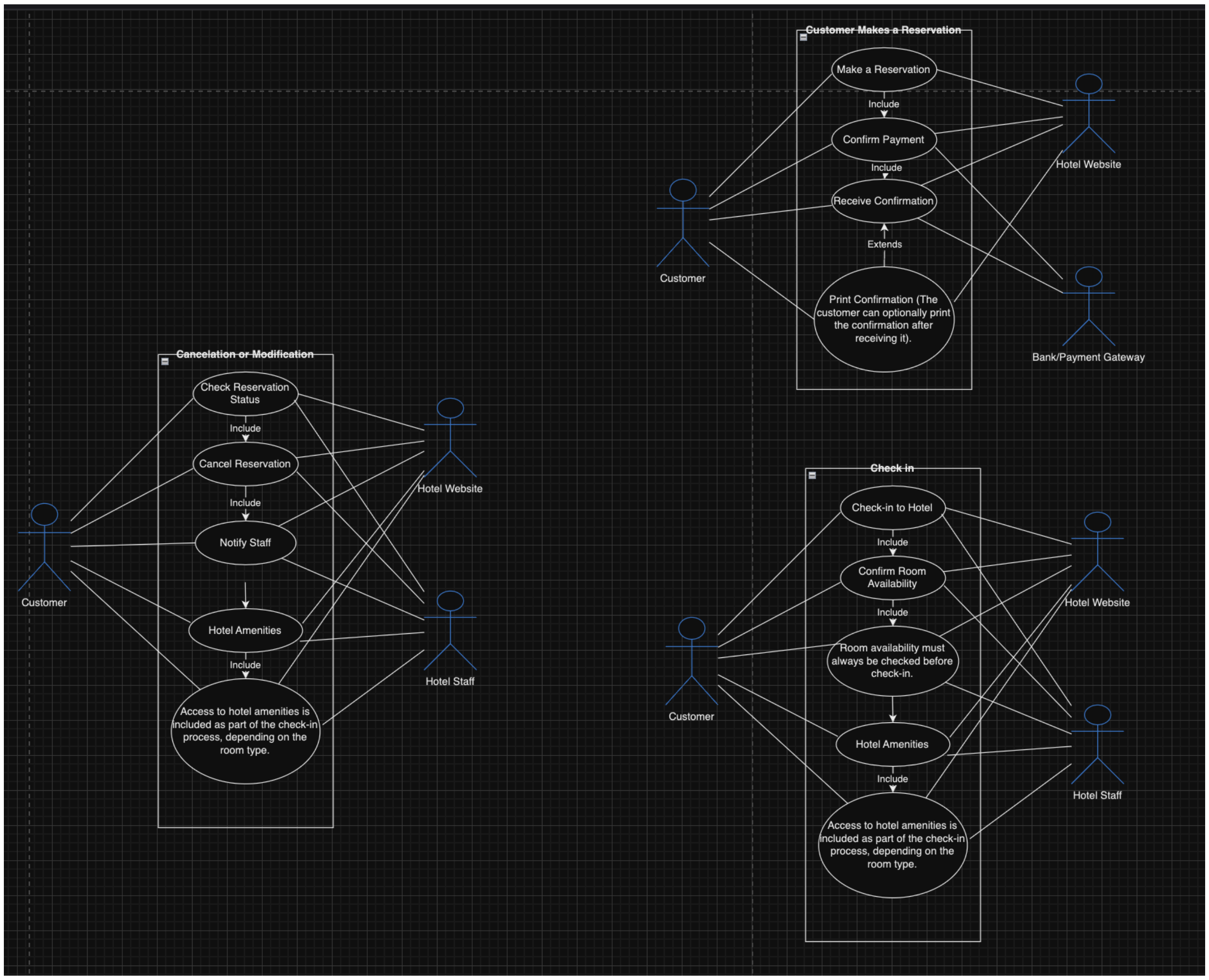
ICS220 - Assignment 1

Mohammad Abdullah Hashim 202107285

- **Identify the software's use cases. Draw the UML use-case diagram and include supporting use-case description tables. At least three scenarios (each with at least two use cases) must be identified. Ensure that the "include" and "extend" relationships are added, where necessary.**


