Public Transportation Station Management System

Object-Oriented Analysis (OOA)

During the object-oriented analysis for the Public Transportation Station Management System, I identified several main objects. The key entities include: Vehicle (representing buses or trains), ExpressBus**,** Station**,** Schedule (timetables for buses or trains), Schedule_ExpressBus (timetables for express buses), Passenger, and Ticket.

Each object has its own attributes describing its specific information. The Vehicle class stores the route, capacity, and operating status. ExpressBus inherits all these attributes and additionally introduces the speed attribute to represent travel speed, enabling the system to calculate shorter travel times. For Station, important attributes include the station name, location, type, a list of schedules, and a dedicated schedule for express buses. Schedule and Schedule_ExpressBus hold data about routes and arrival times. Passenger maintains personal information, account balance, and a list of booked tickets. Finally, Ticket stores information such as route name, vehicle type, departure and destination stops, departure time, arrival time, and total travel time.

Regarding methods, all classes include getter/setter functions to access or update data. Vehicle provides the calculateTravelTime() method to estimate travel duration. ExpressBus overrides this method to compute the duration based on its speed attribute, which is higher than that of a regular bus (class Vehicle). Station supports adding new routes via the addRoute() method. Besides data-handling operations, Passenger can deposit funds, display personal information, book or cancel tickets. Ticket provides the displayTicketInfo() method to show detailed ticket information.

The inheritance relationship is clearly defined between ExpressBus and Vehicle. The ExpressBus class reuses common components such as route, capacity, and status, while extending them with its own speed attribute and overriding the calculateTravelTime() method to reflect the faster speed of an express bus. This design reduces code duplication, improves maintainability, and facilitates the future expansion of the system to include additional types of vehicles.

Explanation about class design in my code

In the class design, the system is organized into well-defined components, with each class handling a distinct set of functions to ensure the code is readable and maintainable. The Vehicle class serves as the base class, managing general information about vehicles, such as route name, passenger capacity, and operational status. ExpressBus is constructed as a specialized type of vehicle, focusing on providing detailed information about faster operations compared to regular buses. This class stores attributes such as vehicle speed while maintaining common vehicle properties like route, capacity, and operational status. This design allows ExpressBus to interact directly with stops and schedules, while also supporting the calculation of travel times based on its specific speed, ensuring schedule information is accurately reflected in the system. The Schedule and Schedule_ExpressBus classes are defined to manage travel time information for each type of vehicle. Schedule stores data regarding arrival times for bus or train routes, whereas Schedule_ExpressBus is dedicated to express bus routes, keeping track of the corresponding estimated arrival times. Separating these two schedule classes enables the system to easily differentiate and handle different types of vehicles, while supporting precise and flexible scheduling for each route, directly linked to the stops managed by the Station class. Additionally, Passenger focuses on personal data, account balance, and purchased tickets, whereas Ticket stores all details of a ticket booking transaction. This arrangement ensures that each class has a clear scope of responsibility, facilitating system expansion or modification of specific parts without affecting others.

Regarding inheritance, the ExpressBus class is built upon the Vehicle class to inherit all common components such as route, capacity, and operational status. On this foundation, ExpressBus adds the speed attribute and overrides the calculateTravelTime() method to reflect faster travel compared to regular buses. This organization eliminates code duplication while maintaining the system's extensibility when integrating new types of vehicles.

Code Walkthrough

The public transportation station management system is built on the principles of object-oriented programming to effectively simulate a real-world system.

- Classes:

+ Vehicles: The base class for all public transportation vehicles. It includes common attributes like route, capacity, and operational status. The calculateTravelTime method is defined as virtual to allow subclasses to change how travel time is calculated.

+ ExpressBus: This class inherits from Vehicles, demonstrating inheritance. It adds its own speed attribute and overrides the calculateTravelTime method to reflect the express bus's faster speed.

+ Station: Manages the stops and schedules for each route. It uses the Schedule and Schedule_ExpressBus classes to distinguish between the schedules of different vehicle types.

+ Passengers and Ticket: These classes handle user interactions. The Passengers class allows users to book, cancel tickets, and deposit money, while the Ticket class stores and displays trip details.

- Core Program Functions:

Within the main function, the program provides two user roles: Passenger and Administrator.

+ Passenger Role:

As a Passenger, a user can:

- View their ticket booking history.
- Book a new ticket for a bus, express bus, or train. This process involves selecting a route, vehicle type, and departure/destination stations. The system then automatically calculates the travel time.
- Cancel a booked ticket.
- Deposit money into their account.

+ Administrator Role: As an Administrator, a user can add a new route to a specific station.

Test results:

- Test case 1:

+ Input:

1
2
2
1
1
2

+ Output:

Choose your role:
1. Passenger

2. Administrator

Choose: 1

==========

Name: Nguyen Khanh Hung

ID: P001

Choose function:

1. View ticket booking history

2. Book ticket

3. Cancel ticket

4. Deposit

Choose: 2

==========

BOOK TICKET

Choose routes:

1. MRT Line 1 (MRT)

2. Bus A

3. Bus B

4. Bus C

Choose: 2

Choose type of bus:

1. Bus (On-time)

2. Express Bus (On-time)

Choose: 1

Choose first stop:

1. City Center Arrival Time: 08:05

2. Green Park Arrival Time: 08:12

3. Airport Arrival Time: 08:20

4. Old Town Arrival Time: 08:30

Choose: 1

Choose final stop:

1. Green Park Arrival Time: 08:12

2. Airport Arrival Time: 08:20

3. Old Town Arrival Time: 08:30

4. Harbor Arrival Time: 08:50

Choose: 2

Booking ticket sucessful!

TICKET INFO:

Route: Bus A

Type: Bus

First Stop: City Center

Final Stop: Airport

Departure Time: 08:05

Arrival Time: 08:20

Travel Time: 15 minutes

+ Explanation about test case:

- In this test case, the user selects the role Passenger (enters 1), then chooses the Book Ticket function (enters 2).
- The system displays a list of routes, and the user selects Bus A (enters 2), with the bus type On-time (enters 1).
- Next, they choose the departure point City Center (08:05) (enters 1) and the destination Airport (08:20) (enters 2).
- The program looks up the distance between the stations (a total of 12.5 km), calculates the travel time (about 15 minutes), and then prints the ticket information, including the route, departure and arrival times, and the journey duration.

- Test case 2:

+ Input:

1

2

4

1

+ Output:

Choose your role:

1. Passenger

2. Administrator

Choose: 1

==========

Name: Nguyen Khanh Hung

ID: P001

Choose function:

1. View ticket booking history

2. Book ticket

3. Cancel ticket

4. Deposit

Choose: 2

==========

BOOK TICKET

Choose routes:

1. MRT Line 1 (MRT)

2. Bus A

3. Bus B

4. Bus C

Choose: 4

Choose type of bus:

1. Bus (Full)

Choose: 1

The bus is full

+ Explanation about test case:

- In this test case, the user selects the Passenger role (enters 1), then chooses the Book Ticket function (enters 2).
- The system displays a list of routes, and the user selects Bus C (enters 4). At the bus type selection step, the system only shows Bus (Full) because the route is already full.
- When the user enters 1, the program displays “The bus is full” meaning the ticket cannot be booked since the bus has no available seats.

About LLM Usage

During my coding process, whenever I encounter syntax or logic issues in my code, I use ChatGPT to point out the errors and suggest how to fix them, then I correct the mistakes in my code myself.