Public Transportation Management System Documentation

Name: Ly Tran Gia Khang

ID: 24110098

---

1. Object-Oriented Analysis (OOA)

Following the 4-step OOA model, the system has the following objects:

1.1 Objects (Nouns)

- Station
- Vehicle
- Bus (inherits from Vehicle)
- EcoBus (inherits from Vehicle)
- Train (inherits from Vehicle)
- Metro (inherits from Vehicle)
- Passenger
- Schedule
- System (manager class)

1.2 Attributes (Descriptive Nouns)

- Station: name, location, type, list of vehicles, list of schedules
- Vehicle: type, route, capacity, bookedSeats, status, speed, distance
- Bus: (inherits Vehicle)
- EcoBus: (inherits Vehicle, modifies travel time calculation)
- Train: (inherits Vehicle)
- Metro: (inherits Vehicle)

- Passenger: name, age, ticketID, bookedVehicles
- Schedule: time, status, vehicle
- System: stations, passengers, vehicles

1.3 Methods (Verbs)

- Station: addVehicle(), addSchedule(), removeSchedule(), listSchedules(), displayInfo()
- Vehicle: bookSeat(), cancelSeat(), assignToStation(), virtual calculateTravelTime(), displayInfo(), displaySummary()
- Bus: override displayInfo(), displaySummary()
- EcoBus: override calculateTravelTime() (20% slower)
- Train: override displayInfo()
- Metro: override displayInfo()
- Passenger: bookTicket(), cancelTicket(), displayInfo(), displaySummary()
- Schedule: displayInfo()
- System: addStation(), addPassenger(), addVehicle(), chooseStation(), choosePassenger(), chooseVehicle(), displayStations(), displayVehicles(), displayPassengers()

1.4 Inheritance Relationships

- Vehicle → Bus, EcoBus, Train, Metro
- EcoBus overrides calculateTravelTime() to increase travel time by 20%
- Passenger books/cancels tickets and interacts with Vehicle objects
- System manages all major entities (stations, vehicles, passengers)

2. Class Design Explanation

The system is designed around five main classes (Station, Vehicle, Passenger, Schedule, System) with four specialized Vehicle subclasses (Bus, EcoBus, Train, Metro). Inheritance is used to avoid redundant code: Vehicle serves as a base class, and specialized types extend or override methods where necessary. For example, EcoBus inherits Vehicle but modifies travel time calculation to simulate slower routes.

3. Code Walkthrough

Example 1 – Override in EcoBus:

```
double calculateTravelTime(double distance) override {

    return (distance / speed) * 1.2;

}
```

-> EcoBus travel times are calculated as 20% longer compared to a regular bus.

Example 2 – System manages all objects:

```
void addStation(Station* _station){ stations.push_back(_station); }

void addPassenger(Passenger* _passenger){
passengers.push_back(_passenger); }

void addVehicle(Vehicle* _vehicle){ vehicles.push_back(_vehicle); }
```

-> The System class uses vectors to store entities, providing centralized management.

