PROJECT-Smart Traffic Signal Optimization

Scenario: You are part of a team working on an initiative to optimize traffic signal management in a busy city to reduce congestion and improve traffic flow efficiency using smart technologies.

Tasks:

1. **Data Collection and Modeling:**

   To collect real-time traffic data from sensors at various intersections, we define a data structure that includes entities such as **Sensor**, **Intersection**, **TrafficData**, and **TrafficSignal**. Each entity has specific attributes necessary for capturing and organizing the traffic information:

- **Sensor**: Contains attributes like **sensorID**, **intersectionID**, **sensorType**, **installationDate**, and **status**. This structure helps identify and manage sensors based on their type and installation details.
- **Intersection**: Includes **intersectionID**, **location**, and **description** to identify and describe each intersection's location and characteristics.
- **TrafficData**: Holds data such as **dataID**, **sensorID**, **timestamp**, **vehicleCount**, **averageSpeed**, and **vehicleTypeDistribution**. This structure records the traffic data collected by sensors at specific times.
- **TrafficSignal**: Stores **signalID**, **intersectionID**, **status**, and **signalTiming** to manage traffic signals' operational status and timing settings.

  Here's a concise Java representation of these entities:

CODE:

```
import java.util.*;

          class Intersection {
          private int id;
          private String location;
          private String description;
          public Intersection(int id, String location, String description) {
          this.id = id;
          this.location = location;
          this.description = description;
          }

          }
         class Sensor {
          private int id;
          private int intersectionId;
          private String type;
          private String installationDate;
          private String status;
          public Sensor(int id, int intersectionId, String type, String installationDate, String
          status) {
          this.id = id;
          this.intersectionId = intersectionId;
          this.type = type;
          this.installationDate = installationDate;
          this.status = status;
          }
          }
          class TrafficData {
```

```
  private int id;
  private int sensorId;
  private Date timestamp;
  private int vehicleCount;
  private double averageSpeed;
  private String vehicleTypeDistribution;

  public TrafficData(int id, int sensorId, Date timestamp, int vehicleCount, double
  averageSpeed, String vehicleTypeDistribution) {
  this.id = id;
  this.sensorId = sensorId;
  this.timestamp = timestamp;
  this.vehicleCount = vehicleCount;
  this.averageSpeed = averageSpeed;
  this.vehicleTypeDistribution = vehicleTypeDistribution;
  }
  }
  class TrafficSignal {
  private int id;
  private int intersectionId;
  private String status;
  private String signalTiming;  // e.g., "Green: 30s, Yellow: 5s, Red: 45s"
  public TrafficSignal(int id, int intersectionId, String status, String signalTiming) {
  this.id = id;
  this.intersectionId = intersectionId;
  this.status = status;
  this.signalTiming = signalTiming;
  }
}
```

2.Algorithm Design:

Sol: Dynamic Signal Timing Algorithm

We develop algorithms to dynamically adjust traffic signal timings based on real-time traffic conditions. Our algorithm considers factors such as traffic density, vehicle queues, peak hours, and pedestrian crossings. A sample approach might involve:

- **Data Analysis**: Collect and analyze real-time data to determine traffic density and vehicle speeds at different times.
- **Signal Timing Adjustment**: Implement logic to adjust signal timings dynamically. For instance, increase green light duration if vehicle density is high or during peak hours.
- **Pedestrian Consideration**: Integrate pedestrian crossing signals into the timing algorithm to ensure safety and efficiency.

Consider using a traffic flow algorithm like the Dijkstra’s Algorithm or Greedy
Algorithm for dynamic adjustments.

