Assignment 1

Saif A. Alhammadi: 202220746

College of Interdisciplinary Sciences

ICS220: Programming Fundamentals

Professor Parkar

30/09/2024

Input:

```
from datetime import datetime

class Reservation:
   def __init__(self, reservationID, checkInDate, checkOutDate, roomType,
status, guestName, guestContact, numberOfNights, totalPrice,
paymentStatus):
       self.__reservationID = reservationID
       self.__checkInDate = datetime.strptime(checkInDate, '%Y-%m-%d')
       self.__checkOutDate = datetime.strptime(checkOutDate, '%Y-%m-%d')
       self.__roomType = roomType
       self.__status = status
       self.__guestName = guestName
       self.__guestContact = guestContact
       self.__numberOfNights = numberOfNights
       self.__totalPrice = totalPrice
       self.__paymentStatus = paymentStatus

   # Setters and Getters
   def set_reservationID(self, reservationID):
       self.__reservationID = reservationID

   def get_reservationID(self):
       return self.__reservationID

   def set_checkInDate(self, checkInDate):
       self.__checkInDate = datetime.strptime(checkInDate, '%Y-%m-%d')

   def get_checkInDate(self):
       return self.__checkInDate

   def set_checkOutDate(self, checkOutDate):
       self.__checkOutDate = datetime.strptime(checkOutDate, '%Y-%m-%d')

   def get_checkOutDate(self):
       return self.__checkOutDate

   def set_roomType(self, roomType):
       self.__roomType = roomType
```

```
    def get_roomType(self):
        return self.__roomType

    def set_status(self, status):
        self.__status = status

    def get_status(self):
        return self.__status

    def set_guestName(self, guestName):
        self.__guestName = guestName

    def get_guestName(self):
        return self.__guestName

    def set_guestContact(self, guestContact):
        self.__guestContact = guestContact

    def get_guestContact(self):
        return self.__guestContact

    def set_numberOfNights(self, numberOfNights):
        self.__numberOfNights = numberOfNights

    def get_numberOfNights(self):
        return self.__numberOfNights

    def set_totalPrice(self, price):
        if price < 0:
            raise ValueError("Price cannot be negative.")
        self.__totalPrice = price

    def get_totalPrice(self):
        return self.__totalPrice

    def set_paymentStatus(self, status):
        self.__paymentStatus = status

    def get_paymentStatus(self):
```

```
        return self.__paymentStatus

    def calculatePrice(self, pricePerNight):
        self.__totalPrice = pricePerNight * self.__numberOfNights

    def __str__(self):
        return f"Reservation for {self.__guestName}, Room:
{self.__roomType}, Total: {self.__totalPrice}, Status: {self.__status}"

# Class: Room
class Room:
    def __init__(self, roomID, roomType, pricePerNight, isAvailable,
maxOccupancy, hasWifi, hasBreakfast, hasAC, hasTV, description):
        self.__roomID = roomID
        self.__roomType = roomType
        self.__pricePerNight = pricePerNight
        self.__isAvailable = isAvailable
        self.__maxOccupancy = maxOccupancy
        self.__hasWifi = hasWifi
        self.__hasBreakfast = hasBreakfast
        self.__hasAC = hasAC
        self.__hasTV = hasTV
        self.__description = description

    # Setters and Getters
    def set_roomID(self, roomID):
        self.__roomID = roomID

    def get_roomID(self):
        return self.__roomID

    def set_roomType(self, roomType):
        self.__roomType = roomType

    def get_roomType(self):
        return self.__roomType

    def set_pricePerNight(self, pricePerNight):
        self.__pricePerNight = pricePerNight
```

```
    def get_pricePerNight(self):
        return self.__pricePerNight

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

    def get_availability(self):
        return self.__isAvailable

    def set_maxOccupancy(self, maxOccupancy):
        self.__maxOccupancy = maxOccupancy

    def get_maxOccupancy(self):
        return self.__maxOccupancy

    def set_hasWifi(self, hasWifi):
        self.__hasWifi = hasWifi

    def get_hasWifi(self):
        return self.__hasWifi

    def set_hasBreakfast(self, hasBreakfast):
        self.__hasBreakfast = hasBreakfast

    def get_hasBreakfast(self):
        return self.__hasBreakfast

    def set_hasAC(self, hasAC):
