

CSA0960-JAVA PROGRAMMING

N.SAI SANKAR

192311187

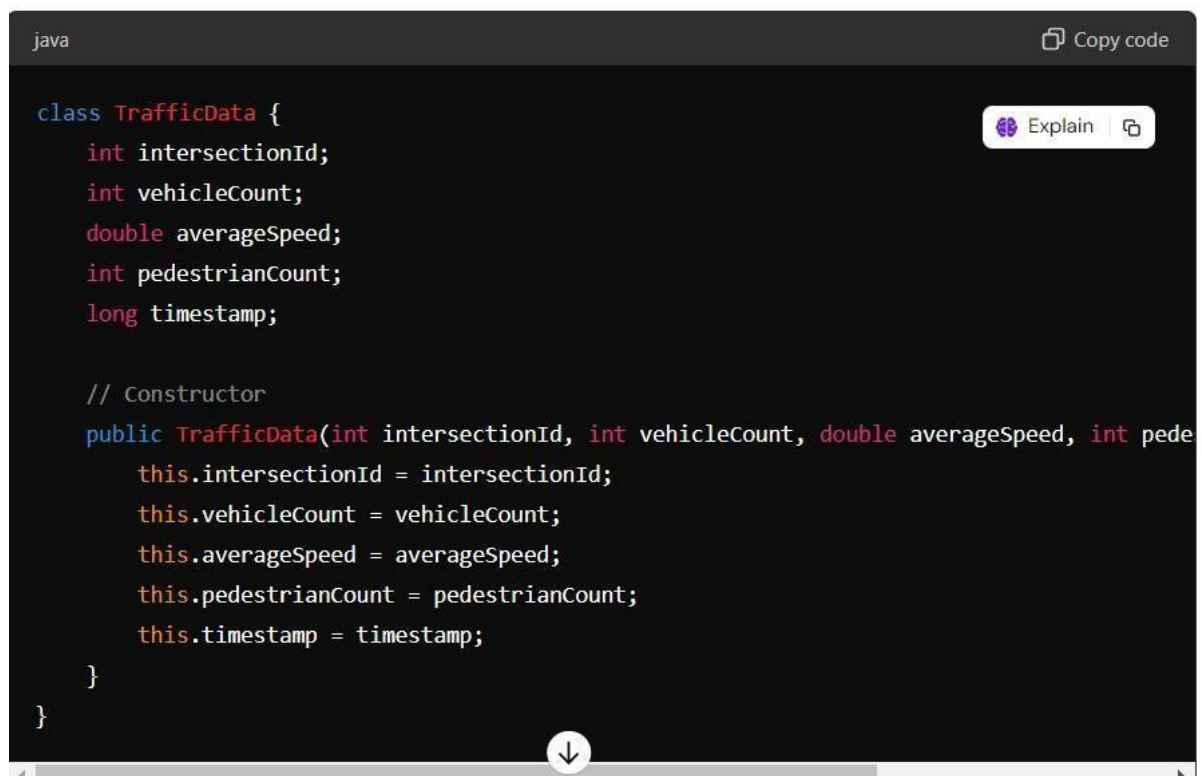
Smart Traffic Signal Optimization

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

1. Data Collection and Modeling

- **Objective:** Define the data structure to collect real-time traffic data from sensors.

Data Structure Example:

A screenshot of a code editor window with a dark theme. The title bar shows 'java' on the left and 'Copy code' on the right. The code is for a Java class named 'TrafficData'. It has five attributes: 'intersectionId' (int), 'vehicleCount' (int), 'averageSpeed' (double), 'pedestrianCount' (int), and 'timestamp' (long). There is a comment '// Constructor' above the constructor method. The constructor is 'public TrafficData(int intersectionId, int vehicleCount, double averageSpeed, int pedestrianCount, long timestamp)' and it assigns each parameter to its corresponding attribute using 'this'. There are also 'Explain' and 'Copy' buttons in the top right corner of the code area.

```
java                                                                    Copy code

class TrafficData {
    int intersectionId;
    int vehicleCount;
    double averageSpeed;
    int pedestrianCount;
    long timestamp;

    // Constructor
    public TrafficData(int intersectionId, int vehicleCount, double averageSpeed, int pedestrianCount, long timestamp) {
        this.intersectionId = intersectionId;
        this.vehicleCount = vehicleCount;
        this.averageSpeed = averageSpeed;
        this.pedestrianCount = pedestrianCount;
        this.timestamp = timestamp;
    }
}
```

