לפנייך אפיון למערכת לפיתוח. בחרי שני design patterns שונים וממשי בעזרתם את המערכת:

המערכת אחראית לניהול רמזור חכם. הרמזור יכול להיות במצב אדום או ירוק.

במצב ההתחלתי הרמזור אדום.

ישנם רכבים חכמים שמחכים להתחלפות הרמזור מאדום לירוק, כל רכב כזה שולח הודעה למערכת הרמזור כשהוא מגיע אליו, וכשמגיע מספר מסוים של רכבים שמחכים הרמזור נהפך לירוק, וכל הרכבים מקבלים התראה שהם יכולים להמשיך בנסיעה.

לאחר פרק זמן קבוע הרמזור מתחלף בחזרה לאדום. ושוב רכבים שמגיעים שולחים הודעה.

שימי לב:

1. לממש את הdesign pattern לפי כל העקרונות שלמדנו בשיעור
2. להוסיף פונקציית main שבודקת את הקוד

using System;

using System.Collections.Generic;

using System.Threading;

// Observer Pattern - Interface for observers (cars)

interface IObserver

{

void Update();

}

// Subject - Traffic Light System

class TrafficLight

{

private List<IObserver> observers = new List<IObserver>();

private TrafficLightState state;

public TrafficLight()

{

state = new RedLightState(this); // Initial state

}

public void AddCar(Car car)

{

observers.Add(car);

Console.WriteLine($"Car {car.CarId} is waiting.");

state.CarArrived();

}

public void NotifyCars()

{

foreach (var car in observers)

{

car.Update();

}

observers.Clear();

}

public void SetState(TrafficLightState newState)

{

state = newState;

}

public int GetWaitingCars() => observers.Count;

}

// Concrete Observer - Car

class Car : IObserver

{

public int CarId { get; }

public Car(int id)

{

CarId = id;

}

public void Update()

{

Console.WriteLine($"Car {CarId} received green light notification and is driving.");

}

}

// State Pattern - Abstract State Class

abstract class TrafficLightState

{

protected TrafficLight trafficLight;

public TrafficLightState(TrafficLight trafficLight)

{

this.trafficLight = trafficLight;

}

public abstract void CarArrived();

}

// Concrete State - Red Light

class RedLightState : TrafficLightState

{

private const int MinCars = 3; // Minimum cars to turn green

public RedLightState(TrafficLight trafficLight) : base(trafficLight) { }

public override void CarArrived()

{

if (trafficLight.GetWaitingCars() >= MinCars)

{

Console.WriteLine("Enough cars arrived. Changing light to green.");

trafficLight.SetState(new GreenLightState(trafficLight));

trafficLight.NotifyCars();

trafficLight.SetState(new RedLightState(trafficLight));

}

}

}

// Concrete State - Green Light

class GreenLightState : TrafficLightState

{

public GreenLightState(TrafficLight trafficLight) : base(trafficLight) { }

public override void CarArrived()

{

Console.WriteLine("Cars can pass. No need to wait.");

}

}

// Main function

class Program

{

static void Main()

{

TrafficLight trafficLight = new TrafficLight();

List<Car> cars = new List<Car> { new Car(1), new Car(2), new Car(3), new Car(4), new Car(5) };

foreach (var car in cars)

{

trafficLight.AddCar(car);

Thread.Sleep(1000); // Simulate cars arriving one by one

}

}

}