Customer Makes a Reservation
Make a Reservation
Include
Confirm Payment
Include
Receive Confirmation
Extends
Print Confirmation (The customer can optionally print the confirmation after receiving it).
Customer
Hotel Website
Bank/Payment Gateway
Cancelation or Modification
Check Reservation Status
Include
Cancel Reservation
Include
Notify Staff
Hotel Amenities
Include
Access to hotel amenities is included as part of the check-in process, depending on the room type.
Customer
Hotel Website
Hotel Staff
Check in
Check-in to Hotel
Include
Confirm Room Availability
Include
Room availability must always be checked before check-in.
Hotel Amenities
Include
Access to hotel amenities is included as part of the check-in process, depending on the room type.
Customer
Hotel Website
Hotel Staff

Scenario 1: Customer Makes a Reservation

Use case	Make a Reservation
Trigger	The customer selects a hotel and wants to book a room.
Pre-conditions	The customer has access to the hotel website and has selected desired hotel details like check-in date, number of nights, and room type.
Main Scenario	1. Customer selects hotel details. 2. Customer provides personal and payment details. 3. The system checks room availability. 4. The system confirms the reservation and payment details. 5. Customer receives a confirmation of the reservation.
Exceptions	- If the room is unavailable, the system shows an error. - If the payment fails, the reservation is not completed.

Scenario 2: Customer Checks-in to the Hotel

Use case	Check-in to Hotel
Trigger	The customer arrives at the hotel and wants to check in.
Pre-conditions	The customer must have a confirmed reservation in the hotel system.
Main Scenario	1. Customer presents their reservation details to the hotel Receptionist. 2. The receptionist verifies the reservation in the system. 3. The system confirms room availability. 4. The receptionist checks in the customer and provides room access.

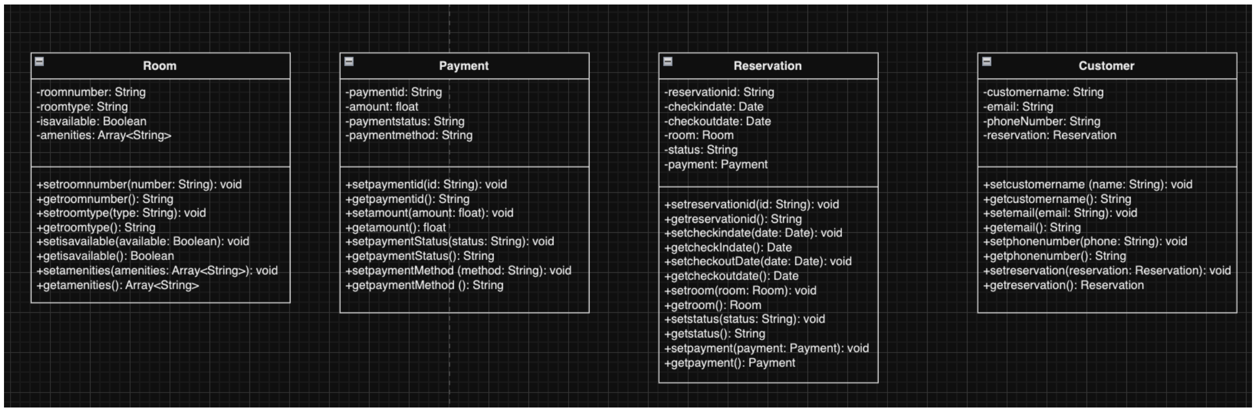
	5. The customer is informed about available amenities.
Exceptions	- If the room is not available, the receptionist offers alternatives. - If the reservation cannot be found, the customer is asked to provide further details or contact support.

Scenario 3: Customer Cancels or Modifies Reservation

Use case	Cancel Reservation
Trigger	The customer decides to cancel a previously made reservation.
Pre-conditions	The customer has a valid reservation that they want to cancel.
Main Scenario	1. Customer logs into the hotel website. 2. Customer selects the option to view their reservation. 3. Customer chooses to cancel the reservation. 4. The system cancels the reservation and notifies the customer. 5. The system sends a cancellation notification to hotel staff. 6. A refund is processed if applicable.
Exceptions	- If the reservation is not eligible for cancellation, the system shows an error message.

	- If the cancellation policy does not allow a refund, the customer is informed.

