DOCUMENTATION FOR PUBLIC TRANSPORTATION MANAGEMENT SYSTEM

- Written by Student ID: 24110073 –

I. OOA Analysis

Classes / nouns:

- Bus

```
class Bus{
    protected:
        string route;
        int capacity;
        bool status;
        string timeStatus;
```

- ExpressBus

```
class ExpressBus : public Bus {
    private:
    double speed; //higher speed for express
    public:
```

- Passenger

```
class Passenger{
    private:
    string name;
    int age;
    int ID;
    bool bookedTicket;
    string busType;
```

- Schedule

```
class Schedule{
    public:
    string vehicleID;
    string arrivalTime;
    string departureTime;
    Schedule(string vID, string at, string dt): vehicleID(vID), arrivalTime(at), departureTime(dt) {}
```

- Station

```
class Station{
    private:
    string name;
    string location;
    string type;
```

Object's attributes / Descriptive nouns:

- Bus:
    1. Route
    2. Capacity
    3. Status
    4. timeStatus
- Express Bus (Inherited bus class's attributes with one new addition speed)
- Passenger
    1. Name
    2. Age
    3. ID
    4. BookedTicket
    5. BusType (selected bus type for the ticket)
- Schedule
    1. vehicleID
    2. arrivalTime
    3. departureTime
- Station
    1. Name
    2. Location
    3. Type

Methods (Verbs):

- Bus
    1. calculateTravelTime (estimates route's travel time in hours)
    2. calculateOccupancy (calculates the set occupancy in percentage)
    3. showOccupancy (shows calculated occupancy)
    4. displayInfo (self explanatory)
    5. TravelTime (shows calculated travel time)
- Express Bus
  - Same as Bus class
- Passenger
    1. bookTicket (self explanatory)
    2. cancelTicket (also self explanatory)
    3. isTicketBooked (checks if the person has already booked)
    4. displayInfo
- Schedule: displayInfo
- Station: displayInfo

II. Inheritance

- ExpressBus inherited attributes from the Bus class (route, capacity, status) with an extra attribute speed (because express buses are faster). The class overrides 3 of the methods (calculateOccupancy, calculateTravelTime, displayInfo) displaying different info for the bus.

III. Code walkthrough

1. Bus

- Represents normal buses

- Has basic attributes (route, capacity, punctuality status, functionality).

- Includes **new** methods to calculate travel time, occupancy and displays info with a **new** functionality status.

- Serves as a base for ExpressBus to inherit.

2. ExpressBus

- Represents express buses that move faster than regular bosses.

- Inherit attributes from Bus class.

- Overrides calculateOccupancy, calculateTravelTime, displayInfo showing different details representing inheritance and polymorphism.

3. Passenger

- Represents passenger using the transportation service.

- Basic attributes (name , ID, has ticket, book/cancel ticket).

- Methods allowing passenger to book and cancel tickets, check for existing tickets and their assigned bus type.

4. Schedule

- Schedule for bus routes.

- Basic attributes (bus ID, arrivalTime, departureTime).

- Allows user to assign a route, arrival and departure time for buses.

5. Station

- Station for passengers to get on the bus.

- Basic info (name , location, station type).

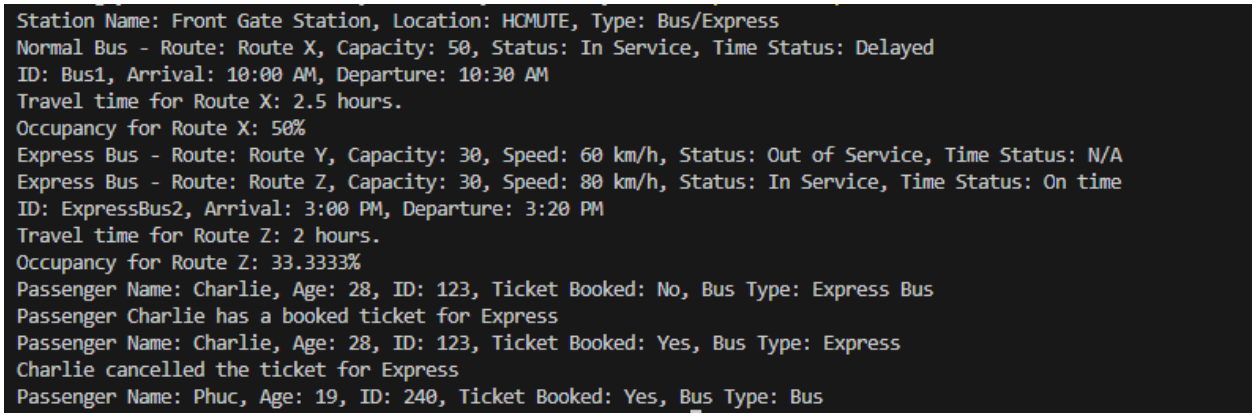
- Allows user to create name and position and type of the station.

Key parts of the code:

- Passenger stores a passenger's personal info and ticket status.
- Bus stores a bus's info and calculations.
- ExpressBus is a more advanced version of the Bus class (speed).
- ExpressBus overrides behavior of Bus demonstrating inheritance and polymorphism.
- Station links the bus and express bus.

IV. Test results

The necessary info about the buses, their routes and schedule and calculations needed for occupancy rate and arrival time with a newly implemented functionality status for the bus if it's still in use (in case the bus is out of service). Stations are created with assigned schedules for the bus routes. Passengers are assigned with a name, ID and ticket status with functional book/cancel ticket function, it also updates the passenger's ticket status.



V. LLM Usage

The majority of the code was written by me but this assignment was made possible with the help of GitHub Copilot. I had problems with setting up the calculations for occupancy rate hence I asked the LLM with the prompt "Help me fix occupancyRate method". It also corrected minor typos and errors commonly made such as CamelCase.