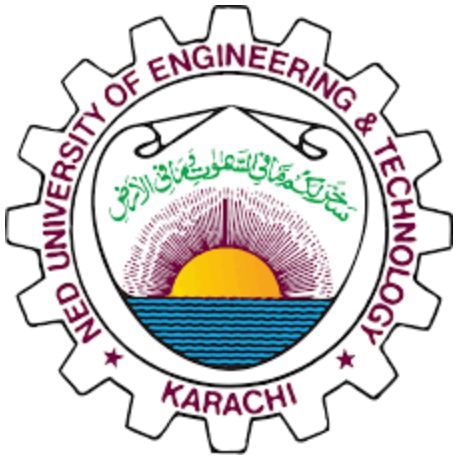


FORMAL METHOD IN SOFTWARE ENGINEERING

(SE-313)

Assignment

Traffic Light Controller

SYED AUN MUHAMMAD (SE-21036)

MOIZ NAVEED (SE-21048)

Department of Software Engineering
NED University of Engineering & Technology

FMS Assignment includes:

- *Problem Statement and Description*
- *Functional Requirement*
- *Design(4+1 view)*
- *VDM Specification*
- *JAVA Code*
- *Testing*
- *GitHub repository:*

1. PROBLEM STATEMENT :

The system manages traffic lights for four directions (NORTH, EAST, SOUTH, WEST) and controls the traffic flow. The program allows the user to set the traffic light for a specific direction to GREEN and specify the duration of the GREEN signal. The system then updates the traffic light states for all directions based on this input.

Problem Scope:

The main components and functionalities of the Traffic Controller System include:

- **TrafficLight Enumeration:**
  - Defines the three possible states of a traffic light: RED, YELLOW, GREEN.
- **Direction Enumeration:**
  - Represents the four cardinal directions: NORTH, EAST, SOUTH, WEST.
- **TrafficLightInfo Class:**
  - Stores the current state (RED, YELLOW, or GREEN) and the timer for a specific direction's traffic light.
- **TrafficControllerSystem Class:**
  - Manages the overall traffic control system.
  - Allows the user to input a direction and the duration for which its traffic light should be set to GREEN.
  - Updates the system state accordingly, adjusting timers for other directions.
  - Utilizes a scheduler to decrement timers for each direction at regular intervals.
  - Displays the updated state of the system after each adjustment.
- **ScheduledExecutorService:**
  - Used to schedule and execute the decrementTimers method at fixed intervals.

Functional Requirements (FR):

- **User Input:**
  - The system must prompt the user to input a direction and the desired duration for the GREEN signal.
  - Validate the input to ensure a valid direction and timer value within the specified range.
- **System State Update:**
  - Update the traffic light state and timers based on user input.
  - Adjust timers for other directions accordingly.
- **Timer Decrementing:**
  - Regularly decrement timers for each direction to simulate the countdown of traffic light signals.
  - Update the system state after each decrement.
- **Display System State:**
  - Provide a display of the current state of the system, showing the traffic light status and remaining timer for each direction.

Non-Functional Requirements (NFR):

- **Efficiency:**
  - The system should efficiently handle timer decrements and updates to provide a smooth simulation.
- **Usability:**
  - The user interface should be user-friendly and provide clear instructions for input.
- **Reliability:**
  - The system should handle user input errors gracefully, providing appropriate error messages.
- **Scheduling Accuracy:**
  - The scheduler should accurately execute the decrementTimers method at fixed intervals.

Constraints:

- **Timer Range:**
  - The timer values for GREEN lights must be within the range of MIN_GREEN_TIMER (1) and MAX_GREEN_TIMER (30).
- **Direction Input:**

    - The user must enter a valid cardinal direction (NORTH, EAST, SOUTH, WEST) when specifying the direction for a GREEN light.
- **Scheduler Interval:**
    - The scheduler should run the decrementTimers method at fixed intervals of 1 second.
- **Concurrency:**
    - The system should handle concurrency concerns, ensuring that the scheduler and user input do not interfere with each other.
- **Graceful Shutdown:**
    - The system should gracefully shut down the scheduler when the program exits.

This Traffic Controller System provides a basic simulation and can be extended for more complex scenarios or integrated into a larger traffic management system.