- **Identify the objects and their respective classes from the use-case descriptions. Draw the UML class diagram and include supporting descriptions to explain the classes identified. At least 4 classes must be identified. Ensure that access specifiers are included for member visibility.**


Room
-roomnumber: String
-roomtype: String
-isavailable: Boolean
-amenities: Array<String>
+setroomnumber(number: String): void
+getroomnumber(): String
+setroomtype(type: String): void
+getroomtype(): String
+setisavailable(available: Boolean): void
+getisavailable(): Boolean
+setamenities(amenities: Array<String>): void
+getamenities(): Array<String>
Payment
-paymentid: String
-amount: float
-paymentstatus: String
-paymentmethod: String
+setpaymentid(id: String): void
+getpaymentid(): String
+setamount(amount: float): void
+getamount(): float
+setpaymentStatus(status: String): void
+getpaymentStatus(): String
+setpaymentMethod (method: String): void
+getpaymentMethod (): String
Reservation
-reservationid: String
-checkindate: Date
-checkoutdate: Date
-room: Room
-status: String
-payment: Payment
+setreservationid(id: String): void
+getreservationid(): String
+setcheckindate(date: Date): void
+getcheckIndate(): Date
+setcheckoutDate(date: Date): void
+getcheckoutdate(): Date
+setroom(room: Room): void
+getroom(): Room
+setstatus(status: String): void
+getstatus(): String
+setpayment(payment: Payment): void
+getpayment(): Payment
Customer
-customername: String
-email: String
-phoneNumber: String
-reservation: Reservation
+setcustomername (name: String): void
+getcustomername(): String
+setemail(email: String): void
+getemail(): String
+setphonenumber(phone: String): void
+getphonenumber(): String
+setreservation(reservation: Reservation): void
+getreservation(): Reservation

1. **Customer**: Represents the customer who makes a reservation and contains their personal information.
2. **Reservation**: Represents the booking details such as check-in and check-out dates, associated room, payment, and status.
3. **Room**: Represents a hotel room, including details like room type, amenities, and availability.
4. **Payment**: Represents the payment details for a reservation, including payment method, amount, and status.

1. Customer Class

Description:
The Customer class stores information related to the customer making the hotel reservation, including their name, contact details, and the reservation object associated with them.
Attributes (Private):

- -customername: String — The name of the customer.
- -email: String — The customer’s email for communication.
- -phoneNumber: String — The customer’s contact number.
- -reservation: Reservation — The reservation associated with the customer.

Methods (Public):

- +setcustomername(name: String): void — Sets the customer’s name.
- +getcustomername(): String — Retrieves the customer’s name.
- +setemail(email: String): void — Sets the customer’s email.
- +getemail(): String — Retrieves the customer’s email.
- +setphonenumber(phone: String): void — Sets the customer’s phone number.
- +getphonenumber(): String — Retrieves the customer’s phone number.
- +setreservation(reservation: Reservation): void — Sets the reservation for the customer.
- +getreservation(): Reservation — Retrieves the reservation for the customer.
- +displayCustomerInfo(): void — Displays all customer-related information.

2. Reservation Class

Description:
The Reservation class represents all details of a customer’s hotel booking, such as the check-in/check-out dates, room details, payment information, and the status of the reservation.
Attributes (Private):

- -reservationid: String — A unique identifier for the reservation.
- -checkindate: Date — The reservation’s check-in date.
- -checkoutdate: Date — The reservation’s check-out date.
- -room: Room — A reference to the associated room.
- -status: String — The status of the reservation (e.g., pending, confirmed, canceled).
- -payment: Payment — A reference to the associated payment.

Methods (Public):

- +setreservationid(id: String): void — Sets the reservation ID.
- +getreservationid(): String — Retrieves the reservation ID.
- +setcheckindate(date: Date): void — Sets the check-in date.
- +getcheckindate(): Date — Retrieves the check-in date.
- +setcheckoutdate(date: Date): void — Sets the check-out date.
- +getcheckoutdate(): Date — Retrieves the check-out date.
- +setroom(room: Room): void — Sets the room for the reservation.
- +getroom(): Room — Retrieves the associated room.
- +setstatus(status: String): void — Sets the reservation status.
- +getstatus(): String — Retrieves the reservation status.

- +setpayment(payment: Payment): void — Sets the payment details for the reservation.
- +getpayment(): Payment — Retrieves the associated payment.
- +displayReservationInfo(): void — Displays all reservation-related information.

