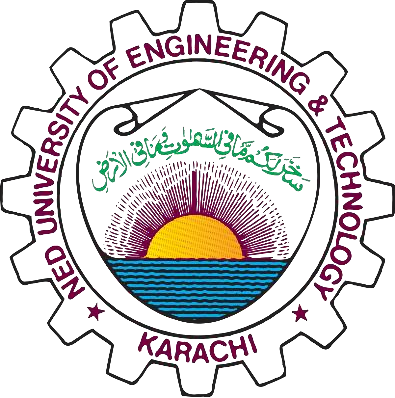
NED University of Engineering and Technology



***BUS TRANSPORT MANAGEMENT SYSTEM***

Project Report

Object Oriented Programming (CT-260)

**Group Members: Muhammad Abdullah Ayub CTCY-042 Muhammad Azhan Javed CTCY-046 Muhammad Hasan Khan CTCY-047 Aaron Mansoor Minhas CTCY-019**

***Bus Transportation Management System Report***

### **Problem Statement**

Managing the transportation logistics for a school can be a complex task involving multiple variables like routes, schedules, and the varying needs of students and staff. Without a systematic approach, ensuring efficient and timely transportation becomes challenging, often leading to confusion and delays. There is a need for a comprehensive system that can manage student data, assign them to appropriate buses, and ensure a smooth operational flow.

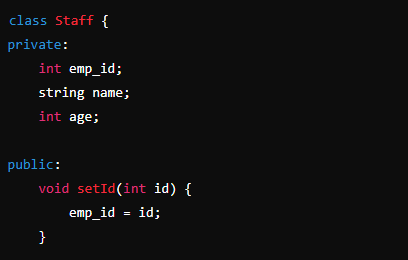
### **Project Description**

The Transportation Management System (TMS) is a software solution designed to streamline the management of school transportation. The system enables administrators to manage student and staff information, assign buses to students, track routes, and maintain schedules. This project aims to automate and simplify the transportation management process, ensuring efficiency and reliability.

### **Principles of OOP Used and Justification**

#### 1. Encapsulation

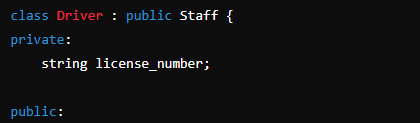
Data members (emp\_id, name, age, etc.) in the classes are typically declared as private and accessed through public methods (setid, getid, etc.). This ensures data security and control over how data is accessed and modified.



#### 2. Inheritance

Inheritance allows a new class to inherit the properties and methods of an existing class. This promotes code reusability and establishes a natural hierarchy between classes.

**Justification:** The Driver class inherits from the Staff class, leveraging the common attributes and methods of the Staff class while adding additional functionalities specific to an administrator.



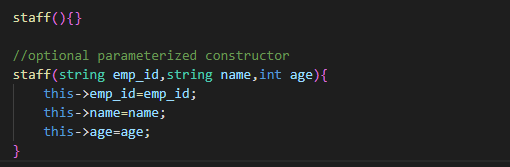
#### 3. Polymorphism

Function overloading for pay\_calc function demonstrates compile-time polymorphism. The function pay\_calc is overloaded to calculate payment differently based on the number of arguments passed.

#### 

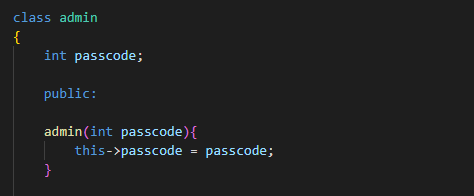
#### 4. Constructor Overloading

Constructors in classes such as staff, driver, attendent, and students demonstrate constructor overloading by providing both default and parameterized constructors.



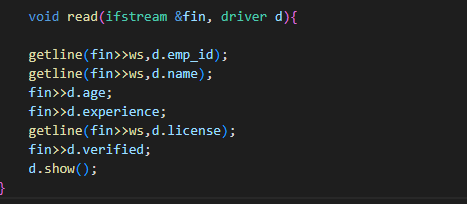
#### 5. Access Specifiers

Access specifiers (public, private) are used to define the visibility of class members. For example, member variables are usually private, while methods accessing or modifying these variables are public.



#### 6. File Handling

The program reads and writes data to/from files (Student\_data.txt, Driver\_data.txt, Attendent\_data.txt). File handling is used for storing and retrieving student and staff information.



#### 6. Object-oriented Modelling

The program models real-world entities (students, staff, admin, etc.) as objects, with each class encapsulating data and behavior related to its corresponding entity.

**Description of Functionalities**

**Adding Student Data**

* **Functionality**: Allows administrators to add student information such as student ID, name, age, class, and assigned bus directly into Student\_data.txt. This is facilitated through encapsulated methods within the Student class, ensuring secure data handling. The data is written to a file using file handling techniques for persistence.

**Assigning Buses**

* **Functionality**: Automatically assigns buses to students based on predefined routes and schedules.

**Displaying Information**

* **Functionality**: Displays detailed information about students, staff, and buses in a user-friendly format.

**Managing Routes and Schedules**

* **Functionality**: Allows administrators to manage and update routes and schedules for efficient transportation.

**File Handling**

* **Functionality**: Saves and loads student and staff information to and from files for persistence.

***Limitations/Future Enhancements***

**Limitations**

**User Interface**

* The current implementation is console-based, which may not be user-friendly for all users.

**Real-Time Tracking**

* The system does not support real-time tracking of buses and students, which could be a significant limitation.

**Future Enhancements**

**Graphical User Interface (GUI)**

* Developing a GUI to enhance user interaction and make the system more accessible to non-technical users.

**Real-Time Tracking**

* Integrating GPS and other tracking technologies to provide real-time tracking of buses and students.

**Mobile Application**

* Creating a mobile application for parents and staff to track buses, view schedules, and receive notifications.

**Automated Notifications**

* Implementing an automated notification system to alert parents and students about bus delays or schedule changes.

### **Conclusion**

The Transportation Management System (TMS) is a robust solution aimed at simplifying the complex process of managing school transportation. By leveraging the principles of object-oriented programming, the system ensures data integrity, reusability, and scalability. While the current implementation addresses many key aspects of transportation management, future enhancements like a GUI, real-time tracking, and automated notifications can significantly improve its functionality and user experience.