Design a Traffic Controller System for a Junction

Requirements

1. Synchronize movement of vehicles without collisions
2. Display wait time for vehicle driver
3. Sense density of vehicles and set wait time accordingly
4. Capture photos of license plate of vehicle violating traffic rule
5. Provide control for emergency vehicles

Class VehicleDriver
Data: vehicle
Behavior: takeAction (trafficLights) {

```
If (tafficLight.greenLight == ON ) {
   If (No Vehicle in front OR Vehicle in front moving OR No Pedestrians to yield) {
      vehicle.drive();
    } else {
      vehicle.applyBrakes();
    }
  } else if (TrafficLight.yellowLight == ON) {
     vehicle.startEngine();
  } else {
     If (vehicle.state == Moving) {
        vehicle.stop();
     }
     If (vehicle.engineState != OFF) {
        vehicle.stopEngine(); // Save fuel,  Save Environment
     }
  }
}
```

Behavior: inspectLights(trafficLights) {

```
For every few seconds {
        takeAction(trafficLights);
    }
}
```

Class TrafficLight
Data: color // Possible colors - Red, Yellow, Green
shape // Possible shapes - Solid, Arrow
blinker // Possible state - Blink or Not

```
        state // ON or OFF
Behavior: turnONSingleLight (Shape, Color, Blink) [
            If (State != ON) {
               State = ON;
            }
            //  Set shape, Color, Blinker
           }
Behavior: turnOFFLight () [
            // Set state to OFF
           }
```

```
Class LEDDisplay
Data: numbers //  0 – 9
      digits // 1-3
      state // ON/OFF
Behavior: displayNumber(number) {
                  If (state != ON) {
                     //set state to ON
                  }
                  // Display the number of
        }
Behavior: turnOFFDisplay () {
                  // Set State to OFF
        }
```

```
Class EmergencyController
Data: state // ON/OFF
Behavior: receiveSingalFromEmergencyVehicle( RemoteSignal) {
                  If (RemoteSignal == ON) {
                           // Set State to ON
                  } else {
                           // Set state to OFF
                  }
        }
Behavior: getState() {
                  // return state of EmergencyController
        }
Behavior: informTrafficControllerSystem() {
                  TrafficControllerSystem.ServiceEmergenyRequest()
        }
```

```
Class VehicleSensor
Data: vehicleCounter, transmitter, receptor
Behavior: calculateDensity() {
        // Sense and count the Vehicles in the lane
    }
Behavior: returnVehicleDensity () {
        This.CalculateDensity();
        // Return Vehicle Density
    }
```

```
Class TrafficViolationCamera
Data: camera, sensors, internetTransmitter, state
Behavior: capturePhoto() {
        //Capture the photo of License Plate
    }
Behavior: sendPhotoToCentralTrafficDatabase() {
        // Send photo to database
    }
Behavior: checkTrafficRuleViolation() {
        If ( state == ON) {
          // Sense movement on Red light except right turn
          This.CapturePhoto()
          This.SendPhotoToCentralTrafficDatabase()
          This.CheckTrafficRuleViolation
        }
    }
Behavior: turnOffCamera() {
        State = OFF;
    }
```

```
Class TickDownTimer
Data: clock
Behavior: initializeTickDownTimer( TimeInSeconds) {
        // Start Clock
        // Tick Down from TimeInSeconds to zero
        This.InformControllerSystem();
    }
```

```
Behavior: informControllerSystem() {
                TrafficControllerSystem.ServiceTimerExpiry();
        }
```

```
Class TrafficControllerSystem
Data: List<Road> roads, List<Lane> lanes, List<TrafficLight> trafficLights, List<LEDDisplay> lights,
List<VehicleSensors> sensors,

Behavior: initializeAndStartTrafficControllerSystem() {
                foreach (road in roads) {
                        foreach( Lane in the road) {
                                // create an object of traffic light
                                TrafficLight t = new TrafficLight();
                                trafficLights.add(t);
                        }
                }

                foreach (road in roads) {
                        // Create an instance of traffic Violation camera
                        TrafficViolationCamera c = new TrafficViolationCamera();
                        c.CheckTrafficRuleViolation();

                        // Create an object of LED display
                        LEDDisplay d = new LEDDisplay();
                        Lights.add(d);

                        // Create an object of Vehicle Sensor
                        VehicleSensor s = new VehicleSensor();
                        Sensors.add(s);
                }

                // create an object of Tick Down Timer
                TickDownTimer timer = new TickDownTimer;

                Foreach s in sensors {
                        s. ReturnVehicleDensity();
                        // Calculate the maximum density;
                }
                TimeValueForMaximumDensity  =
This.CalculateTickDownTimerValue(maximumVehicleDensity);
                Timer.InitializeTickDownTimer(TimeValueForMaximumDensity);
                // For road with maximum density turn green light and suitable shape & blinker
```

```
                // For all other roads turn the red light and suitable shape & blinker
        }

Behavior: calculateTickDownTimerValue (VehicleDensity) {
                Return a time Value Suitable for the density;
        }
Behavior: serviceTimerExpiry() {
                // Find the next road based on clockwise direction
                s.ReturnVehicleDensity();
                TimeValueForDensity = this. CalculateTickDownTimerValue (vehicleDensity);
                Timer.InitializeTickDownTimer(TimeValueForDensity);
                // Turn on the green light for that road
        }

Behavior: serviceEmergenyRequest() {
                // Turn all traffic lights to RED
        }
```