3. Room Class

Description:
The Room class holds information about the hotel room being booked, including its number, type, availability, and amenities.
Attributes (Private):

- -roomnumber: String — The room's number or description.
- -roomtype: String — The type of room (e.g., standard, deluxe, suite).
- -isavailable: Boolean — The availability status of the room.
- -amenities: Array<String> — A list of amenities available in the room.

Methods (Public):

- +setroomnumber(number: String): void — Sets the room number.
- +getroomnumber(): String — Retrieves the room number.
- +setroomtype(type: String): void — Sets the room type.
- +getroomtype(): String — Retrieves the room type.
- +setisavailable(available: Boolean): void — Sets the room availability.
- +getisavailable(): Boolean — Retrieves the room availability.
- +setamenities(amenities: Array<String>): void — Sets the amenities for the room.
- +getamenities(): Array<String> — Retrieves the list of amenities.
- +displayRoomInfo(): void — Displays all room-related information.

4. Payment Class

Description:
The Payment class stores payment information for the reservation, including the amount, payment method, and status.
Attributes (Private):

- -paymentid: String — A unique identifier for the payment.
- -amount: float — The total amount paid for the reservation.
- -paymentstatus: String — The current status of the payment (e.g., processed, pending, refunded).
- -paymentmethod: String — The method of payment (e.g., credit card, PayPal).

Methods (Public):

- +setpaymentid(id: String): void — Sets the payment ID.
- +getpaymentid(): String — Retrieves the payment ID.
- +setamount(amount: float): void — Sets the payment amount.
- +getamount(): float — Retrieves the payment amount.
- +setpaymentStatus(status: String): void — Sets the payment status.
- +getpaymentStatus(): String — Retrieves the payment status.
- +setpaymentMethod(method: String): void — Sets the payment method.
- +getpaymentMethod(): String — Retrieves the payment method.
- +displayPaymentInfo(): void — Displays all payment-related information.