        self.__hasAC = hasAC

    def get_hasAC(self):
        return self.__hasAC

    def set_hasTV(self, hasTV):
        self.__hasTV = hasTV

    def get_hasTV(self):
        return self.__hasTV

    def set_description(self, description):
```

```
        self.__description = description

    def get_description(self):
        return self.__description

    def __str__(self):
        return f"Room {self.__roomID}, Type: {self.__roomType}, Price:
{self.__pricePerNight}, Available: {self.__isAvailable}"

class Guest:
    def __init__(self, guestID, name, contactNumber, email,
billingAddress):
        self.__guestID = guestID
        self.__name = name
        self.__contactNumber = contactNumber
        self.__email = email
        self.__billingAddress = billingAddress
        self.__reservationList = []  # Initially no reservations

    # Setters and Getters
    def set_guestID(self, guestID):
        self.__guestID = guestID

    def get_guestID(self):
        return self.__guestID

    def set_name(self, name):
        self.__name = name

    def get_name(self):
        return self.__name

    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_contactNumber(self):
        return self.__contactNumber

    def set_email(self, email):
        self.__email = email
```

```
    def get_email(self):
        return self.__email

    def set_billingAddress(self, billingAddress):
        self.__billingAddress = billingAddress

    def get_billingAddress(self):
        return self.__billingAddress

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

    def get_reservations(self):
        return self.__reservationList

    def __str__(self):
        return f"Guest: {self.__name}, Contact: {self.__contactNumber},
Email: {self.__email}, Reservations: {len(self.__reservationList)}"

class Receptionist:
    def __init__(self, receptionistID, name, contactNumber, email):
        self.__receptionistID = receptionistID
        self.__name = name
        self.__contactNumber = contactNumber
        self.__email = email

    # Setters and Getters
    def set_receptionistID(self, receptionistID):
        self.__receptionistID = receptionistID

    def get_receptionistID(self):
        return self.__receptionistID

    def set_name(self, name):
        self.__name = name

    def get_name(self):
        return self.__name
```

```
    def set_contactNumber(self, contactNumber):
        self.__contactNumber = contactNumber

    def get_contactNumber(self):
        return self.__contactNumber

    def set_email(self, email):
        self.__email = email

    def get_email(self):
        return self.__email

    def checkInGuest(self, guest, reservation):
        current_date = datetime.now()
        if reservation.get_status() != "Confirmed":
            print(f"Cannot check in: Reservation is not confirmed for guest
{guest.get_name()}.")
        elif current_date < reservation.get_checkInDate():
            print(f"Cannot check in: Too early for guest
{guest.get_name()}.")
        elif current_date > reservation.get_checkOutDate():
            print(f"Cannot check in: Reservation expired for guest
{guest.get_name()}.")
        else:
            print(f"Guest {guest.get_name()} has been successfully checked
in.")

    def checkOutGuest(self, guest, reservation):
        current_date = datetime.now()
        if current_date < reservation.get_checkInDate():
            print(f"Cannot check out: Guest {guest.get_name()} hasn't
checked in yet.")
        elif current_date > reservation.get_checkOutDate():
            print(f"Guest {guest.get_name()} has already checked out.")
        else:
            print(f"Guest {guest.get_name()} has successfully checked
out.")

    def __str__(self):
```

```
        return f"Receptionist {self.__name}, Contact:
{self.__contactNumber}"

class Payment:
    def __init__(self, paymentID, paymentDate, paymentType, cardNumber,
cardExpiry, cardHolderName, amount, paymentStatus):
        self.__paymentID = paymentID
        self.__paymentDate = datetime.strptime(paymentDate, '%Y-%m-%d')
        self.__paymentType = paymentType
        self.__cardNumber = cardNumber
        self.__cardExpiry = datetime.strptime(cardExpiry, '%m/%y')
        self.__cardHolderName = cardHolderName
        self.__amount = amount
        self.__paymentStatus = paymentStatus

