Obeject-Oriented Program

Assignment

Name: Vansh kumar

UID: 24BCA10486

Subject: Object-Oriented Program (OOPS)

Section: 3A

Submitted To: Jyoti Rani

Designation: Assistant Professor

Topic: Traffic Management Simulation System.

Abstract:

The system models a traffic junction with signals for four directions (North, South, East, West). Each direction has vehicles that must follow traffic light rules. The program simulates signal changes (RED, YELLOW, GREEN) and controls the flow of vehicles accordingly.

It aims to demonstrate how OOP concepts can be used to design a simple yet efficient real-world system.

Objective:

To simulate and manage traffic at an intersection using Object-Oriented Programming concepts such as **classes, inheritance, polymorphism, and encapsulation.**

Key OOP Concepts Used:

- **Class and Object:** For vehicles, signals, and junctions.
- **Inheritance:** Different types of vehicles (Car, Bus, Bike) inherit from the Vehicle base class.
- **Polymorphism:** Displaying behavior of vehicles differently using virtual functions.
- **Encapsulation:** Keeping signal states and vehicle counts private.

- **Abstraction:** Hiding the complexity of traffic flow logic inside classes.

Class Diagram (Conceptual):

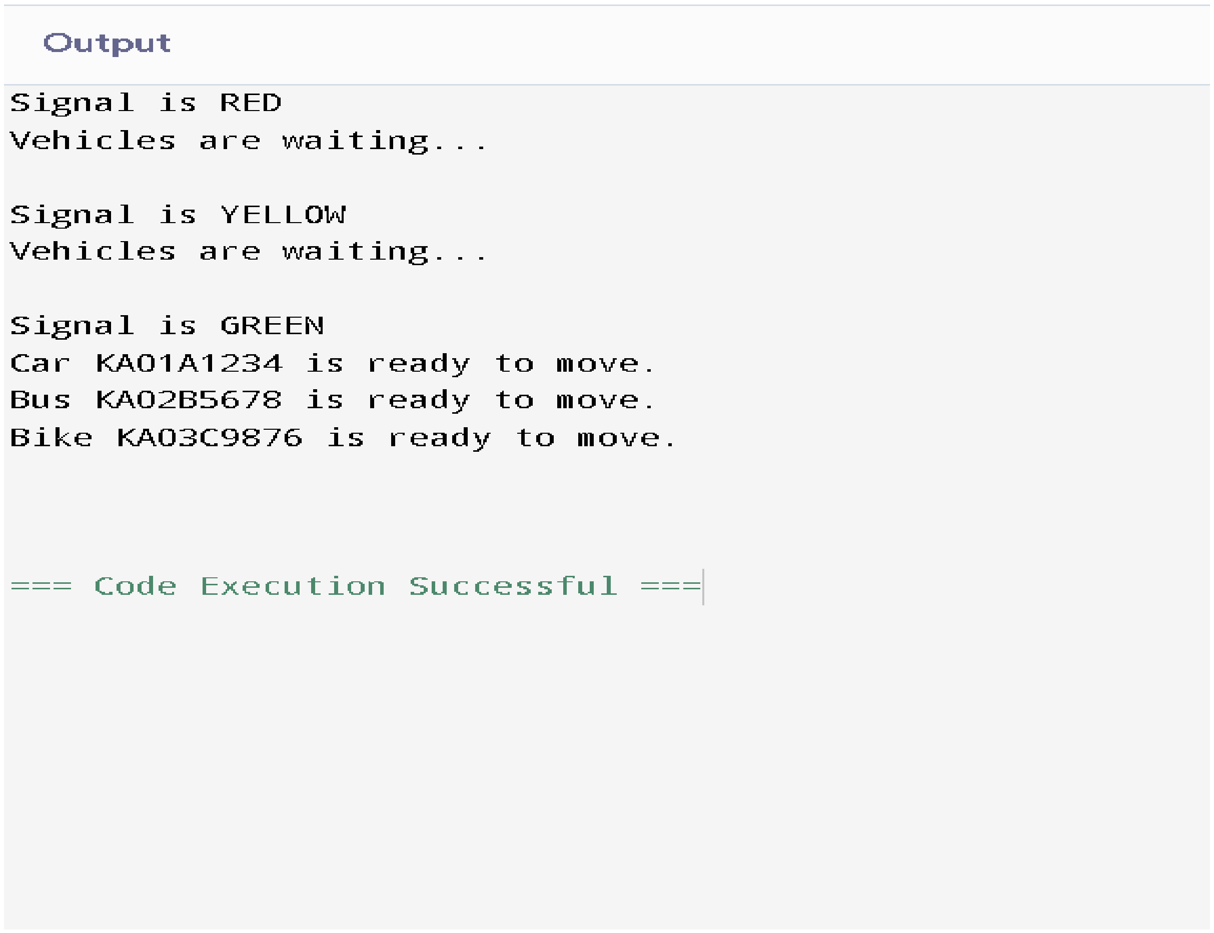
```
         +-----------------+
      |     Vehicle      |
      +-----------------+
      | - vehicleNo      |
      | - type           |
      +-----------------+
      | + displayInfo()   |
      +-----------------+
               ^
               |
  -----------------------------------
  |             |              |
+--------+     +--------+       +--------+
|  Car   |     |  Bus   |       |  Bike  |
+--------+     +--------+       +--------+
| +displayInfo() | +displayInfo()   | +displayInfo() |
+--------+     +--------+       +--------+
      +---------------------+
      |   TrafficSignal     |
      +---------------------+
      | - signalState       |
      +---------------------+
      | +changeSignal()     |
      | +displaySignal()    |
      +---------------------+

      +----------------------+
      |    TrafficControl    |
      +----------------------+
      | - signal : TrafficSignal |
      | - vehicles[]         |
      +----------------------+
      | +simulateTraffic()   |
      +----------------------+
```

