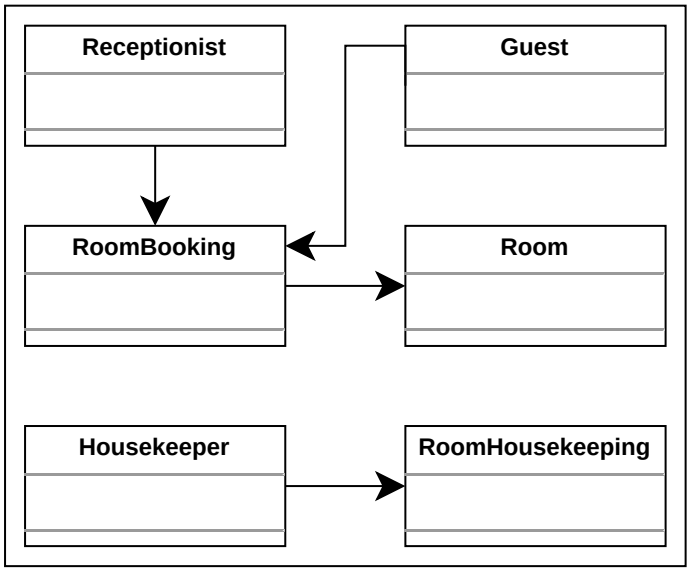
Now, we'll discuss the relationships between the classes we have defined above in our hotel management system.

### Association

The class diagram has the following association relationships:

#### One-way association

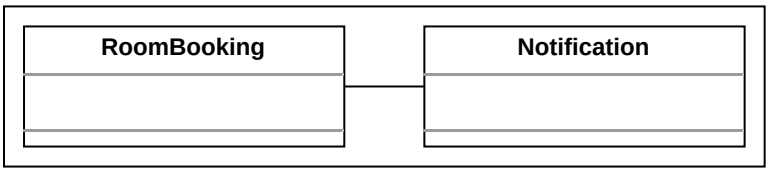
- The `Housekeeper` class has a one-way association with `RoomHousekeeping`.
- Both `Receptionist` and `Guest` have a one-way association with `RoomBooking`.
- The `RoomBooking` class has a one-way association with `Room`.




A one-way association relationship between classes

### Two-way association

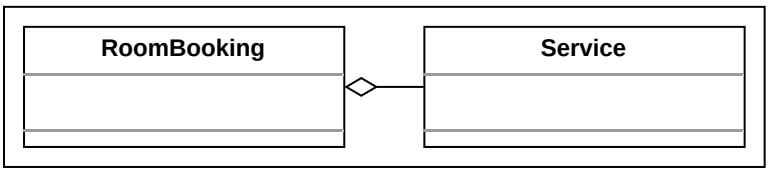
The `RoomBooking` class has a two-way association with `Notification`.




A two-way association relationship between classes

## Aggregation

- The `RoomBooking` class is an aggregate of `Service`.

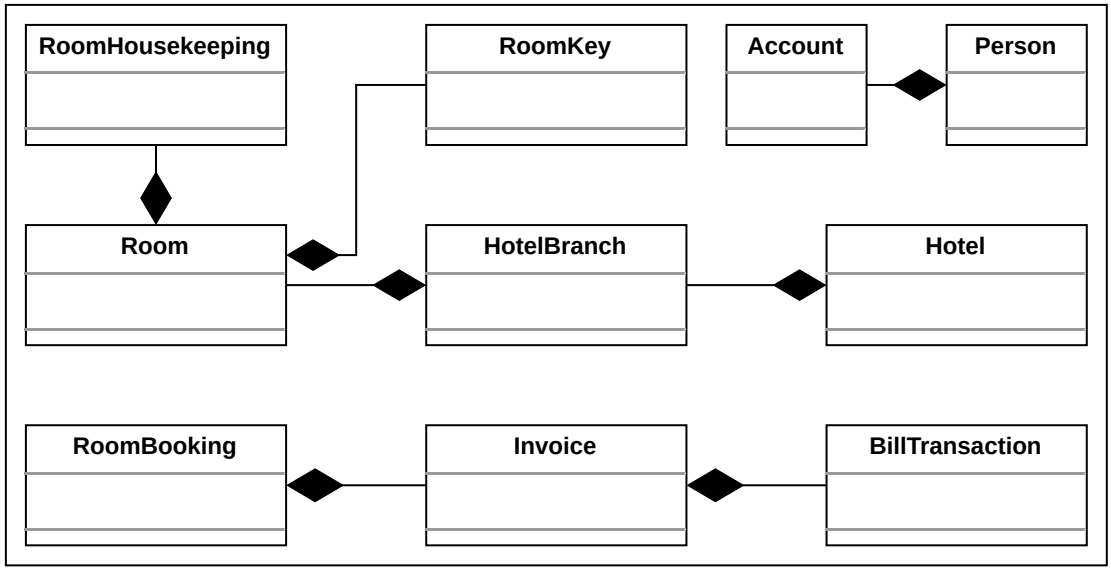



An aggregation relationship between classes

## Composition

- The `RoomBooking` class is composed of `Invoice`.
- The `Invoice` class is composed of `BillTransaction`.
- The `Person` class is composed of `Account`.