    # Setters and Getters
    def set_paymentID(self, paymentID):
        self.__paymentID = paymentID

    def get_paymentID(self):
        return self.__paymentID

    def set_paymentDate(self, paymentDate):
        self.__paymentDate = datetime.strptime(paymentDate, '%Y-%m-%d')

    def get_paymentDate(self):
        return self.__paymentDate

    def set_paymentType(self, paymentType):
        self.__paymentType = paymentType

    def get_paymentType(self):
        return self.__paymentType

    def set_cardNumber(self, cardNumber):
        self.__cardNumber = cardNumber

    def get_cardNumber(self):
        return self.__cardNumber
```

```
    def set_cardExpiry(self, cardExpiry):
        self.__cardExpiry = datetime.strptime(cardExpiry, '%m/%y')

    def get_cardExpiry(self):
        return self.__cardExpiry

    def set_cardHolderName(self, cardHolderName):
        self.__cardHolderName = cardHolderName

    def get_cardHolderName(self):
        return self.__cardHolderName

    def set_amount(self, amount):
        self.__amount = amount

    def get_amount(self):
        return self.__amount

    def set_paymentStatus(self, status):
        self.__paymentStatus = status

    def get_paymentStatus(self):
        return self.__paymentStatus

    def validateCard(self):
        # Check if the card number is 16 digits and the expiry date is in
the future
        if len(self.__cardNumber) == 16 and self.__cardExpiry >
datetime.now():
            return True
        else:
            return False

    def __str__(self):
        return f"Payment {self.__paymentID}, Amount: {self.__amount},
Status: {self.__paymentStatus}"

class Invoice:
    def __init__(self, invoiceID, reservationID, guestID, issueDate,
totalAmount, taxAmount, paymentMethod, isPaid):
```

```
        self.__invoiceID = invoiceID
        self.__reservationID = reservationID
        self.__guestID = guestID
        self.__issueDate = issueDate
        self.__totalAmount = totalAmount
        self.__taxAmount = taxAmount
        self.__paymentMethod = paymentMethod
        self.__isPaid = isPaid

    # Setters and Getters
    def set_invoiceID(self, invoiceID):
        self.__invoiceID = invoiceID

    def get_invoiceID(self):
        return self.__invoiceID

    def set_reservationID(self, reservationID):
        self.__reservationID = reservationID

    def get_reservationID(self):
        return self.__reservationID

    def set_guestID(self, guestID):
        self.__guestID = guestID

    def get_guestID(self):
        return self.__guestID

    def set_issueDate(self, issueDate):
        self.__issueDate = issueDate

    def get_issueDate(self):
        return self.__issueDate

    def set_totalAmount(self, amount):
        self.__totalAmount = amount

    def get_totalAmount(self):
        return self.__totalAmount
```

```
    def set_taxAmount(self, taxAmount):
        self.__taxAmount = taxAmount

    def get_taxAmount(self):
        return self.__taxAmount

    def set_paymentMethod(self, paymentMethod):
        self.__paymentMethod = paymentMethod

    def get_paymentMethod(self):
        return self.__paymentMethod

    def markAsPaid(self):
        self.__isPaid = True

    def __str__(self):
        return f"Invoice {self.__invoiceID}, Total: {self.__totalAmount},
Paid: {self.__isPaid}"

# Test code
guest1 = Guest(1, 'Saif Alhammadi', '0502228220', '202220746@zu.ac.ae',
'Al Muroor 31 St')
room1 = Room(101, 'Deluxe', 200.00, True, 2, True, True, True, True,
'Spacious room with a view')
reservation1 = Reservation(1001, '2024-12-28', '2025-01-02', 'Deluxe',
'Confirmed', 'Saif Alhammadi', '0502228220', 5, 0, 'Pending')
reservation1.calculatePrice(200)
guest1.add_reservation(reservation1)
payment1 = Payment(501, '2024-12-27', 'Credit Card', '1234567890123456',
'12/25', 'Saif Alhammadi', 1500.00, 'Pending')
if payment1.validateCard():
    payment1.set_paymentStatus('Processed')
invoice1 = Invoice(101, 1001, 1, '2024-12-28', 1500, 100, 'Credit Card',
False)
invoice1.markAsPaid()
receptionist1 = Receptionist(301, 'Zayed', '0508633327',
'202323353@zu.ac.ae')
receptionist1.checkInGuest(guest1, reservation1)
receptionist1.checkOutGuest(guest1, reservation1)
```

```
# Print details
print(guest1)
print(reservation1)
print(payment1)
print(invoice1)
```