CODE:

```
import java.util.*;
class TrafficSignalOptimizer {
private Map<Integer, List<TrafficData>> trafficDataMap = new HashMap<>();
private Map<Integer, TrafficSignal> signals = new HashMap<>();
public void collectTrafficData(int sensorId, TrafficData data) {
trafficDataMap.computeIfAbsent(sensorId, k -> new ArrayList<>()).add(data);
}
public void optimizeSignalTiming() {
for (Map.Entry<Integer, List<TrafficData>> entry : trafficDataMap.entrySet()) {
int sensorId = entry.getKey();
List<TrafficData> dataList = entry.getValue();
TrafficSignal signal = signals.get(sensorId);
int totalVehicleCount =
dataList.stream().mapToInt(TrafficData::getVehicleCount).sum();
double averageSpeed =
dataList.stream().mapToDouble(TrafficData::getAverageSpeed).average().orElse(0);
String newTiming = calculateSignalTiming(totalVehicleCount, averageSpeed);
signal.setSignalTiming(newTiming);
}
}

private String calculateSignalTiming(int vehicleCount, double averageSpeed) {
if (vehicleCount > 100 || averageSpeed < 20) {
return "Green: 40s, Yellow: 5s, Red: 50s";
} else if (vehicleCount > 50) {
return "Green: 30s, Yellow: 5s, Red: 45s";
} else {
return "Green: 20s, Yellow: 5s, Red: 40s";
}
}
public void setTrafficSignals(Map<Integer, TrafficSignal> signals) {
this.signals = signals;
}
}
```

3.Implementation:

Java Application Setup

Created a simple Java application to integrate with traffic sensors and control signals.

CODE:

```
import java.util.*;
public class TrafficManagementApp {
public static void main(String[] args) {
TrafficSignalOptimizer optimizer = new TrafficSignalOptimizer();
optimizer.collectTrafficData(101, new TrafficData(1001, 101, new Date(), 120, 35.5,
"Cars: 80%, Trucks: 15%, Buses: 5%"));
optimizer.collectTrafficData(102, new TrafficData(1002, 102, new Date(), 80, 25.0,
"Cars: 60%, Trucks: 20%, Buses: 20%"));
Map<Integer, TrafficSignal> signals = new HashMap<>();
signals.put(101, new TrafficSignal(201, 1, "Operational", "Green: 30s, Yellow: 5s,
Red: 45s"));
signals.put(102, new TrafficSignal(202, 2, "Operational", "Green: 25s, Yellow: 5s,
Red: 40s"));
optimizer.setTrafficSignals(signals);
optimizer.optimizeSignalTiming();
signals.values().forEach(signal -> System.out.println("Signal " + signal.getId() + " at
Intersection " + signal.getIntersectionId() + " Timing: " + signal.getSignalTiming()));
}
}
```

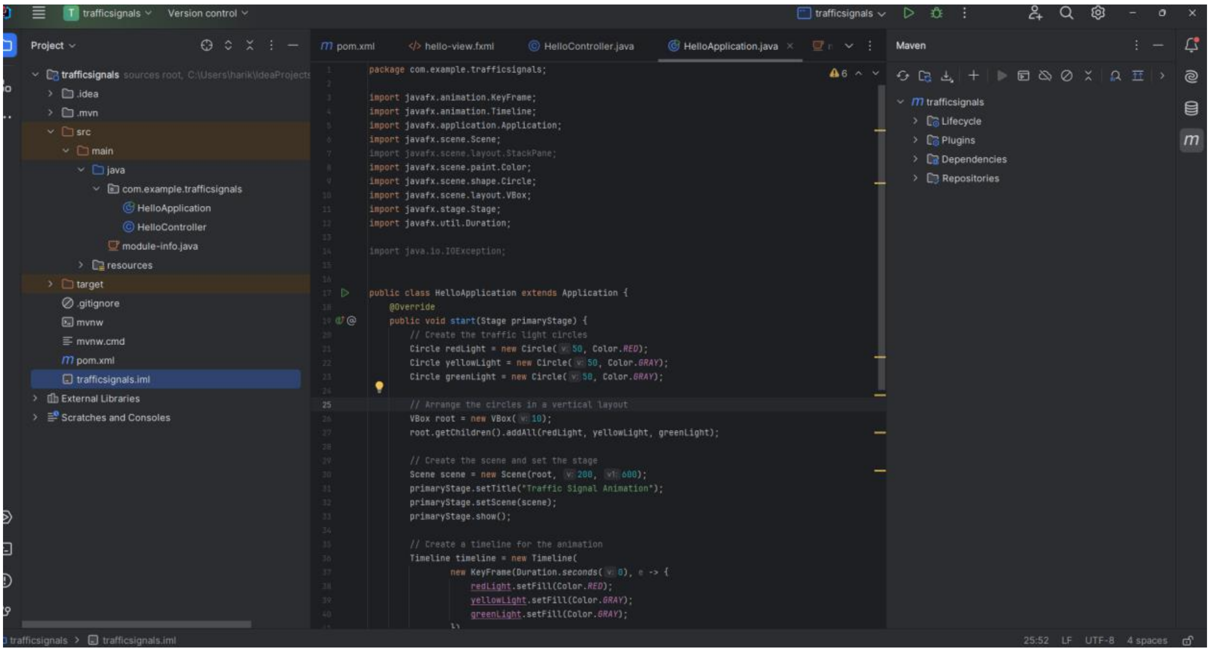
4.Visualization and Reporting:

