Assignment 2

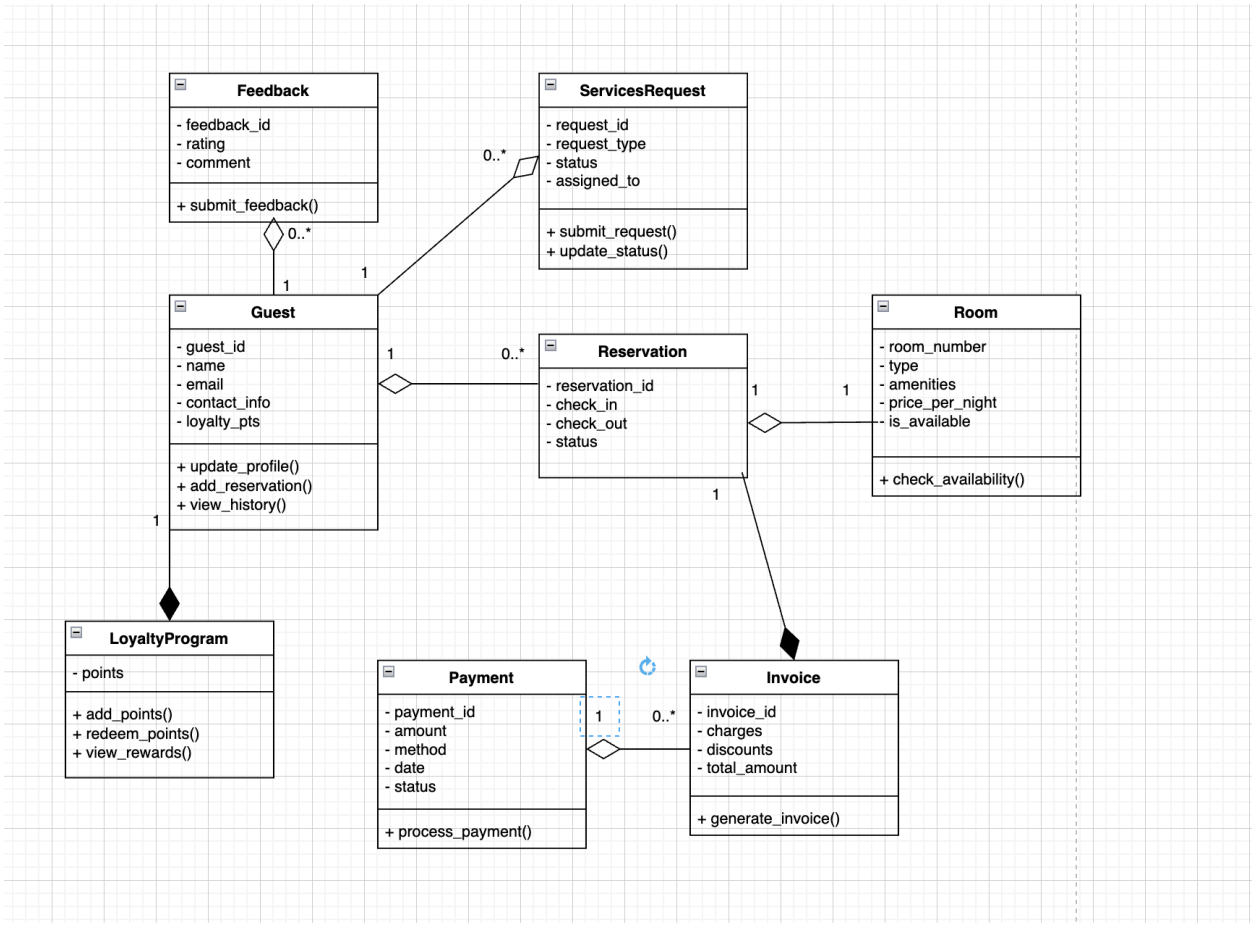
ICS220 > 22111 Program. Fund. >

Zayed University

Sultan Allanjawi / 202307697

Prof: Sujith Mathew

UML Diagram:



Code:

```
class Room:
    """
    Represents a hotel room with specific details.

    """
    def __init__(self, room_number, room_type, amenities, price_per_night,
is_available=True):
        self.__room_number = room_number
        self.__type = room_type
        self.__amenities = amenities
        self.__price_per_night = price_per_night
```

```
        self.__is_available = is_available

    def check_availability(self):
        return self.__is_available

    def set_availability(self, availability):
        self.__is_available = availability

    def get_price(self):
        return self.__price_per_night

    def __str__(self):
        return f"Room {self.__room_number} ({self.__type}) - {'Available'
if self.__is_available else 'Booked'}"


class Guest:
    """
    Represents a hotel guest.

    """
    def __init__(self, guest_id, name, email, contact_info):
        self.__guest_id = guest_id
        self.__name = name
        self.__email = email
        self.__contact_info = contact_info
        self.__loyalty_pts = 0
        self.__reservations = []

    def update_profile(self, name=None, email=None, contact_info=None):
        if name: self.__name = name
        if email: self.__email = email
        if contact_info: self.__contact_info = contact_info

    def add_reservation(self, reservation):
        self.__reservations.append(reservation)

    def view_history(self):
        return self.__reservations
```

```
    def __str__(self):
        return f"Guest: {self.__name}, Email: {self.__email}"


class Reservation:
    """
    Represents a reservation made by a guest for a specific room.

    """
    def __init__(self, reservation_id, guest, room, check_in, check_out):
        self.__reservation_id = reservation_id
        self.__guest = guest
        self.__room = room
        self.__check_in = check_in
        self.__check_out = check_out
        self.__status = "Pending"

    def calculate_total(self):
        nights = (self.__check_out - self.__check_in).days
        return nights * self.__room.get_price()

    def confirm_reservation(self):
        self.__status = "Confirmed"
        self.__room.set_availability(False)

    def cancel_reservation(self):
        self.__status = "Cancelled"
        self.__room.set_availability(True)

    def __str__(self):
        return f"Reservation {self.__reservation_id}: {self.__status}"


class Invoice:
    """
    Represents an invoice for a reservation.

    """
    def __init__(self, invoice_id, reservation, charges=None, discounts=0):
        self.__invoice_id = invoice_id
```

```
        self.__reservation = reservation
        self.__charges = charges if charges else []
        self.__discounts = discounts
        self.__total_amount = 0

    def generate_invoice(self):
        base = self.__reservation.calculate_total()
        extra = sum(self.__charges)
        self.__total_amount = base + extra - self.__discounts
        return self.__total_amount

    def __str__(self):
        return f"Invoice {self.__invoice_id} - Total:
${self.__total_amount}"


class Payment:
    """
    Represents a payment transaction.

    """
    def __init__(self, payment_id, amount, method, date):
        self.__payment_id = payment_id
        self.__amount = amount
        self.__method = method
        self.__date = date
        self.__status = "Pending"

    def process_payment(self):
        self.__status = "Paid"

    def __str__(self):
        return f"Payment {self.__payment_id} - {self.__status} via
{self.__method}"


class LoyaltyProgram:
    """
    Manages the loyalty points of a guest.
```

```
    """
    def __init__(self):
        self.__points = 0

    def add_points(self, points):
        self.__points += points

    def redeem_points(self, amount):
        if amount <= self.__points:
            self.__points -= amount
            return True
        return False

    def view_rewards(self):
        return self.__points

    def __str__(self):
        return f"Loyalty Points: {self.__points}"


class ServiceRequest:
    """
    Represents a guest's service request (e.g., housekeeping).

    """
    def __init__(self, request_id, guest, request_type):
        self.__request_id = request_id
        self.__guest = guest
        self.__request_type = request_type
        self.__status = "Pending"
        self.__assigned_to = None

    def submit_request(self):
        self.__status = "Submitted"

    def update_status(self, new_status):
        self.__status = new_status

    def __str__(self):
```

```
        return f"ServiceRequest {self.__request_id} - {self.__request_type}
[{self.__status}]"


class Feedback:
    """
    Stores guest feedback about their stay.

    """
    def __init__(self, feedback_id, guest, rating, comment):
        self.__feedback_id = feedback_id
        self.__guest = guest
        self.__rating = rating
        self.__comment = comment

    def submit_feedback(self):
        return f"Feedback {self.__feedback_id} submitted with rating
{self.__rating}"

    def __str__(self):
        return f"Feedback: {self.__rating}/5 - {self.__comment}"
```