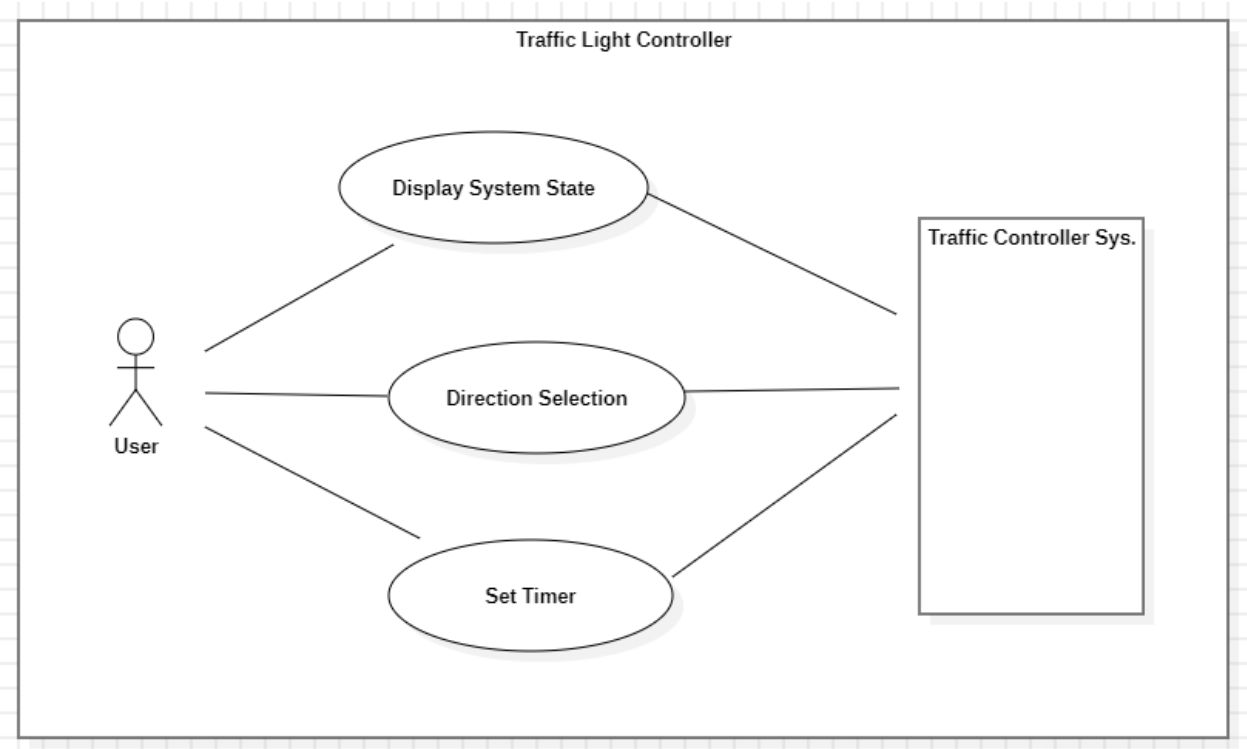
DESIGNS

4+1 View:

- *Logical View*
- *Process View*
- *Physical View*
- *Development View*
- *+1 Scenarios and Usecase View*

+1 SCENARIOS VIEW:

