Team: noobCoders

Name	Id
Tonusree Rani Datta	2243081044
Jannatul Naim Jerin	2243081051
Naznin Sultana	2243081043
Md. Mahmudul Hasan	2243081079
Md. Mojahid	2241081394

Static Class: ToLogin

Purpose: Protect system access using predefined credentials.

Properties

- **userPhone**
- **userPassword**

Static Method:

- **Login()**
  - Asks for number & password
  - Validates and returns true/false

Interfaces and the Train Class

Interfaces: TrainA, TrainB

- Display(string message): For showing general messages.
- DisplayTrain(): For presenting specific train details.

These interfaces enforce a contract for how train information is displayed, promoting consistency.

Train Class:

Purpose: Represents a train with route, seat info, and message display features.

- **Properties:**TrainNumber, TrainName, From, To, TotalSeat, AvailableSeat
- **Constructors:**
- Default
- Main
- Copy

Methods and Operator Overloading in Train Class

Implemented Methods

- `Display()`

  Outputs a general message related to the train status or operations.

- `DisplayTrain()`

  Presents a detailed overview of the train's information, including route, capacity, and availability.

Operator Overloading

- `++` Operator

  `Increments AvailableSeat`

- `--` Operator

  `Decrements AvailableSeat`

Ticket System: Base and Derived Classes

Class: TicketBase

Purpose: Acts as a parent class for common ticket-related properties.

- **Properties:** TicketId, PassengerName, TrainNumber, Class, Cancelled
- Provides virtual DisplayTicket()

Class: Ticket

Purpose: Represents a real booked ticket in the system.

- **Inheritance:**TicketBase.
- **Override** DisplayTicket()**:** Customizes the display logic.
- **Conditional Printing:**Cancelled is false, ensuring users only see active bookings.

The Ticket class demonstrates polymorphism and focused functionality.

Abstract Reservation System Design

1

Abstract Class: TrainReservationSystemBase

Abstract Methods:

Purpose: Defines the essential functionalities of a reservation system.

- **ViewTrains()**: To display all available trains.
- **BookTicket()**: Handles the process of reserving a seat.
- **ViewBooked()**: Shows all currently booked tickets.
- **CancelTicket()**: Manages the cancellation of a reservation.

These abstract methods compel derived classes to provide concrete implementations for each critical function.

2

TrainReservationSystem

Purpose: Implements the entire reservation operation using lists of trains & tickets.

- **Override** ViewTrains()**:** Displaying the list of all available trains
- **Override** BookTicket()**:** This overridden method provides the complete ticket booking process
- **Override** ViewBooked()**:** This method shows all booked tickets
- **Override** CancelTicket()**:** This method performs the ticket cancellation steps

This concrete class brings the abstract design to life, handling the full spectrum of reservation processes.

Fare Calculation & Booking Workflow

The system provides a clear and interactive booking process, including dynamic fare calculation based on route and class, guiding the user from train selection to ticket confirmation.

Fare Calculation (Overloading)

- **Base Fare:(**From) and destination (**To**) of the journey.
- **Class-Based Additions:**
  - AC (Air-Conditioned)
  - F_SEAT (First Class Seating)
  - S_CHAIR (Shovon Chair)
  - SHOVAN (Standard Seating)

Method overloading allows for flexible fare calculation logic.

Booking Flow

01

User enters desired route.

02

User specifies the travel date.

03

System displays available trains.

04

User selects preferred travel class.

05

Calculated fare is presented to the user.

06

Ticket is created, and available seats are updated.

User Interaction & Feedback

Sample Output and System Responses

The console-based system provides clear and immediate feedback to the user throughout the reservation process, confirming actions and displaying relevant information.

"Login Successful!"

Confirms successful authentication, granting access to the system's features.

Train List Display

Presents a clear, formatted list of available trains, including their routes and seat availability.

Fare Presentation

Clearly shows the calculated fare for the selected journey and class before final confirmation.

"Ticket booked successfully!"

Confirms that the reservation has been completed and the ticket is issued.

"Ticket cancelled successfully!"

Verifies that the requested ticket cancellation has been processed.

Key OOP Concepts Demonstrated

This reservation system serves as an excellent practical example of how fundamental OOP principles can be effectively applied to build robust and maintainable software.

Encapsulation

Bundling data and methods that operate on the data within a single unit (e.g., Train, Ticket classes).

Inheritance

Ticket class inheriting properties and methods from TicketBase, promoting code reuse.

Polymorphism

Method and operator overloading (e.g., DisplayTicket(), ++/-- operators) adapting behavior based on context.

Abstraction

Using interfaces (TrainA, TrainB) and abstract classes (TrainReservationSystemBase) to define blueprints for functionality.

Conclusion

This system demonstrates strong C# OOP concepts with real-world reservation functionality, including interfaces, inheritance, abstract classes, overloading, constructors, and structured system flow.


Thank
you!