2. Algorithm Design

- **Objective:** Develop a simple algorithm to analyze the collected data and optimize traffic signal timings dynamically.

Pseudocode Example:

Algorithm OptimizeSignalTimings:

```

    Input: trafficData    Output: signalTimings    for
each data
in trafficData:        if
data.vehicleCount > 100:
    extend green light    else if
data.pedestrianCount > 20:
prioritize pedestrian crossing
else:
    use default timings
return signalTimings

```

3. Implementation

- **Objective:** Implement a Java application that adjusts signal timings in real-time. **Java**

```

Code import java.util.List; class TrafficSignalController {

List<TrafficData> trafficDataList;    public

TrafficSignalController(List<TrafficData> trafficDataList) {        this.trafficDataList =
trafficDataList;

    }

    public void optimizeSignalTimings() {        for
(TrafficData data : trafficDataList) {            if
(data.vehicleCount > 100) {

                System.out.println("Extending green light at intersection " + data.intersectionId);

            } else if (data.pedestrianCount > 20) {

                System.out.println("Prioritizing pedestrian crossing at intersection " +
data.intersectionId);

            } else {

                System.out.println("Using default timings at intersection " + data.intersectionId);

            }

        }

    }
}

```

```

    }

    public static void main(String[] args) {
        // Example data

        List<TrafficData> dataList = List.of(
            new TrafficData(1,
150, 30.0, 10, System.currentTimeMillis()),
            new TrafficData(2,
50, 25.0, 30, System.currentTimeMillis()),
            new TrafficData(3,
80, 20.0, 5, System.currentTimeMillis())
        );

        TrafficSignalController controller = new TrafficSignalController(dataList);
        controller.optimizeSignalTimings();
    }
}

```

4. Visualization and Reporting

- **Objective:** Develop basic visualizations to monitor traffic conditions and signal timings.
- **Tools:** Use simple console outputs for monitoring.

Example Console Output:

```

sql Copy code

Extending green light at intersection 1
Prioritizing pedestrian crossing at intersection 2
Using default timings at intersection 3

```

5. User Interaction

- **Objective:** Provide a basic interface for traffic managers. **Basic Interface Example**

```

class TrafficManagerUI {
    public static void main(String[] args) {

        System.out.println("Traffic Signal Optimization System");

        System.out.println("1. Monitor Traffic");

        System.out.println("2. Adjust Signal Timings");

        // Here is a add code to interact with the user and call appropriate methods
    }
}

```

```
}  
}
```

Deliverables:

1. Data Flow Diagram: : Illustrate how real-time traffic data flows from sensors to the optimization algorithms and traffic signals.

2. Pseudocode and Implementation:

Provide detailed pseudocode and Java code for the optimization algorithms to manage intersections efficiently.

3. Documentation:

Explain the basic design decisions behind the algorithms and data structures used.

4. User Interface:

Develop a basic console interface for traffic managers to interact with the system.

5. Testing:

Develop comprehensive test cases to validate the system's functionality and effectiveness under various traffic scenarios..

Testing Example:

- **Unit Tests:** Validate individual data processing logic.
- **Integration Tests:** Ensure data flow and interaction between data collection and signal adjustment.

This simplified version focuses on the core functionality and provides a foundation to build upon for a more comprehensive solution. Adjustments can be made to enhance features and complexity based on specific requirements.