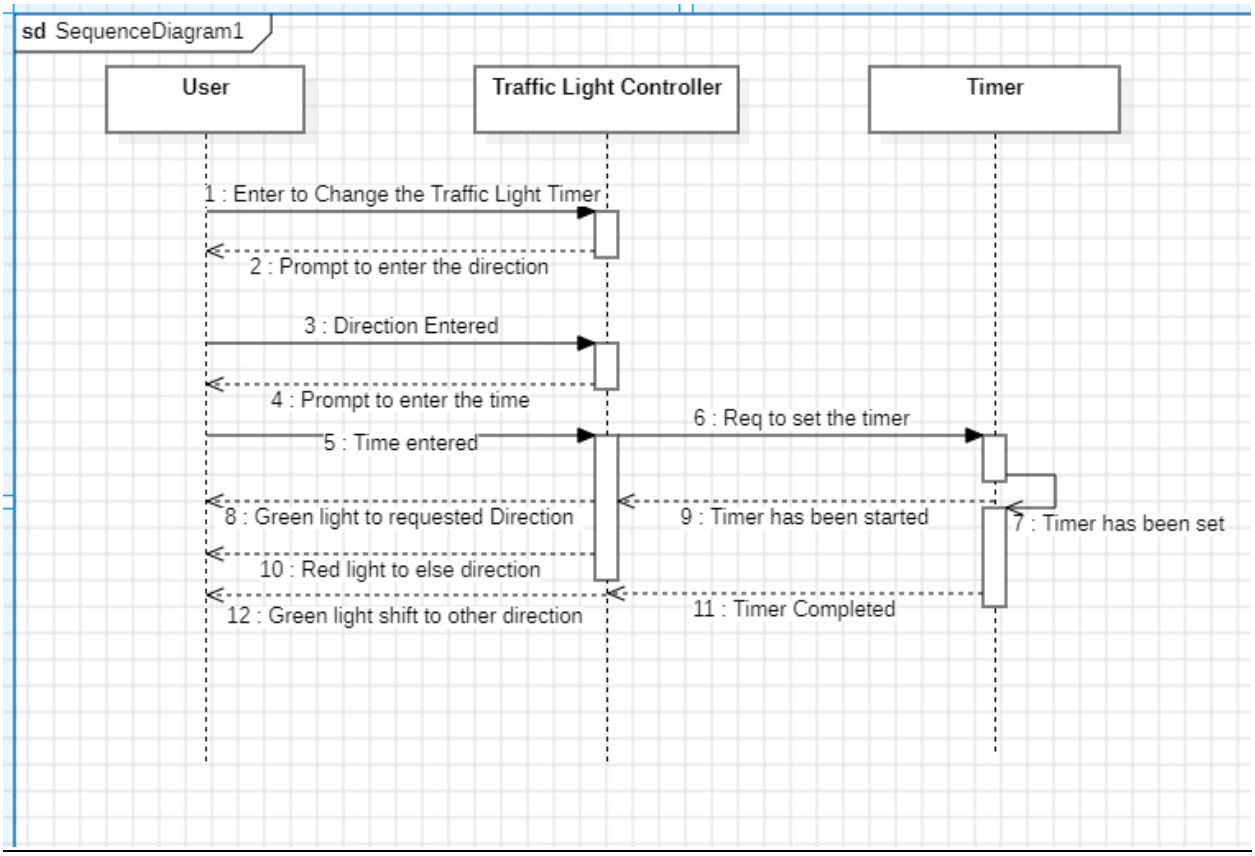
- **Usecase Diagram:**



PROCESS VIEW:

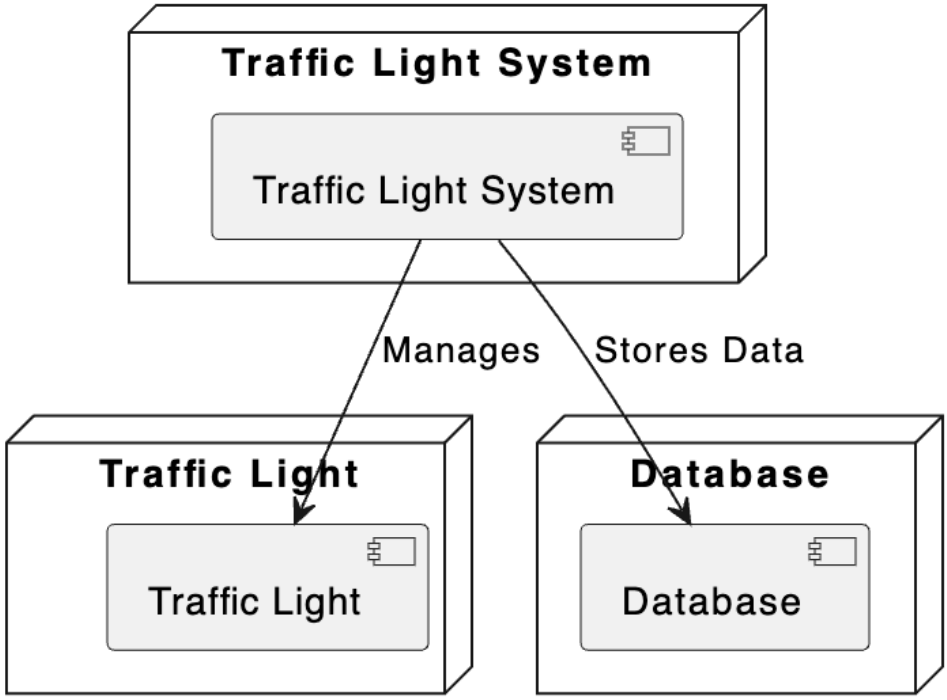
- **Sequence Diagram:**

USER:



