In a luxury hotel, guests can book rooms for a certain duration. Before booking, a guest can check room availability by calling the receptionist. Each room has specific features such as the number of beds, bed size, and window view. The system allows guests to view these details and select a room that suits their preferences.

Once the guest finds an available room, they must book it for an active stay. The system records the guest's details, including their name, contact information, and booking history. After the booking is confirmed, the guest is assigned a room number, and the room's availability status is updated.

The hotel has multiple employees, among them managers overseeing the booking process. A manager is responsible for approving bookings, preparing the bill, and applying discounts when available. Discounts are occasionally offered on some rooms, and they are automatically deducted from the total bill when the guest checks out.

Throughout their stay, guests may request housekeeping services but it is optional. Upon request, housekeepers are assigned to specific rooms and are responsible for cleaning the rooms and delivering food from the hotel's attached **restaurant**. Guests can view the menu and place food orders by calling the receptionist. The receptionist then assigns a housekeeper to deliver the ordered food to the guest's room.

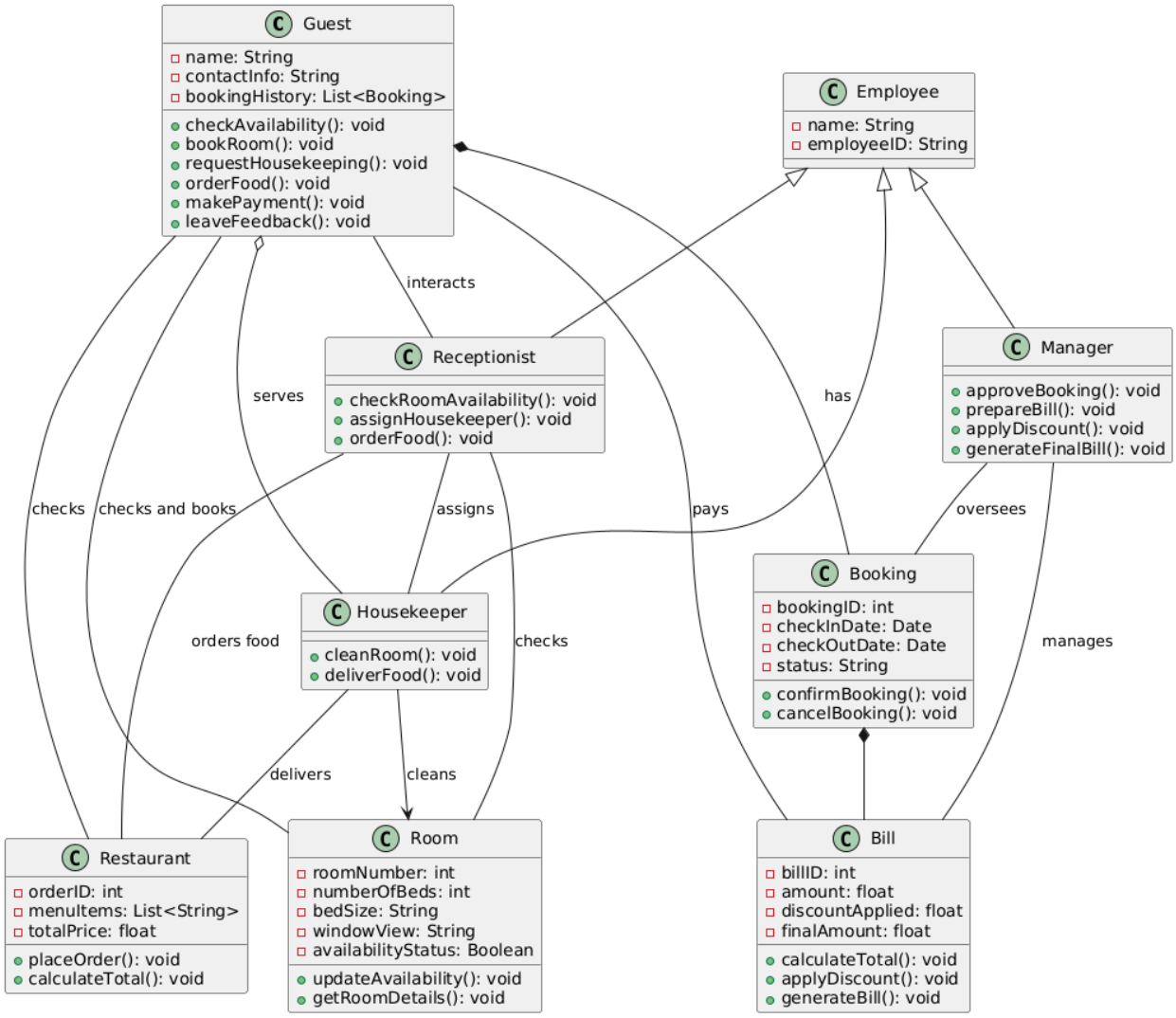
Each guest's bill is prepared without any discrepancies or miscalculations when they check out. The manager calculates the total amount, applies any applicable discounts, and generates a final bill. The guest then makes the payment and leaves feedback about their stay. The system should maintain the data integrity of the guests and provide a responsive user interface for smooth operation.

1. Design a class diagram that reflects the above system best. Use standard rules and notations. [12]
2. Identify three functional and three non-functional requirements from the above scenario. [3]

Solution:

Green - Functional requirements
Yellow - Non-functional requirements

Find the possible solution to the class diagram on the next page (provide the **multiplicities** according to your preference, but it should be realistic)


Guest
- name: String
- contactInfo: String
- bookingHistory: List<Booking>
+ checkAvailability(): void
+ bookRoom(): void
+ requestHousekeeping(): void
+ orderFood(): void
+ makePayment(): void
+ leaveFeedback(): void
Employee
- name: String
- employeeID: String
Receptionist
+ checkRoomAvailability(): void
+ assignHousekeeper(): void
+ orderFood(): void
Manager
+ approveBooking(): void
+ prepareBill(): void
+ applyDiscount(): void
+ generateFinalBill(): void
Housekeeper
+ cleanRoom(): void
+ deliverFood(): void
Booking
- bookingID: int
- checkInDate: Date
- checkOutDate: Date
- status: String
+ confirmBooking(): void
+ cancelBooking(): void
Restaurant
- orderID: int
- menuItems: List<String>
- totalPrice: float
+ placeOrder(): void
+ calculateTotal(): void
Room
- roomNumber: int
- numberOfBeds: int
- bedSize: String
- windowView: String
- availabilityStatus: Boolean
+ updateAvailability(): void
+ getRoomDetails(): void
Bill
- billID: int
- amount: float
- discountApplied: float
- finalAmount: float
+ calculateTotal(): void
+ applyDiscount(): void
+ generateBill(): void
interacts
serves
has
checks
checks and books
assigns
pays
oversees
orders food
checks
manages
delivers
cleans