**7.a Smart City Traffic Management System**

**Code :**

import java.util.\*;

// TrafficSignal class (Aggregation: Independent operation)

class TrafficSignal {

private String signalId;

private String status; // Red, Yellow, Green

public TrafficSignal(String signalId) {

this.signalId = signalId;

this.status = "Red"; // Default status

}

public void changeSignal() {

String[] signals = {"Red", "Yellow", "Green"};

this.status = signals[new Random().nextInt(signals.length)];

}

public String getStatus() {

return status;

}

public String getSignalId() {

return signalId;

}

}

// Junction class (Composition: Contains multiple TrafficSignals)

class Junction {

private String junctionId;

private List<TrafficSignal> signals;

public Junction(String junctionId, int numSignals) {

this.junctionId = junctionId;

this.signals = new ArrayList<>();

for (int i = 1; i <= numSignals; i++) {

this.signals.add(new TrafficSignal(junctionId + "-Signal" + i));

}

}

public void updateSignals() {

for (TrafficSignal signal : signals) {

signal.changeSignal();

}

}

public void displayTrafficSignalStatus() {

System.out.println("Junction: " + junctionId);

for (TrafficSignal signal : signals) {

System.out.println(" Signal " + signal.getSignalId() + " -> " + signal.getStatus());

} }

}

// City class (Contains multiple Junctions)

class City {

private String cityName;

private List<Junction> junctions;

public City(String cityName) {

this.cityName = cityName;

this.junctions = new ArrayList<>();

}

public void addJunction(Junction junction) {

junctions.add(junction);

}

public void updateCityTraffic() {

for (Junction junction : junctions) {

junction.updateSignals();

}

}

public void displayCityTrafficStatus() {

System.out.println("City: " + cityName);

for (Junction junction : junctions) {

junction.displayTrafficSignalStatus();

} }

}

public class SmartCityTrafficManagement {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter city name: ");

String cityName = scanner.nextLine();

City city = new City(cityName);

System.out.print("Enter number of junctions: ");

int numJunctions = scanner.nextInt();

for (int i = 1; i <= numJunctions; i++) {

System.out.print("Enter name for Junction " + i + ": ");

String junctionName = scanner.next();

System.out.print("Enter number of signals for " + junctionName + ": ");

int numSignals = scanner.nextInt();

Junction junction = new Junction(junctionName, numSignals);

city.addJunction(junction);

}

System.out.println("\nInitial Traffic Status:");

city.displayCityTrafficStatus();

// Simulating real-time traffic signal changes

System.out.println("\nUpdating Traffic Signals...");

city.updateCityTraffic();

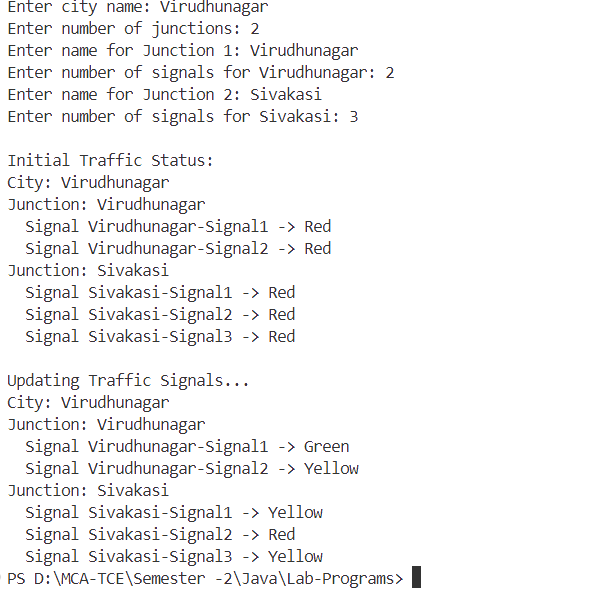
city.displayCityTrafficStatus();

scanner.close();

}

}

**Output :**

****

**7.b Smart Home System**

**Code :**

import java.util.\*;

// SmartDevice Class (Represents IoT-enabled devices)

class SmartDevice {

private String name;

private boolean isOn;

public SmartDevice(String name) {

this.name = name;

this.isOn = false; // Default state is OFF

}

public void toggle() {

isOn = !isOn;

System.out.println(name + " is now " + (isOn ? "ON" : "OFF"));

}

public String getStatus() {

return name + " - " + (isOn ? "ON" : "OFF");

