

Student Name

Roll No

Section

Student Signature

Vetted by

Vetter Signature

### Question Q1

Weightage:15; Marks: 30

You're tasked with developing a tour management system for a travel agency offering various types of tours: adventure, cultural, and wildlife. Each event type has unique ticket pricing scenario.

- a) Design an abstract base class named "Tour" with common tour attributes such as name, duration, and destination. Include abstract methods calculateTicketPrice() and scheduleTour() to accommodate different pricing and scheduling logic for each tour type.
- b) Create derived classes for each tour type: "AdventureTour", "CulturalTour", and "WildlifeTour". Override the abstract methods in each derived class to implement specific pricing and scheduling logic. For adventure tours, pricing may be based on activities and tour duration. Cultural tours might have pricing based on the number of sites visited and additional experiences offered. Wildlife tours could have pricing based on the rarity of sightings and guide expertise.
- c) Utilize dynamic dispatch to determine the appropriate version of the methods at runtime based on the actual object type.
- d) Populate the array with instances of derived classes representing different tour types, such as AdventureTour, CulturalTour, and WildlifeTour.
- e) Demonstrate the functionality by creating instances of different tour types and calling the calculateTicketPrice() and scheduleTour() methods for each tour.

## Question Q2

Weightage: 15; Marks: 30

Your task is to develop a Ticket Reservation System for a theater where users can view available seats, reserve seats, and cancel reservations.

**Create Seat Class:** The Seat class represents individual seats in the theater.. Member variables include row, seatNumber, and reserved. Methods include: reserve(): Marks the seat as reserved. cancelReservation(): Marks the seat as available again. getStatus(): Retrieves the status of the seat (available or reserved).

Now, in the 'Theater' class, manage the theater's seating arrangement. Dynamically handle a collection of 'Seat' objects representing all seats in the theater. Implement methods such as 'initializeSeats(int numRows, int numSeatsPerRow)' to set up the seating arrangement, 'viewAvailableSeats()' to display a seating chart showing available and reserved seats, 'reserve(int row, int seat)' to allow a user to reserve a specific seat, and 'cancelReservation(int row, int seat)' to enable a user to cancel a reservation for a specific seat. Additionally, the 'Theater' class should include a private helper method 'isValidSeat(int row, int seat)' to validate whether the specified seat exists.

**Note:** As you design the system, consider how the 'Theater' class can interact with the 'Seat' objects to manage seat reservations effectively. Think about encapsulation and data integrity: how can you ensure that only the 'Theater' class has the necessary access to modify the reservation status of 'Seat' objects? Consider the principles of object-oriented design and abstraction in your solution.

Ensure proper functionality and exception handling, such as attempting to reserve or cancel a reservation for a nonexistent seat, trying to reserve an already reserved seat, or attempting to cancel a reservation for an unreserved seat. Keep in mind that you need to maintain the integrity of the system while achieving the desired functionality. How can you achieve this through the design of your classes and their interactions?

**Main Function:**

In the main() function, an instance of the Theater class is created.

The seating arrangement is initialized with the desired number of rows and seats per row.

Users can view available seats, reserve seats, and cancel reservations through method calls.

The seating chart is displayed after each operation to reflect the changes.

**Note:** Apart from above mentioned functions, you can create more helper functions if required.

**The system must provide a user-friendly interface for managing seat reservations in a theater, ensuring proper management of seats and handling of user requests with exception handling mechanisms.**



**Question Q3****Weightage: 20; Marks: 40**

You're tasked with developing a flight reservation system for a global airline company that offers various types of flights, including standard, premium, and luxury classes.

- a) Begin by designing a base class called "Flight" that encompasses common functionality shared by all flights, such as flight number, departure and arrival airports, departure and arrival times, and airline information. Additionally, create three derived classes, namely "StandardFlight," "PremiumFlight," and "LuxuryFlight," which inherit from the "Flight" class. Each derived class should include attributes specific to the type of flight, such as seating configurations, in-flight amenities, and fare types.
- b) Ensure that each derived class includes unique attributes relevant to the type of flight, such as the availability of gourmet meals, spacious seating arrangements, and exclusive lounge access.
- c) Develop a generic booking system capable of handling reservations for all types of flights using templates. This system should streamline the booking process and ensure consistency across all flight bookings, regardless of the chosen class or destination.
- d) Enhance the booking system by implementing operator overloading for tax addition. Develop three operator overloading functions within the "Flight" class to accommodate different tax rates for flight bookings. The first operator overloading function should add a 50% tax rate to the total booking amount, suitable for routes or destinations subject to higher taxes or fees. The second operator overloading function should apply a 20% tax rate, catering to routes or destinations with moderate tax rates. Additionally, design a third operator overloading function to return both amounts after applying respective taxes, providing passengers with transparent pricing and empowering them to make informed decisions.