Sample Code (C++):

#include #include #include #include using namespace std;

```
// Base Class class Vehicle { protected: string vehicleNo, type; public:
Vehicle(string no, string t) : vehicleNo(no), type(t) {} virtual void
displayInfo() { cout << type << " " << vehicleNo << " is waiting at the
signal.\n"; } };

// Derived Classes class Car : public Vehicle { public: Car(string no) :
Vehicle(no, "Car") {} void displayInfo() override { cout << "Car " <<
vehicleNo << " is ready to move.\n"; } };

class Bus : public Vehicle { public: Bus(string no) : Vehicle(no, "Bus")
{} void displayInfo() override { cout << "Bus " << vehicleNo << " is
ready to move.\n"; } };

class Bike : public Vehicle { public: Bike(string no) : Vehicle(no,
"Bike") {} void displayInfo() override { cout << "Bike " << vehicleNo
<< " is ready to move.\n"; } };

// Traffic Signal Class class TrafficSignal { private: string signalState;
public: TrafficSignal() : signalState("RED") {} void changeSignal(string
state) { signalState = state; } void displaySignal() { cout << "Signal is "
<< signalState << endl; } string getSignal() { return signalState; } };

// Traffic Control Class class TrafficControl { private: TrafficSignal
signal; vector<Vehicle*> vehicles; public: void addVehicle(Vehicle* v)
{ vehicles.push_back(v); }

void simulateTraffic() {
    string states[] = {"RED", "YELLOW", "GREEN"};
    for (int i = 0; i < 3; i++) {
        signal.changeSignal(states[i]);
        signal.displaySignal();

        if (signal.getSignal() == "GREEN") {
            for (auto v : vehicles) v->displayInfo();
        } else {
            cout << "Vehicles are waiting...\n";
        }

        this_thread::sleep_for(chrono::seconds(2));
        cout << endl;
    }
}


};

// Main Function int main() { TrafficControl control;
control.addVehicle(new Car("KA01A1234")); control.addVehicle(new
Bus("KA02B5678")); control.addVehicle(new Bike("KA03C9876"));

control.simulateTraffic();

return 0;


}
```

Output:



```
Signal is RED
Vehicles are waiting...

Signal is YELLOW
Vehicles are waiting...

Signal is GREEN
Car KA01A1234 is ready to move.
Bus KA02B5678 is ready to move.
Bike KA03C9876 is ready to move.



=== Code Execution Successful ===
```

Conclusion:

This project demonstrates how OOP concepts can be used to model a real-world system like traffic management. It shows how classes, inheritance, and polymorphism can make code modular, reusable, and easy to maintain.