To monitor traffic conditions and signal timings in real-time, we develop visualizations using JavaFX and Intellij These tools help traffic managers visualize data and generate reports on traffic flow improvements and congestion reduction. Here’s an example using JavaFX and Intellij:

JAVAFX CODE:

```
package com.example.trafficsignalss;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.util.Duration;
import java.io.IOException;
public class HelloApplication extends Application {
@Override
public void start(Stage primaryStage) {
// Create the traffic light circles
Circle redLight = new Circle(50, Color.RED);
Circle yellowLight = new Circle(50, Color.GRAY);
Circle greenLight = new Circle(50, Color.GRAY);
VBox root = new VBox(10);
root.getChildren().addAll(redLight, yellowLight, greenLight);
Scene scene = new Scene(root, 200, 600);
primaryStage.setTitle("Traffic Signal Animation");
primaryStage.setScene(scene);
primaryStage.show();
Timeline timeline = new Timeline(
new KeyFrame(Duration.seconds(0), e -> {
redLight.setFill(Color.RED);
yellowLight.setFill(Color.GRAY);
greenLight.setFill(Color.GRAY);
}),
new KeyFrame(Duration.seconds(3), e -> {
redLight.setFill(Color.GRAY);
yellowLight.setFill(Color.YELLOW);
greenLight.setFill(Color.GRAY);
}),
new KeyFrame(Duration.seconds(6), e -> {
redLight.setFill(Color.GRAY);
yellowLight.setFill(Color.GRAY);
greenLight.setFill(Color.GREEN);
}),
new KeyFrame(Duration.seconds(9), e -> {
redLight.setFill(Color.RED);
yellowLight.setFill(Color.GRAY);
greenLight.setFill(Color.GRAY);
})
);
timeline.setCycleCount(Timeline.INDEFINITE);
timeline.play();
}

public static void main(String[] args) {
launch(args);
}
}
```



```java
package com.example.trafficsignals;

import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.io.IOException;


public class HelloApplication extends Application {
    @Override
    public void start(Stage primaryStage) {
        // Create the traffic light circles
        Circle redLight = new Circle( v: 50, Color.RED);
        Circle yellowLight = new Circle( v: 50, Color.GRAY);
        Circle greenLight = new Circle( v: 50, Color.GRAY);

        // Arrange the circles in a vertical layout
        VBox root = new VBox( v: 10);
        root.getChildren().addAll(redLight, yellowLight, greenLight);

        // Create the scene and set the stage
        Scene scene = new Scene(root,  v: 200,  v1: 600);
        primaryStage.setTitle("Traffic Signal Animation");
        primaryStage.setScene(scene);
        primaryStage.show();

        // Create a timeline for the animation
        Timeline timeline = new Timeline(
                new KeyFrame(Duration.seconds( v: 0), e -> {
                    redLight.setFill(Color.RED);
                    yellowLight.setFill(Color.GRAY);
                    greenLight.setFill(Color.GRAY);
                })
```

5.User Interaction:

We design a user interface for traffic managers to monitor and adjust signal timings, along with a dashboard for city officials to view performance metrics and historical data. Using JavaFX, we create an intuitive and interactive interface:

JAVAFX OUTPUT .mp4

These tasks collectively form a comprehensive approach to implementing a smart traffic signal optimization system, enhancing urban traffic management through real-time data analysis, intelligent signal control, and user-friendly interfaces.