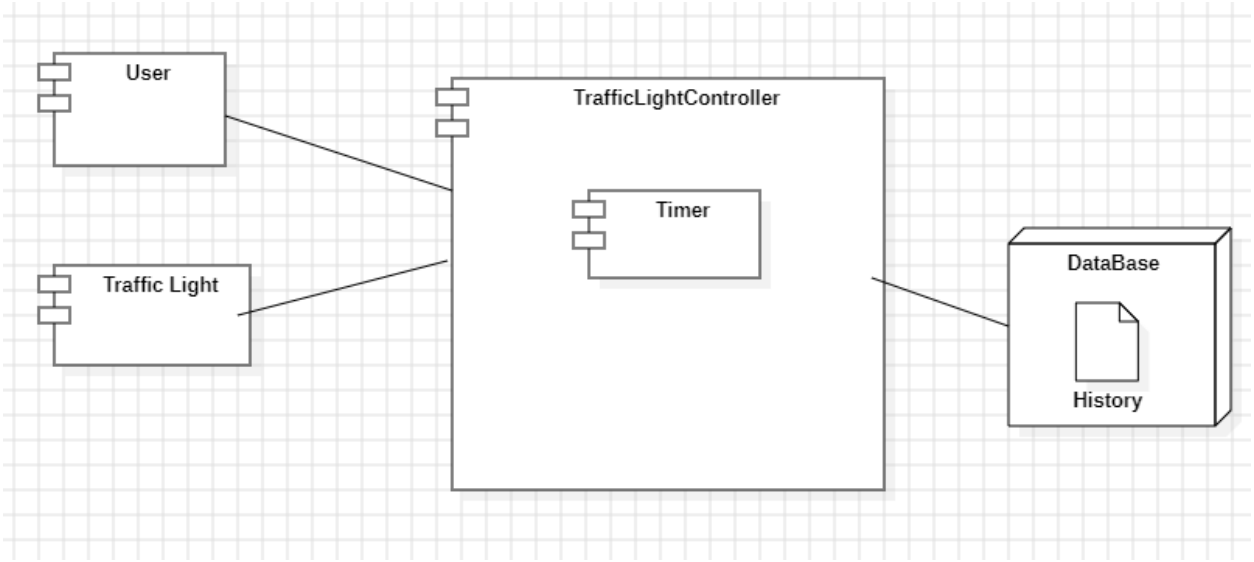
PHYSICAL VIEW:

- **Deployment Diagram:**



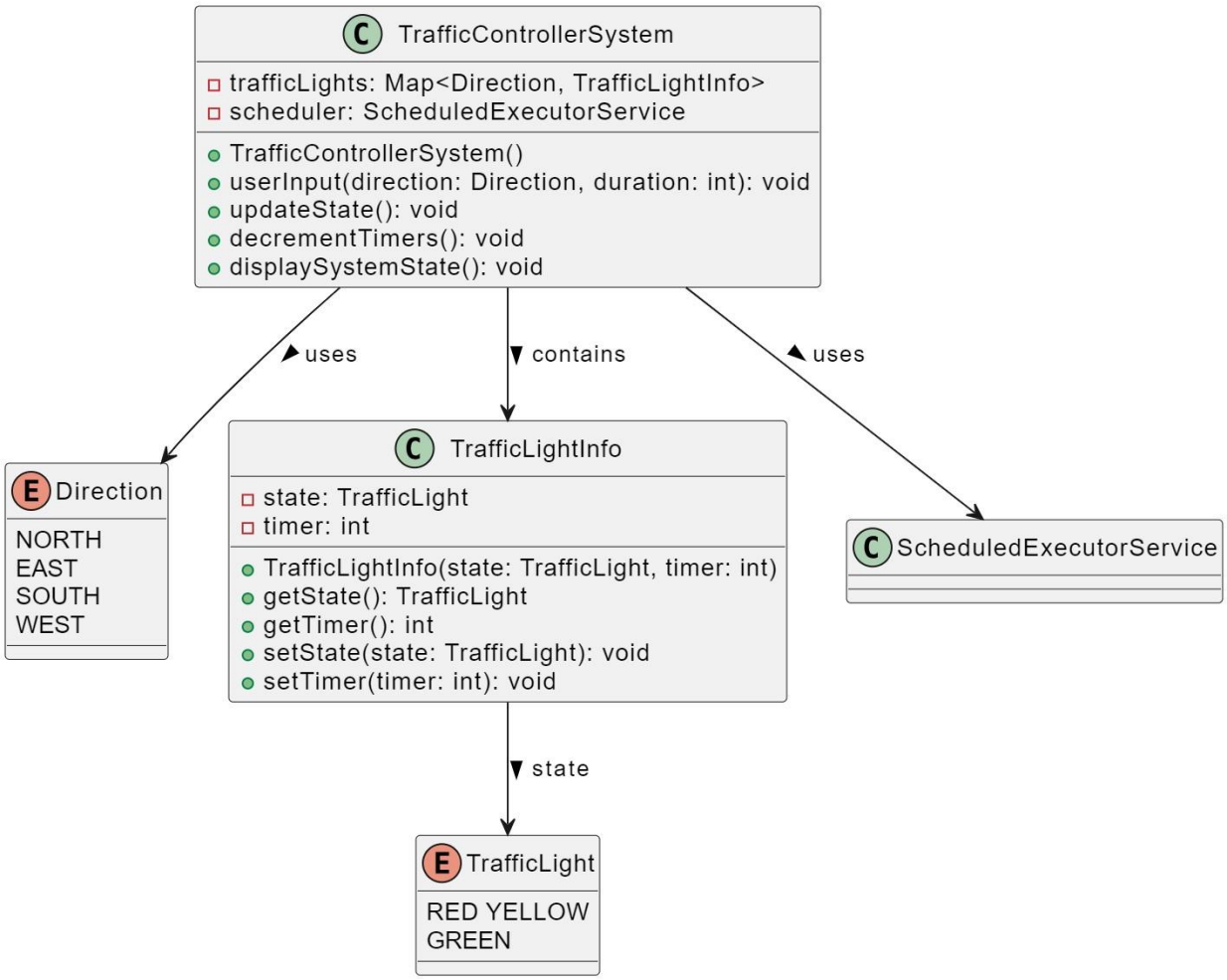
DEVELOPMENT VIEW:

