Practical Exercise — Smart Vehicle Management System

1. Description

This practical exercise demonstrates the **four major Object-Oriented Programming (OOP)** principles in C++ using a Smart Vehicle Management System. Students will create different types of vehicles such as *Cars, Trucks, and Electric.*

Cars while applying *abstraction, encapsulation, inheritance, and polymorphism* in a real-world scenario. The system simulates vehicle operations such as starting vehicles, managing data safely, and calculating operating costs differently depending on the vehicle type.

2. Objective

Develop a Vehicle Management System that demonstrates the four core principles of Object-Oriented Programming (OOP):

- **Abstraction** → Hide implementation details using abstract classes and virtual functions.
- **Encapsulation** → protecting vehicle data using private members- and public getter/setter methods.
- **Inheritance** → creating specialized vehicle types from a base Vehicle class
- **Polymorphism** → allowing different vehicles to behave differently through the same interface

3. Requirements

2.1. Abstraction

- *Create an abstract base class named **Vehicle**.*
- *Use pure virtual functions such as **start() and calculateCost()**.*

2.3. Encapsulation

- Create private member variables such as *vehicle number, brand, and speed*.
- Provide public **setter and getter** methods and also validate invalid speed values

2.4. Inheritance

- Demonstrate at least - *Single Inheritance and Multilevel Inheritance*
    - Implement single inheritance using *Car and Truck classes*.
    - Implement multilevel inheritance using ElectricCar derived from Car.
    - Example: *Vehicle → Car → ElectricCar*

2.5. Polymorphism

- Use a Vehicle pointer array to demonstrate runtime polymorphism.
    - *virtual double calculateCost() - e*ach class calculates cost differently.

smart_vehicle_management.cpp

```cpp
#include <iostream>
#include <string>

using namespace std;

// ======= Abstract Base Class ========
class Vehicle {
    private:
        string vehicleNumber;
        string brand;
        int speed;

    public:
        Vehicle() {
            vehicleNumber = "Unknown";
            brand = "Unknown";
            speed = 0;
        }

        // Parameterized Constructor
        Vehicle(string num, string br, int sp) {
            vehicleNumber = num;
            brand = br;
            setSpeed(sp);
        }

        // Encapsulation (Setters & Getters)
        void setVehicleNumber(string num) {
            vehicleNumber = num;
        }

        string getVehicleNumber() {
            return vehicleNumber;
        }

        void setBrand(string br) {
            brand = br;
        }

        string getBrand() {
            return brand;
```

```
        }

        void setSpeed(int sp) {
            if(sp >= 0)
                speed = sp;
            else {
                cout << "Invalid speed! Setting speed to 0.";
                speed = 0;
            }
        }

        int getSpeed() {
            return speed;
        }

        // Abstraction (Pure Virtual Functions)
        virtual void start() = 0;
        virtual double calculateCost() = 0;
        // Virtual Destructor
        virtual ~Vehicle() {}
};

// ================ SINGLE INHERITANCE ================
class Car : public Vehicle {
    protected:
        double fuelUsed;

    public:
        Car(string num, string br, int sp, double fuel) : Vehicle(n
            fuelUsed = fuel;
        }

    void start() override {
        cout <<"Car is starting..."<<endl;
    }

    doublecalculateCost() override {
        return fuelUsed * 100;
    }

    void  displayCarInfo() {
        cout <<"\\n--- Car Information ---\\n";
        cout << "Vehicle No: "<< getVehicleNumber() << endl;
        cout << "Brand: "<< getBrand() << endl;
        cout << "Speed: "<< getSpeed() << " km/h\\n";
        cout << "Fuel Used: "s<< fuelUsed << " liters\\n";
    }
```

```
};

// ================= MULTILEVEL INHERITANCE ================
class ElectricCar : public Car {
    private:
        double batteryUnits;

    public:
        ElectricCar(
            string num,
            string br,
            int sp,
            double fuel,
            double battery
        )
    : Car(num, br, sp, fuel) {
        batteryUnits = battery;
    }

    void start() override {
        cout<<"Electric Car starts silently..."<<endl;
    }

    double calculateCost() override {
        return batteryUnits * 15;
    }

    void displayElectricInfo() {
        cout << "\\n--- Electric Car Information ---\\n";
        cout << "Vehicle No: " << getVehicleNumber() << endl;
        cout << "Brand: " << getBrand() << endl;
        cout << "Speed: " << getSpeed() << " km/h\\n";
        cout << "Battery Units: " << batteryUnits << " kWh\\n";
    }
};

// ================= SINGLE INHERITANCE ================
class Truck : public Vehicle {
    private:
        double cargoWeight;

    public:
        Truck(string num, string br, int sp, double cargo)
            : Vehicle(num, br, sp) {
            cargoWeight = cargo;
        }
```

```
        void start() override {
            cout << "Truck engine is roaring..." << endl;
        }

        double calculateCost() override {
            return cargoWeight * 50;
        }

    void displayTruckInfo() {
        cout << "\\n--- Truck Information ---\\n";
        cout << "Vehicle No: "<< getVehicleNumber() << endl;
        cout << "Brand: "<< getBrand() << endl;
        cout << "Speed: "<< getSpeed() << " km/h\\n";
        cout << "Cargo Weight: "<< cargoWeight << " tons\\n";
    }
};

// ================ MAIN FUNCTION ================
int main() {
    Car car1("CAR101", "Toyota", 120, 10);

    ElectricCar ecar1("EV202", "Tesla", 150, 0, 25);

    Truck  truck1("TR303", "Volvo", 100, 50, 100);

    Vehicle* vehicles[3];

    vehicles[0] = &car1;
    vehicles[1] = &ecar1;
    vehicles[2] = &truck1;

    cout<<"===== Vehicle Management System =====\\n\\n";

    for(int i = 0;i < 3; i++) {
        vehicles[i]->start();
        cout<<"Operating Cost: "<< vehicles[i]->calculateCost()<< e
    }

    car1.displayCarInfo();
    ecar1.displayElectricInfo();
    truck1.displayTruckInfo();

    return 0;
}
```

📄 Sample Output

console_output.txt

```
===== Vehicle Management System =====

Car is starting...
Operating Cost: 1000

Electric Car starts silently...
Operating Cost: 375

Truck engine is roaring...
Operating Cost: 400


--- Car Information ---
Vehicle No: CAR101
Brand: Toyota
Speed: 120 km/h
Fuel Used: 10 liters


--- Electric Car Information ---
Vehicle No: EV202
Brand: Tesla
Speed: 150 km/h
Battery Units: 25 kWh


--- Truck Information ---
Vehicle No: TRK303
Brand: Volvo
Speed: 90 km/h
```