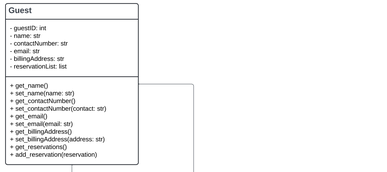
Output:

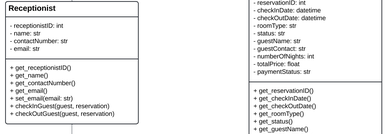
```
Cannot check in: Too early for guest Saif Alhammadi.
Cannot check out: Guest Saif Alhammadi hasn't checked in yet.
Guest: Saif Alhammadi, Contact: 0502228220, Email: 202220746@zu.ac.ae, Reservations: 1
Reservation for Saif Alhammadi, Room: Deluxe, Total: 1000, Status: Confirmed
Payment 501, Amount: 1500.0, Status: Processed
Invoice 101, Total: 1500, Paid: True
```

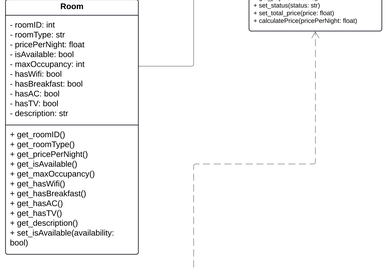
GitHub Repository Link:

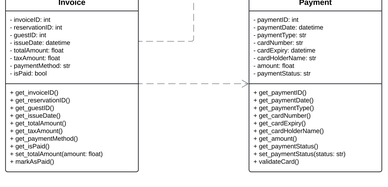
https://github.com/saif-alh/ICS220-Assignment-1---Saif.git

UML Use Cases, Classes, & Descriptions:

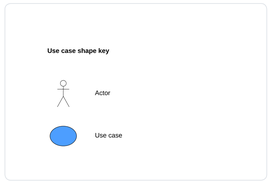




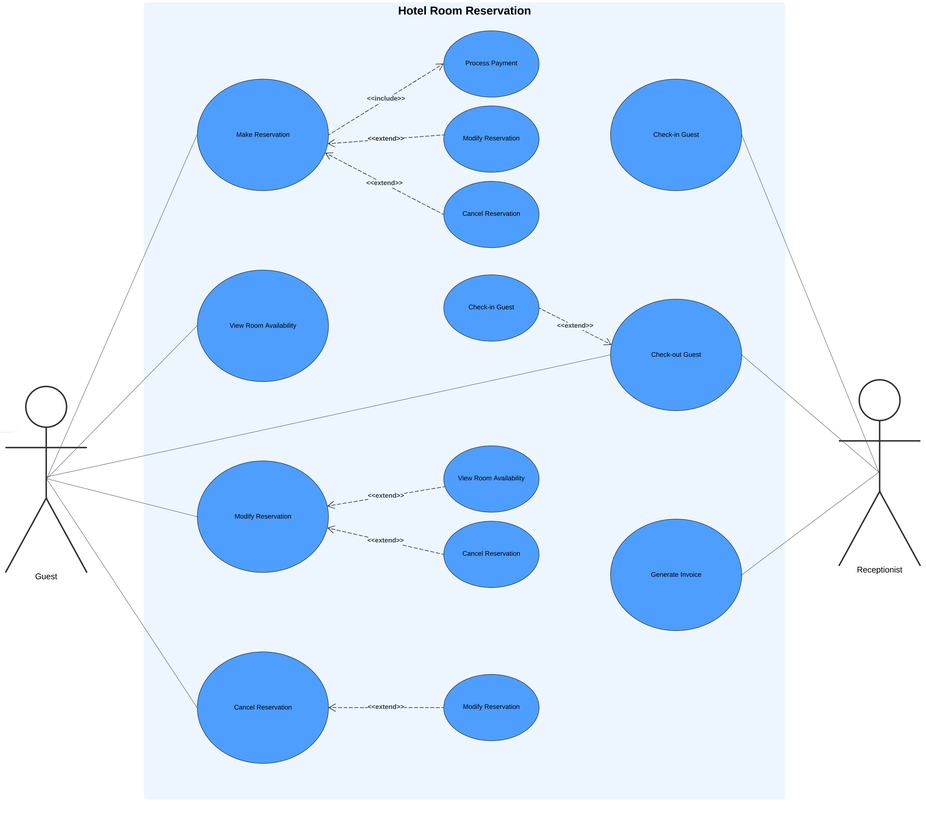




Use Case	Actor(s)	Description	Preconditions	Postconditions	Main Flow
Make Reservation	Guest	The guest makes a reservation online or at the hotel.	Guest is logged in or provides personal details.	Reservation is confirmed, and payment is processed.	1. Guest selects dates. 2. Guest selects room type. 3. Guest enters personal information. 4. Reservation is confirmed.
Cancel Reservation	Guest	Guest cancels a previously made reservation.	A reservation must exist, and it must be within cancellation period.	Reservation is canceled, and any refund processed.	1. Guest selects reservation to cancel. 2. System checks cancellation rules. 3. Reservation is canceled, and guest is notified.
Process Payment	Guest	Payment is processed when a reservation is made.	Reservation exists and is confirmed.	Payment is processed, and the reservation is finalized.	1. System processes the payment. 2. Payment status is updated. 3. Guest receives confirmation.
Check-in Guest	Receptionist	The receptionist checks in a guest upon arrival.	Reservation exists, and the guest has arrived on the check-in date.	Guest is checked in, and room is marked as occupied.	1. Receptionist selects guest. 2. Receptionist confirms reservation details. 3. Guest is checked in.
Check-out Guest	Receptionist	The receptionist checks out a guest.	Guest must have checked in, and today must be the check-out date.	Guest is checked out, room is marked available, and invoice is generated.	1. Receptionist confirms the guest's check-out. 2. Room is marked available. 3. Invoice is generated.
View Room Availability	Guest	The guest views available rooms before making a reservation.	Guest must access the system.	Available rooms are displayed for the selected date range.	1. Guest enters desired dates. 2. System shows available rooms. 3. Guest proceeds to make a reservation.
Generate Invoice	Receptionist	An invoice is generated at the time of check-out.	Guest has checked out, and the stay is complete.	Invoice is generated and paid.	1. Receptionist generates the invoice. 2. System calculates the total. 3. Payment is processed if not already.
Modify Reservation	Guest	The guest modifies a previously made reservation.	A reservation must exist and be within the modification period.	Reservation details are updated, and payment adjustments are made.	1. Guest selects reservation. 2. Guest modifies details. 3. Payment adjustments are made if needed.



Use case shape bank



Summary of Learnings:

LO1: Object-Oriented Analysis and Design (OOAD)

I successfully analyzed the real-world scenario of a hotel reservation system and represented it using UML diagrams. I created a detailed UML class diagram to map entities like Guest, Room, and Reservation along with their attributes and methods. Additionally, I designed a use case diagram to illustrate interactions between the Guest and Receptionist, showcasing relationships like include and extend and clearly outlining the system's functionality.

LO2: Object-Oriented Programming (OOP)

I applied object-oriented programming principles to create Python programs that were well-structured and functional. I used classes and objects to represent different components of the hotel system, ensuring each class had the necessary attributes and methods, along with getter and setter functions. My program was error-free and handled real-world scenarios effectively, such as reservations, payments, and guest check-ins.

LO4: Software Documentation

I ensured my code was well-documented, using clear comments and following coding standards. The structure of my code and the accompanying UML diagrams made it easy to understand how different parts of the system work together. My documentation was tailored to a technical audience, ensuring the design, code, and system functionality were communicated and easy to follow.