- **Create Python classes with the constructor, attributes (at least 5), and required setter/getter methods for all the identified classes. Identify and include other required function-headers in the classes where the function's body is just a pass statement and include a comment to indicate what the function should achieve.**

```
## Room Class
class Room:
  """Class to represent a room"""
  # Constructor/initializer
  def __init__(self, roomnumber, roomtype, isavailable=True, amenities=None):
      if amenities is None:
          amenities = []
      self.__roomnumber = roomnumber
      self.__roomtype = roomtype
      self.__isavailable = isavailable
      self.__amenities = amenities
  # Setter and Getter methods
  def setroomnumber(self, number):
      self.__roomnumber = number
  def getroomnumber(self):
      return self.__roomnumber
  def setroomtype(self, type):
      self.__roomtype = type
  def getroomtype(self):
      return self.__roomtype
  def setisavailable(self, available):
      self.__isavailable = available
  def getisavailable(self):
      return self.__isavailable
  def setamenities(self, amenities):
      self.__amenities = amenities
  def getamenities(self):
      return self.__amenities
  # Method to display room information
  def displayRoomInfo(self):
      availability = "Available" if self.__isavailable else "Not Available"
      print(f"Room Number: {self.__roomnumber}")
      print(f"Room Type: {self.__roomtype}")
      print(f"Availability: {availability}")
      print(f"Amenities: {', '.join(self.__amenities)}")
  # Destructor
  def __del__(self):
      print(f"The room {self.__roomnumber} was deleted.")

# Payment Class
class Payment:
  """Class to represent payment information"""
  # Constructor/initializer
  def __init__(self, paymentid, amount, paymentstatus='Pending', paymentmethod='Credit Card'):
      self.__paymentid = paymentid
      self.__amount = amount
      self.__paymentstatus = paymentstatus
      self.__paymentmethod = paymentmethod
  # Setter and Getter methods
  def setpaymentid(self, id):
      self.__paymentid = id
  def getpaymentid(self):
      return self.__paymentid
  def setamount(self, amount):
      self.__amount = amount
  def getamount(self):
      return self.__amount
  def setpaymentStatus(self, status):
      self.__paymentstatus = status
  def getpaymentStatus(self):
      return self.__paymentstatus
  def setpaymentMethod(self, method):
      self.__paymentmethod = method
  def getpaymentMethod(self):
      return self.__paymentmethod
  # Method to display payment information
  def displayPaymentInfo(self):
      print(f"Payment ID: {self.__paymentid}")
      print(f"Amount: ${self.__amount}")
      print(f"Status: {self.__paymentstatus}")
      print(f"Method: {self.__paymentmethod}")
  # Destructor
  def __del__(self):
      print(f"The payment record {self.__paymentid} was deleted.")

# Reservation Class
class Reservation:
  """Class to represent a reservation"""
  # Constructor/initializer
  def __init__(self, reservationid, checkindate, checkoutdate, room, status='Pending', payment=None):
      self.__reservationid = reservationid
      self.__checkindate = checkindate
      self.__checkoutdate = checkoutdate
      self.__room = room  # Room object
      self.__status = status
      self.__payment = payment  # Payment object
  # Setter and Getter methods
  def setreservationid(self, id):
      self.__reservationid = id
  def getreservationid(self):
      return self.__reservationid
  def setcheckindate(self, date):
      self.__checkindate = date
  def getcheckindate(self):
      return self.__checkindate
  def setcheckoutdate(self, date):
      self.__checkoutdate = date
  def getcheckoutdate(self):
      return self.__checkoutdate
  def setroom(self, room):
      self.__room = room
  def getroom(self):
      return self.__room
  def setstatus(self, status):
      self.__status = status
  def getstatus(self):
      return self.__status
  def setpayment(self, payment):
      self.__payment = payment
  def getpayment(self):
      return self.__payment
  # Method to display reservation information
  def displayReservationInfo(self):
      print(f"Reservation ID: {self.__reservationid}")
      print(f"Status: {self.__status}")
      print(f"Check-in Date: {self.__checkindate}")
      print(f"Check-out Date: {self.__checkoutdate}")
      if self.__room:
          self.__room.displayRoomInfo()
      if self.__payment:
          self.__payment.displayPaymentInfo()
  # Destructor
  def __del__(self):
      print(f"The reservation {self.__reservationid} was deleted.")

# Customer Class
class Customer:
  """Class to represent a customer"""
  # Constructor/initializer
  def __init__(self, customername, email, phonenumber, reservation=None):
      self.__customername = customername
      self.__email = email
      self.__phonenumber = phonenumber
      self.__reservation = reservation  # Reservation object
  # Setter and Getter methods
  def setcustomername(self, name):
      self.__customername = name
  def getcustomername(self):
      return self.__customername
  def setemail(self, email):
      self.__email = email
  def getemail(self):
      return self.__email
  def setphonenumber(self, phone):
      self.__phonenumber = phone
  def getphonenumber(self):
      return self.__phonenumber
  def setreservation(self, reservation):
      self.__reservation = reservation
  def getreservation(self):
      return self.__reservation
  # Method to display customer information
  def displayCustomerInfo(self):
      print(f"Customer Name: {self.__customername}")
      print(f"Email: {self.__email}")
      print(f"Phone Number: {self.__phonenumber}")
      if self.__reservation:
          self.__reservation.displayReservationInfo()
  # Destructor
  def __del__(self):
      print(f"The customer record for {self.__customername} was deleted.")
```

- **Create objects of all the identified classes and use the object's functions to populate and display all the information shown in Figure 1**

```
# 1. Create Room Object
```

```
room = Room(roomnumber="2 Queen Beds", roomtype="No Smoking/Desk/Safe/Coffee Maker In
Room/Hair Dryer", isavailable=False, amenities=["Desk", "Safe", "Coffee Maker", "Hair Dryer"])
# 2. Create Payment Object
payment = Payment(paymentid="Mastercard (ending in 9904)", amount=201.48,
paymentstatus="Processed", paymentmethod="Mastercard")
# 3. Create Reservation Object
reservation = Reservation(reservationid="52523687", checkindate="Sun, Aug 22, 2010 - 03:00 PM",
checkoutdate="Tue, Aug 24, 2010 - 12:00 PM", room=room, status="Confirmed", payment=payment)
# 4. Create Customer Object
customer = Customer(customername="Ted H Vera", email="tedvera@mac.com",
phonenumber="505-661-1110", reservation=reservation)
# Display information using the object's methods
# Customer Info
print("Customer Information:")
customer.displayCustomerInfo()
# Reservation Info
print("\nReservation Information:")
reservation.displayReservationInfo()
# Room Info
print("\nRoom Information:")
room.displayRoomInfo()
# Payment Info
print("\nPayment Information:")
payment.displayPaymentInfo()
```