



Object Oriented Programming (OOP)

Course Project: Travel Agency System

Background

You are tasked with designing and implementing a travel agency system for a tourism company. The system should allow customers to browse available tours and destinations, make bookings, and cancel bookings. The system should also allow administrators to manage tours, destinations, and bookings.

Requirements

The system should be implemented in C# using object-oriented programming principles. The system should consist of the following classes:

TravelEntity Class

This is a base class for all travel-related entities, including tours and destinations. It should have the following properties:

- **Name** (string): The name of the travel entity.
- **Description** (string): A description of the travel entity.

Tour Class

This class represents a tour offered by the travel agency. It should inherit from the **TravelEntity** class and have the following additional properties:

- **Duration** (int): The duration of the tour in days.
- **Destinations** (List of **Destination** objects): A list of destinations that the tour will visit.
- **Price** (float): The price of the tour per person.
- **Bookings** (List of **Booking** objects): A list of bookings for the tour.



The **Tour** class should have the following methods:

- **AddDestination** (**Destination** object): Adds a destination to the tour's list of destinations.
- **RemoveDestination** (**Destination** object): Removes a destination from the tour's list of destinations.
- **GetAvailableSeats**: Returns the number of available seats for the tour.
- **MakeBooking** (**Customer** object, **int** numberOfSeats): Creates a booking for the tour.
- **CancelBooking** (**Booking** object): Cancels a booking for the tour.
- **GetBookings**: Returns a list of bookings for the tour.
- **GetDestinations**: Returns a list of destinations for the tour.
- **GetPrice**: Returns the price of the tour per person.

Destination Class

This class represents a destination that the travel agency offers tours to. It should inherit from the **TravelEntity** class and have the following additional properties:

- **Seats** (**int**): The number of available seats for the destination.

Booking Class


This class represents a booking made by a customer for a tour. It should have the following properties:

- **Customer** (**Customer** object): The customer who made the booking.
- **Tour** (**Tour** object): The tour that was booked.
- **NumberOfSeats** (**int**): The number of seats booked for the tour.

Customer Class

This class represents a customer who can make bookings for tours. It should have the following properties:

- **Name** (**string**): The name of the customer.

- 
- **Email** (string): The email address of the customer.

Implementation Guidelines

- Implement the **TravelEntity**, **Tour**, **Destination**, **Booking**, and **Customer** classes in C#.
- Use inheritance to implement the **Tour** and **Destination** classes as derived classes of the **TravelEntity** class.
- Use encapsulation to protect the data stored in the classes from being accessed or modified directly.
- Implement the methods in the **Tour**, **Destination**, and **Booking** classes as described in the requirements.
- Write a program that demonstrates the functionality of the system.

Sample for Main Class and Expected output.

```
static void Main(string[] args)
{
    // Create a new destination in Egypt
    Destination destination = new Destination();
    destination.Name = "Luxor";
    destination.Description = "Explore the ancient wonders of Luxor, Egypt";
    destination.Seats = 50;

    // Create a new tour
    Tour tour = new Tour();
    tour.Name = "Luxor Adventure";
    tour.Description = "Experience the magic of Luxor and its historical sites";
    tour.Duration = 5;
    tour.Price = 1500.00f;

    // Add the destination to the tour
    tour.AddDestination(destination);

    // Create a new customer
    Customer customer = new Customer();
    customer.Name = "Yasser Mohammed";
    customer.Email = "yasser.mohammed@example.com";

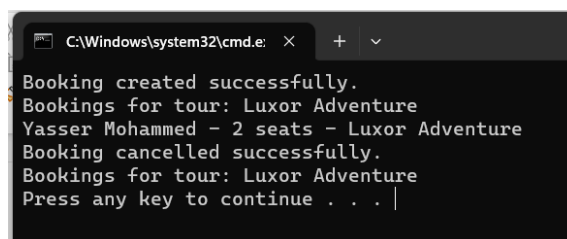
    // Make a booking for the customer
    tour.MakeBooking(customer, 2);

    // Get the list of bookings for the tour
    List<Booking> bookings = tour.GetBookings();

    // Print out the booking information
    Console.WriteLine("Bookings for tour: {0}", tour.Name);
    foreach (Booking booking in bookings)
    {
        Console.WriteLine("{0} - {1} seats - {2}", booking.Customer.Name, booking.NumberOfSeats, booking.Tour.Name);
    }

    // Cancel the booking
    Booking bookingToCancel = bookings[0];
    tour.CancelBooking(bookingToCancel);

    // Print out the updated booking information
    bookings = tour.GetBookings();
    Console.WriteLine("Bookings for tour: {0}", tour.Name);
    foreach (Booking booking in bookings)
    {
        Console.WriteLine("{0} - {1} seats - {2}", booking.Customer.Name, booking.NumberOfSeats, booking.Tour.Name);
    }
}
```



```
C:\Windows\system32\cmd.e: X + v
Booking created successfully.
Bookings for tour: Luxor Adventure
Yasser Mohammed - 2 seats - Luxor Adventure
Booking cancelled successfully.
Bookings for tour: Luxor Adventure
Press any key to continue . . . |
```

In this example, we establish a new destination in Egypt's Luxor, famed for its ancient wonders. A tour named "Luxor Adventure" is introduced, offering a 5-day exploration of Luxor's historical sites at a cost of 1500 Egyptian pounds. The Luxor destination is then associated with the tour. A customer named Yasser Mohammed is created, and a booking is made for him to partake in the Luxor Adventure tour for 2 individuals. The program showcases the booking specifics: "Luxor Adventure" tour, Yasser Mohammed as the participant, and a reservation for 2 seats. The demonstration also includes a cancellation of Yasser Mohammed's booking. As a result, the program displays the revised list of bookings for the "Luxor Adventure" tour. This example illustrates the system's capacity to manage customer bookings, tour details, and destination information within the travel agency context.