Test cases:

Test case 1

```
from datetime import date

# Create guest and room
guest1 = Guest("1", "Sultan Allanjawi", "Sultan@gmail.com", "055-444-444")
room1 = Room("101", "Suite", ["Wi-Fi", "TV", "snacks"], 150.0)

# Create reservation
reservation1 = Reservation("R1", guest1, room1, date(2025, 4, 1),
date(2025, 4, 4))
guest1.add_reservation(reservation1)
```

```
reservation1.confirm_reservation()

print(guest1)
print(room1)
print(reservation1)
```

Output:

Guest: Sultan Allanjawi, Email: Sultan@gmail.com
Room 101 (Suite) - Booked
Reservation R1: Confirmed

Test case 2

```
# Create invoice for the reservation
invoice1 = Invoice("Inv1", reservation1, charges=[20, 30], discounts=25)
total = invoice1.generate_invoice()

# Make payment
payment1 = Payment("P1", total, "Credit Card", date(2025, 3, 27))
payment1.process_payment()

print(invoice1)
print(payment1)
```

Output:

Invoice Inv1 - Total: $475.0
Payment P1 - Paid via Credit Card

Test case 3

```
# Guest submits feedback
feedback1 = Feedback("F1", guest1, 5, "Amazing place, very clean and great
service!")
print(feedback1.submit_feedback())
print(feedback1)
```

```
# Guest requests room service
service_request1 = ServiceRequest("S1", guest1, "Room Cleaning")
service_request1.submit_request()
service_request1.update_status("In Progress")

print(service_request1)
```

Output:

Feedback F1 submitted with rating 5
Feedback: 5/5 - Amazing place, very clean and great service!
ServiceRequest S1 - Room Cleaning [In Progress]

Test case 4

```
# Create a guest and room
guest2 = Guest("2", "Sultan Allanjawi", "Sultan@gmail.com", "055-444-444")
room2 = Room("102", "Double", ["Wi-Fi", "TV"], 120.0)

# Create and confirm a reservation
reservation2 = Reservation("R2", guest2, room2, date(2025, 4, 5),
date(2025, 4, 7))
guest2.add_reservation(reservation2)
reservation2.confirm_reservation()

# Cancel the reservation
reservation2.cancel_reservation()

# Generate a refund invoice (simulate by using a negative invoice)
invoice2 = Invoice("Inv2", reservation2, charges=[], discounts=0)
refunded_amount = invoice2.generate_invoice()  # Should still calculate
value, but treat it as refund

print(reservation2)
print(f"Refund Processed: ${refunded_amount}")
print(room2)  # Room should now be available again
```

Output:

Reservation R2: Cancelled
Refund Processed: $240.0
Room 102 (Double) - Available

Test case 5

```
# Create a guest and two rooms
guest3 = Guest("3", "Sultan Allanjawi", "Sultan@gmail.com", "055-444-444")
room3a = Room("103", "Single", ["Wi-Fi"], 90.0)
room3b = Room("104", "Suite", ["Wi-Fi", "TV", "Mini-bar"], 200.0)

# Create and confirm two reservations
res1 = Reservation("R3", guest3, room3a, date(2025, 5, 1), date(2025, 5,
3))
res2 = Reservation("R4", guest3, room3b, date(2025, 6, 1), date(2025, 6,
4))
res1.confirm_reservation()
res2.confirm_reservation()

guest3.add_reservation(res1)
guest3.add_reservation(res2)

# Display reservation history
print(f"Reservation History for {guest3}")
for r in guest3.view_history():
   print(r)
```

Output:

Reservation History for Guest: Sultan Allanjawi, Email: Sultan@gmail.com
Reservation R3: Confirmed
Reservation R4: Confirmed

Summary of learning:

In this assignment, I learned how to create a UML class diagram and convert it into actual Python code. Through this assignment I practiced implementing object-oriented programming by using classes and objects alongside aggregation and composition relationships. I also discovered how to create test cases and structure code in a clear way.

Github Link:

https://github.com/SultanAllanjawi/Assignment-2-programming-fundamental-