}

public String getName() {

return name;

}

}

// Room Class (Contains multiple SmartDevices - Composition)

class Room {

private String name;

private List<SmartDevice> devices;

public Room(String name) {

this.name = name;

this.devices = new ArrayList<>();

}

public void addDevice(String deviceName) {

devices.add(new SmartDevice(deviceName));

}

public void toggleDevice(String deviceName) {

for (SmartDevice device : devices) {

if (device.getName().equalsIgnoreCase(deviceName)) {

device.toggle();

return;

}

}

System.out.println("Device not found in " + name);

}

public void displayDevices() {

System.out.println("Room: " + name);

for (SmartDevice device : devices) {

System.out.println(" " + device.getStatus());

}

}

public String getName() {

return name;

}

}

// SmartHome Class (Has multiple Rooms - Aggregation)

class SmartHome {

private List<Room> rooms;

public SmartHome() {

this.rooms = new ArrayList<>();

}

public void addRoom(String roomName) {

rooms.add(new Room(roomName));

}

public Room getRoom(String roomName) {

for (Room room : rooms) {

if (room.getName().equalsIgnoreCase(roomName)) {

return room;

}

}

return null;

}

public void displayHomeStatus() {

System.out.println("Smart Home Status:");

for (Room room : rooms) {

room.displayDevices();

}

}

}

public class SmartHomeSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

SmartHome smartHome = new SmartHome();

while (true) {

System.out.println("\n1. Add Room\n2. Add Device\n3. Toggle Device\n4. Show Status\n5. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter room name: ");

String roomName = scanner.nextLine();

smartHome.addRoom(roomName);

break;

case 2:

System.out.print("Enter room name: ");

roomName = scanner.nextLine();

Room room = smartHome.getRoom(roomName);

if (room != null) {

System.out.print("Enter device name: ");

String deviceName = scanner.nextLine();

room.addDevice(deviceName);

} else {

System.out.println("Room not found!");

}

break;

case 3:

System.out.print("Enter room name: ");

roomName = scanner.nextLine();

room = smartHome.getRoom(roomName);

if (room != null) {

System.out.print("Enter device name: ");

String deviceName = scanner.nextLine();

room.toggleDevice(deviceName);

} else {

System.out.println("Room not found!");

}

break;

case 4:

smartHome.displayHomeStatus();

break;

case 5:

System.out.println("Exiting Smart Home System.");

scanner.close();

return;

default:

System.out.println("Invalid choice. Try again.");

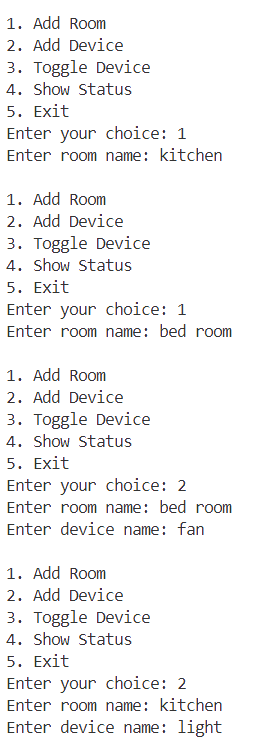
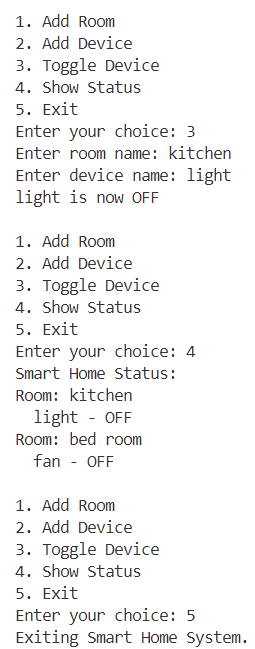
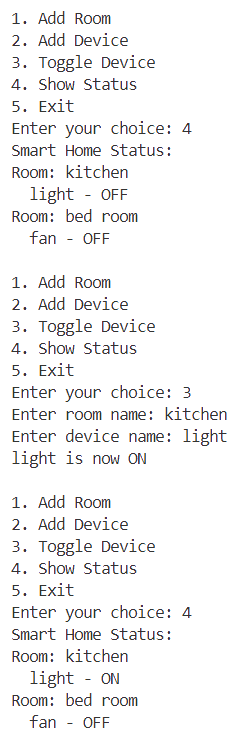
}

}

}

}

**Output :**

** **