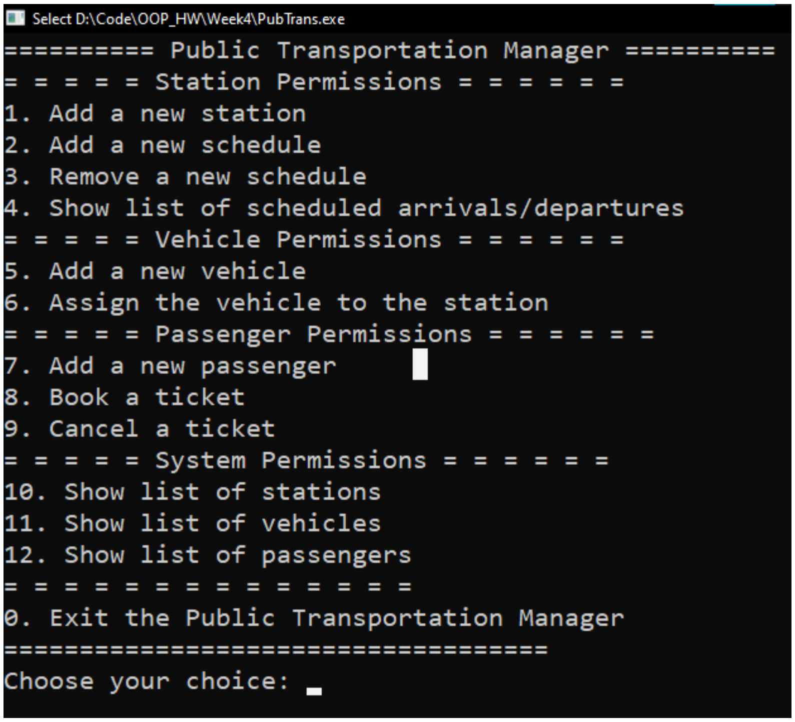
4. Testing & Output

4.1 Test Cases

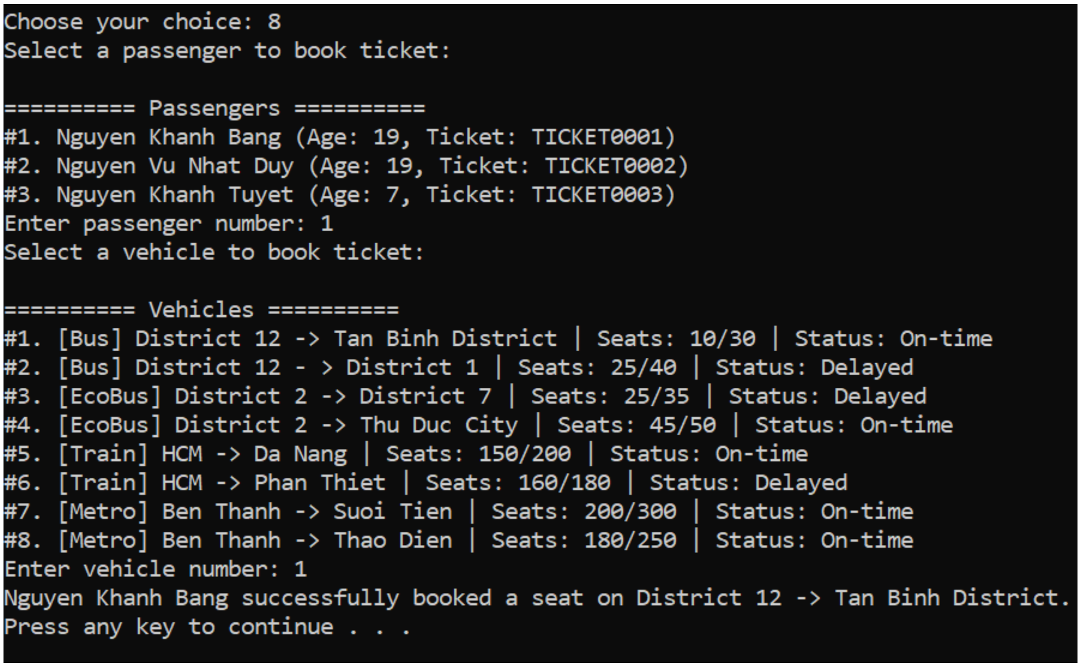
The main() function demonstrates the following:

- Creating stations of different types (Bus, Train, Metro).
- Adding vehicles and assigning them to stations.
- Adding schedules and preventing mismatched assignments (e.g., Bus to Train station).
- Booking tickets for passengers, preventing duplicate bookings.
- Cancelling tickets and updating seat counts.
- Displaying lists of stations, vehicles, passengers, and schedules.
- Edge case: booking when vehicle is at full capacity.
- Edge case: adding overlapping schedules at the same time slot.

4.2 Sample Output (excerpt)



```
========== Public Transportation Manager ==========
= = = = = Station Permissions = = = = = =
1. Add a new station
2. Add a new schedule
3. Remove a new schedule
4. Show list of scheduled arrivals/departures
= = = = = Vehicle Permissions = = = = = =
5. Add a new vehicle
6. Assign the vehicle to the station
= = = = = Passenger Permissions = = = = = =
7. Add a new passenger
8. Book a ticket
9. Cancel a ticket
= = = = = System Permissions = = = = = =
10. Show list of stations
11. Show list of vehicles
12. Show list of passengers
= = = = = = = = = = = = = = =
0. Exit the Public Transportation Manager
====================================
Choose your choice:
```



```
Choose your choice: 8
Select a passenger to book ticket:

========== Passengers ==========
#1. Nguyen Khanh Bang (Age: 19, Ticket: TICKET0001)
#2. Nguyen Vu Nhat Duy (Age: 19, Ticket: TICKET0002)
#3. Nguyen Khanh Tuyet (Age: 7, Ticket: TICKET0003)
Enter passenger number: 1
Select a vehicle to book ticket:

========== Vehicles ==========
#1. [Bus] District 12 -> Tan Binh District | Seats: 10/30 | Status: On-time
#2. [Bus] District 12 - > District 1 | Seats: 25/40 | Status: Delayed
#3. [EcoBus] District 2 -> District 7 | Seats: 25/35 | Status: Delayed
#4. [EcoBus] District 2 -> Thu Duc City | Seats: 45/50 | Status: On-time
#5. [Train] HCM -> Da Nang | Seats: 150/200 | Status: On-time
#6. [Train] HCM -> Phan Thiet | Seats: 160/180 | Status: Delayed
#7. [Metro] Ben Thanh -> Suoi Tien | Seats: 200/300 | Status: On-time
#8. [Metro] Ben Thanh -> Thao Dien | Seats: 180/250 | Status: On-time
Enter vehicle number: 1
Nguyen Khanh Bang successfully booked a seat on District 12 -> Tan Binh District.
Press any key to continue . . .
```

5. Use of LLM (ChatGPT)

During development, I used ChatGPT as a support tool for brainstorming and debugging. For example, I asked how to properly implement inheritance in

C++ and how to handle input validation. I also explored how to structure menu-driven programs. The final implementation and report are my own work, with ChatGPT serving as guidance and support.

6. Conclusion

The Public Transportation Management System successfully demonstrates key OOP concepts:

- Encapsulation: private attributes with getters/setters.
- Inheritance & Polymorphism: Vehicle as base class, EcoBus overrides travel time calculation.
- Abstraction: Vehicle defines pure virtual functions (displayInfo, displaySummary).

The program runs correctly, supports all required operations, and models real-world public transportation management in Vietnam by including buses, express buses, trains, and metro lines.