- The `Room` class is composed of `RoomHousekeeping` and `RoomKey`.
- The `HotelBranch` class is composed of `Room`.
- The `Hotel` class is composed of `HotelBranch`.




A composition relationship between classes

### Inheritance

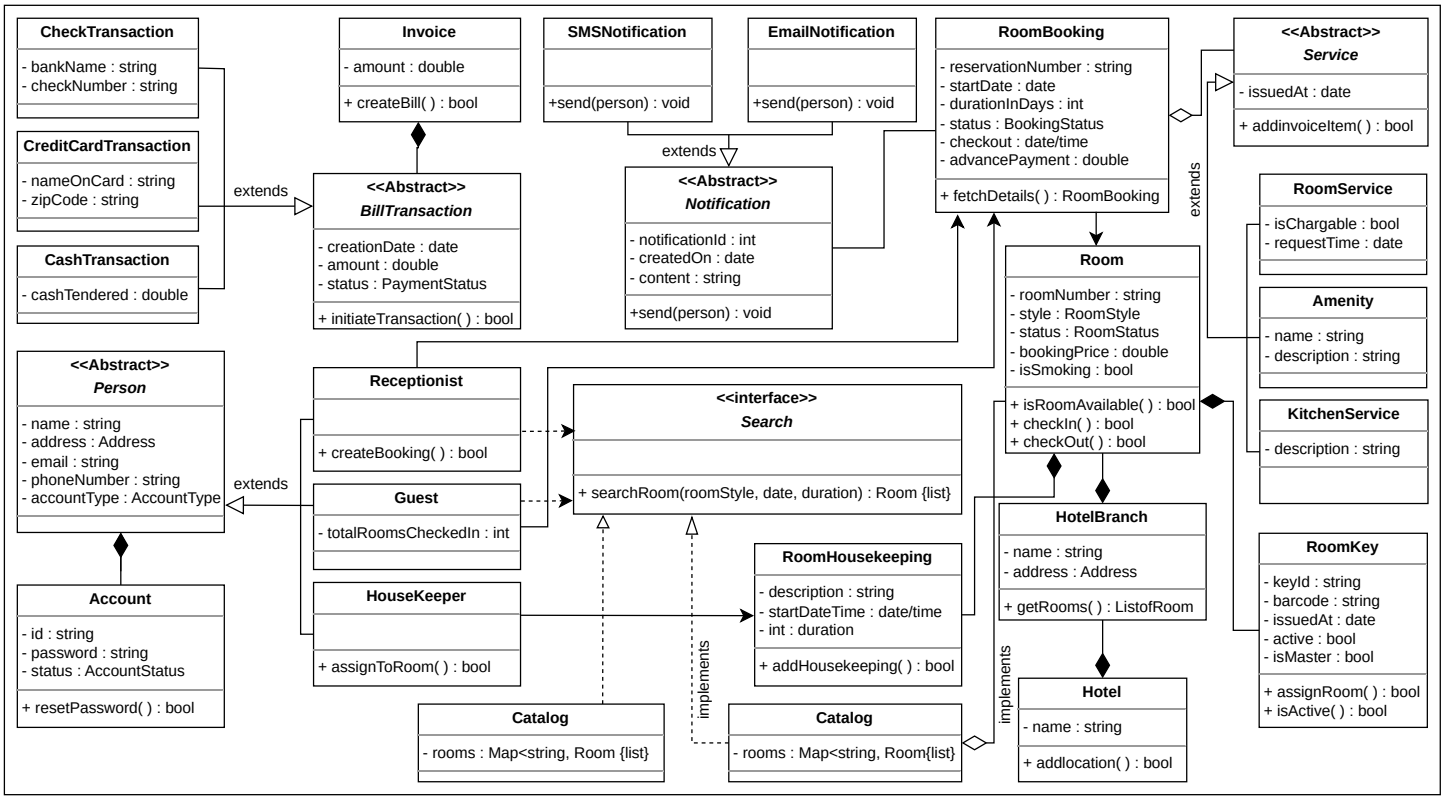
The following classes show an inheritance relationship:

- `Receptionist`, `Guest`, `Housekeeper`, and `Server` classes extend the `Person` class.
- `Amenity`, `RoomService`, and `KitchenService` classes extend the `Service` class.
- Both `PostalNotification` and `EmailNotification` classes extend the `Notification` class.
- The `Catalog` class implements the `Search` interface.

> **Note:** We have already discussed the inheritance relationship between classes in the component section above one by one.

## Class diagram of the hotel management system

Here is the complete class diagram for our hotel management system:




The class diagram of hotel management system

## Design pattern

The Strategy design pattern is applied here, which will design a separate strategy or algorithm to calculate the rate of each room.

The `Hotel` class follows the Singleton design pattern, because there will only be a single instance of the `Hotel` class.

Inside the hotel management system, there can be multiple rooms in a hotel, and each room has its own formula for calculating the booking cost. Therefore, `Room` and related classes incorporate the properties of the Factory design pattern.

## Additional requirements

The interviewer can introduce some additional requirements in the given hotel management system, or they can ask some follow-up questions. Let's see examples of the additional requirements:

**Discount:** A discount will be applied to the payment depending on special events such as the new year, branch opening, and so on. The class diagram provided