**Object - Oriented Programming**

**Name:** Lê Sĩ Nhật Khuê **Student's ID:** 24110102

**Assignment 5 – Week 4:** Public Transportation Station Management System

## I/ Object-Oriented Analysis (OOA) Model

Base on the information provided, we have some analysis

- About **objects**, we can define some objects such as Station, Vehicle, Express Bus, Passenger.
- About **attributes** for each object:
    + For **Station:** name, location, type, listSchedules.
    + For **Vehicle:** route, capacity, status and station (to store the station).
    + For **ExpressBus:** inherits from Vehicle and add speed, stops.
    + For **Passenger:** name, ID, listBookedTickets.
- About **methods** for each object:
    + For **Station:** addSchedule, displayInfo.
    + For **Vehicle:** assignStation, displayInfo.
    + For **ExpressBus:** calculateTravelTime, displayInfo.
    + For **Passenger:** bookRide, cancelRide, displayInfo.
- About **inheritance relationships**:
    + ExpressBus inherits from Vehicle.
    + A station can have multiple vehicles, but a vehicle usually belongs to one station at a time.
    + A vehicle can serve multiple passengers.

## II/ Overview the Public Transportation Station Management System

The Public Transportation Management System is designed to model a simplified environment of stations, vehicles, schedules, and passengers. The goal is to capture the essential aspects of a transportation system while keeping the design manageable for an educational assignment. The system demonstrates how different objects interact, how inheritance can be applied, and how encapsulation ensures that data is properly managed.

- Two function **getDistrictName()** and **getStationName()** are helper functions mapping integers to names. This design choice allows users to input simple numeric values while still displaying user-friendly outputs.

- **Schedules class** defines arrival and departure times for stations, supporting station management in a structured way. It ensures that each station can track when a vehicle arrives or leaves, simulating a real-world transportation schedule.

**- Stations class** represents physical locations such as bus, train, or metro stations. Besides storing station details, it manages schedules and coordinates the vehicles assigned to it. This reflects the real-world importance of stations as central hubs in a transportation network.

**- Vehicles class** defines common attributes such as vehicle ID, type, and capacity. It also allows vehicles to be assigned to stations, showing the relationship between transportation assets and their locations.

- **ExpressBus class** inherits from Vehicle and extends it with additional attributes like speed and stop count. It overrides the travel time calculation and information display methods, which illustrates how inheritance and method overriding can adapt a base design for more specialized use cases.

**- Passengers class** interacts with the system by booking vehicles, showing the association between passengers and transport services. This design simulates a core feature of any transportation system: enabling people to choose and reserve transport options according to their needs.

- **In main():** Firstly, we have some the available information. Then we have a menu for adding stations, vehicles, passengers, booking tickets, and viewing data to the user can test the code easily.

## III/ How to do this assignment

Firstly, I carefully read the requirements. I noted the real-world context (transportation system) and identified possible objects such as *stations, vehicles, passengers,* and *schedules*.

After that, I started with step 1 is **model OOA** (Object – Oriented Analysis. I listed all potential objects, their attributes, and methods. I also thought about the relationships like vehicles are assigned to stations, passengers book vehicles, and vehicles can have schedules… To refine my design, I used an LLM (**ChatGPT**) to brainstorm ideas.

Next, I started coding by myself with each class respectively from relationships and supported by ChatGPT when I struggle about the concepts, or ideas to implement. This process occur in many hours in day, around the writing and fixing code.

During the assignment, I encountered **several difficulties**.

+ Firstly, although the classes already had attributes and methods, I still struggled to design a logical way of linking the objects together in a realistic application. Finding the right relationships between stations, vehicles, and passengers was not straightforward.

+ Secondly, I faced some issues with inheritance, especially when overriding methods. Small errors in overriding caused unexpected behavior, and I had to review my class design to ensure consistency.

+ Finally, I found it challenging to make my main function consistent and organized. Since testing depends on how the program is structured, I had to carefully refine main so that different test cases could be executed smoothly and demonstrate the system's functionalities effectively.

**Despite these challenges**, I gradually solved them by rethinking the class design, debugging the inheritance relationships, and restructuring the testing process. These experiences helped me strengthen both my understanding of object-oriented programming and my practical coding skills.