PUBLIC TRANSPORTATION STATION MANAGEMENT SYSTEM - REPORT

Name: Huỳnh Vũ Minh Hiền **Student's ID:** 24110091
Subject: Object - Oriented Programming

I. Object-Oriented Analysis (OOA) Model

1. Object and attribute:

- Object:

+ **Station**

+ **Vehicle**

+ **ExpressBus** (specialized Vehicle)

+ **Passenger**

+ **Schedule**

- Attribute:

+ **Station**: name, location, type, schedules

+ **Vehicle**: vehicleID, vehicleType, capacity, route, status

+ **ExpressBus** (inherits Vehicle): speed, stops

+ **Passenger**: name, ID, bookedTickets

+ **Schedule**: vehicleID, departureTime, arrivalTime, status

II. Identify Methods

Station: addSchedule(), removeSchedule(), displayInfo()

Vehicle: calculateTravelTime(), getID(), getType(), getRoute(), displayInfo()

ExpressBus: calculateTravelTime() (overridden), displayInfo() (overridden)

Passenger: bookRide(), cancelRide(), displayInfo()

Schedule: setStatus(), getStatus(), displayInfo()

III. Inheritance Relationships:

- **Vehicle** is a base class.

- **ExpressBus** inherits from **Vehicle**.

- Other classes (Passenger, Schedule, Station) are interact with Vehicle objects.

IV. Overview of the Public Transportation Station Management System

- **Vehicle (Base Class):** Represents general vehicles in the system. Includes basic details (ID, type, capacity, route, status).

- **ExpressBus (Derived Class):** Inherits from Vehicle but overrides calculateTravelTime() to simulate reduced travel time. Adds attributes like speed and number of stops.

- **Schedule:** Represents planned departure and arrival times for a specific vehicle. Tracks status (on time, delayed, canceled).

- **Station:** Manages schedules of vehicles at a specific location. Provides adding/removing schedule functionality.

- **Passenger:** Represents a passenger with ability to book or cancel rides. Tracks booked tickets.

This design follows **encapsulation** (private attributes with getters/setters), **inheritance** (ExpressBus : Vehicle), and **polymorphism** (overriding calculateTravelTime).

2. Code Walkthrough

Schedule class:
Stores and displays departure/arrival times for a vehicle.

Vehicle class:
Base class with calculateTravelTime() (default: baseTime).

ExpressBus class:
Derived from Vehicle. Overrides calculateTravelTime() to reduce time by 20%.

Passenger class:
Allows booking and canceling rides. Stores booked tickets in a vector.

Station class:
Stores multiple schedules and prints station information.

Main function:

- Creates a station, vehicles, schedules, and a passenger.
- Adds schedules to the station.
- Demonstrates booking, canceling, and displaying details.
- Tests polymorphism via overridden travel time.

V. System Operations

The Public Transportation Station Management System supports the following operations:

Station Operations

- Create stations with name, location, and type.
- Add or remove schedules (arrival/departure).
- Display all schedules and station details.

Vehicle Operations

- Create vehicles (bus, train) with route, capacity, and status.
- Assign vehicles to schedules at stations.
- Support specialized vehicles (ExpressBus) with overridden travel time calculation.

Passenger Operations

- Create passengers with personal information (name, ID).

- Book rides by selecting vehicle IDs.

- Cancel booked rides.

- Display passenger details and tickets.

Schedule Operations

- Create schedules including vehicleID, departure time, arrival time, and status.

- Update schedule status (on time, delayed, canceled).

- Display schedule details.

System Demonstration in Main Function

- Creates station, vehicles, schedules, and passengers.

- Adds schedules to stations and displays them.

- Allows passenger to book/cancel rides.

- Tests polymorphism (ExpressBus vs Vehicle travel time).

VI. Testing the System & Results

Test Case 1: Create and Display Station + Vehicles

```
Station st("Central Station", "Downtown", "Bus");Vehicle v1("B001", "Bus", 40, "Route A");ExpressBus eb1("E100", 30, "Route B");
```

Output: Vehicle and ExpressBus details displayed correctly.

Test Case 2: Add Schedules to Station

```
Schedule sc1("B001", "08:00", "10:00");Schedule sc2("E100", "09:00", "10:30");
st.addSchedule(sc1);
st.addSchedule(sc2);
st.displayInfo();
```

Output: Station shows both schedules with status “on time”.

Test Case 3: Passenger Books Rides

```
Passenger p1("Quang Truong", "P403");
p1.bookRide("B102");
p1.bookRide("B206");
```

Output: Passenger booked tickets are shown.

Test Case 4: Polymorphism – Travel Time

```
double baseTime = 60;
cout << v1.calculateTravelTime(baseTime);
cout << eb1.calculateTravelTime(baseTime);
```

Output:

Normal bus = 60 minutes.

Express bus = 48 minutes (20% faster).

3. LLM Usage

I used **ChatGPT (LLM)** for:

- Create attributes and methods for each class.

- Redesign the report.

- Add English comment for clarity.

- Idea development: outlining the overall structure of the program, including which classes should exist and how they can be connected.

- Debugging help: when errors occurred, I checked the LLM's suggestions, compared them with my code, and corrected mistakes myself.

4. Conclusion

This project demonstrates **Object-Oriented Programming** concepts including:

- **Encapsulation** (private attributes, public methods).

- **Inheritance** (ExpressBus inherits Vehicle).

- **Polymorphism** (overridden method calculateTravelTime).

- **Composition** (Station contains multiple Schedules).