Team Name: The OOPtimizers

Online Ticket Management System

Name	Task	ID
Utso Chandro Roy	User Class	2233081517
Md. Tamal Ali	Event Class	2231081100
Md. Salauddin	Ticket Class	2233081499
Md. Abdul Haque Huzaifa	Admin Class	2233081500
Shahag Raihan	TicketStats Class	2233081508

##User Class

Summary of the code : -

The code defines a `User` class in C# to represent user information with properties and methods for managing user data. Below is a breakdown of the functionality:

1.Properties
2.Constructors
3.Methods

1.Properties

`1.UserName`: Stores the user's name (public).
`2.Email`: Stores the user's email (public).
`3.Password`: Stores the user's password (get and set).

2. Constructors

1.**Default Constructor** :
Initializes the user with default values . Example,

`UserName`: "Default"
`Email`: "unknown@gmail.com"
`Password`: "12345678"

2.Parameterized Constructor

Allows registering a new user with specific `UserName`, `Email`, and `Password`.

3. Copy Constructor (`User(User other)`)
Creates a deep copy of another `User` object.

3.Methods

1. UserInfo()

   Displays the user's information (name, email, password) in a formatted output.

2. `DeleteUser()`:

   Clears the user’s information (sets `UserName`, `Email`, and `Password` to empty strings) and prints a confirmation message.

Slide 1: Introduction to the Event Class

1. **Topic:**
    - The `Event` class manages events for which tickets are available.
    - **Features:** Add, update, and remove events.
2. **Class Components:**
    - Created a class named `Event`.
    - Defined **3 public fields**:
        - `EventName`: Stores the name of the event.
        - `EventDate`: Stores the date of the event.
        - `EventPlace`: Stores the location of the event.
3. **Static Variable:**
    - `Eventcnt`: Tracks the total number of events.
4. **Constructor:**
    - Accepts three parameters: `name`, `date`, and `place`.
    - Initializes the fields and increments the event count (`Eventcnt`).
5. **Methods for Event Management:**
    - **UpdateEvent (1 parameter):** Updates the event name.
    - **UpdateEvent (2 parameters):** Updates the event date and place.
    - **UpdateEvent (3 parameters):** Updates the event name, date, and place. (Method Overloading)
    - **EventInfo:** Displays the event details.
    - **ShowEventCnt:** Displays the total number of events (static method).
    - **RemoveEvent:** Decrements the event count (static method).

Slide 2: Main Method Overview

1. **Create Events:**
    - Created two event objects using the parameterized constructor.
    - Displayed their details using the `EventInfo` method.
2. **Update Event:**
    - Used the `UpdateEvent` method to update the name, date, and place of an event.
    - Displayed the updated event details.
3. **Remove an Event:**
    - Removed one event using the `RemoveEvent` method.
    - Displayed the updated event count using `ShowEventCnt`.

Methods

`TicketInfo()`:

- Displays detailed information about the ticket, including:
    1. Event name
    2. Ticket ID
    3. Seat number
    4. Price

`Ticket` Class##

Summary of the code : -

This code defines a `Ticket` class that models the details of a ticket for an event, supports displaying ticket information, and includes operator overloading for price addition.

Class Members : -

1. `eventName`: Stores the name of the event. (`string`)
2. `ticketID`: Stores a unique identifier for the ticket. (`string`)
3. `seat number` : Stores the seat number associated with the ticket.(`int`)
4. `price` : Stores the ticket price.(`double`)

Constructor : -

`Ticket(string EventName, string TicketId, int SeatNumber, double Price)`:

- Initializes a `Ticket` object with:

  1.`EventName` (event name)

  2.`TicketId` (ticket ID)

  3.`SeatNumber` (seat number)

  4.`Price` (ticket price)

Operator Overloading : -

Allows adding two `Ticket` objects by summing their prices.

Admin class

Summary of the code : -
This code defines a system for managing ticket booking, cancellation, and event searching in a ticket management system, with a focus on inheritance and interface implementation.

Key components :-

1. **Interface `FindEvent`:**

- Declares the method `SearchEvent(string Eventname)`.
- This method is used for searching events by name.

2. **Class `Admin` :**
   Inherits from :

- `TicketManagement` (not defined in the provided code, assumed to manage ticket-related operations).
- Implements `FindEvent` interface.

2. Methods in `Admin` Class:

1. **`BookTicket(string EventName, string TicketId, int SeatNumber, double Price)` (Overridden):**
   - Creates a new ticket using the provided details.
   - Displays a success message and ticket information.
   - Updates ticket sales statistics and generates an invoice using the `TicketStats` class (assumed external class).

`2. SearchEvent(string Eventname)` (Interface Implementation):

- Searches for an event name in a predefined list.
- Prints a success message if the event is found; otherwise, displays a "not found" message.

1. **Inheritance:**
   The `Admin` class overrides `BookTicket` and `CancelBooking` from its parent class `TicketManagement`.

2. **Interface Implementation:**
   The `Admin` class implements the `SearchEvent` method from the `FindEvent` interface.

3. **Event Management:**
   Predefined events (`MMA`, `BeatBoxing`, etc.) are searched using the `SearchEvent` method.

TicketStats class##

Summary of the code : -
This code defines a ticketStates class in C# that serves as a utility for tracking ticket sales and generating invoices.

Class Overview : -

Class Type: `static`

- All members and methods in the class are `static`.
- Cannot be instantiated; members are accessed directly using the class name.

1. **Static Members**

`1.totalTicketSold`: Tracks the total number of tickets sold (int type).

`2.totalRevenue`: Tracks the total revenue from all ticket sales (double type).

2. Methods:

`Stats(double price)`:

- Increments the `totalTicketSold` counter.
- Adds the given `price` to `totalRevenue`.

`StatsInfo(Ticket ticket)`:

- Calculates the total cost of a specific ticket using `ticket.Cost()`.

  Displays a detailed invoice:

- Ticket information.
- Total tickets sold.

- Total cost for the current ticket.
- Cumulative total revenue.