- **Component Diagram:**



LOGICAL VIEW:

- **Class diagram:**



VDM SPECIFICATIONS

```
types
  TrafficLight = <RED> | <YELLOW> | <GREEN>;
  Direction = <NORTH> | <EAST> | <SOUTH> | <WEST>;

  TrafficLightInfo :: light: TrafficLight
                      timer: nat;

  TrafficControllerSystem
  :: directionTrafficMap: map Direction to TrafficLightInfo
     inv mk_ TrafficControllerSystem(dtm) ==
       forall direction in set dom dtm &
         let light = dtm(direction).light,
             timer = dtm(direction).timer
         in
           (light = <RED> and timer = 0) or
           (light = <YELLOW> and timer = 0) or
           (light = <GREEN> and timer = 0) or
           (light = <RED> and timer > 0);

instance variables
  public dtm: map Direction to TrafficLightInfo;

operations
  public Initialize: () ==> TrafficControllerSystem
  Initialize() == return mk_ TrafficControllerSystem({});

  public SetTrafficLight: Direction * TrafficLight * nat ==> TrafficControllerSystem
  SetTrafficLight(direction, light, timer) ==
    let
      currentInfo = dtm(direction),
      updatedInfo = mk_TrafficLightInfo(light, timer)
    in
      dtm := dtm ++ { direction |-> updatedInfo } munion
```

```
        { d |-> mk_TrafficLightInfo(<RED>, currentInfo.timer + timer + 3) | d in set dom dtm \
{ direction } };

  public DecrementTimers: () ==> TrafficControllerSystem
  DecrementTimers() ==
    dtm := { d |-> DecrementTimer(d, dtm(d)) | d in set dom dtm };

  public DecrementTimer: Direction * TrafficLightInfo ==> TrafficLightInfo
  DecrementTimer(direction, info) ==
    if info.light = <RED> and info.timer = 0
    then mk_TrafficLightInfo(<YELLOW>, 1)
    elseif info.light = <YELLOW> and info.timer = 0
    then mk_TrafficLightInfo(<GREEN>, 27)
    elseif info.light = <GREEN> and info.timer = 0
    then mk_TrafficLightInfo(<RED>, 99)
    elseif info.light = <RED> and info.timer = 0
    then mk_TrafficLightInfo(<GREEN>, 30)
    else mk_TrafficLightInfo(info.light, info.timer - 1)
    end if;

  public GetSystemState: () ==> seq of seq of (Direction * TrafficLight * nat)
  GetSystemState() ==
    return [[d, dtm(d).light, dtm(d).timer | d in set dom dtm]];

end TrafficControllerSystem
```

JAVA CODE

TrafficLightController.java:

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

public class TrafficControllerSystem {

    // Enum for traffic lights
    public enum TrafficLight {
        RED, YELLOW, GREEN
    }

    // Enum for directions
    public enum Direction {
        NORTH, EAST, SOUTH, WEST
    }

    // Constants for timer range
    public static final int MIN_GREEN_TIMER = 1;
    public static final int MAX_GREEN_TIMER = 30;

    // Map to associate each direction with traffic light status and timer
    private static final Direction[] ALL_DIRECTIONS = Direction.values();
    private static final Map<Direction, TrafficLightInfo> directionTrafficMap = new
HashMap<>();

    // TrafficLightInfo class to store light status and timer
    public static class TrafficLightInfo {
        private TrafficLight light;
        private int timer;

        public TrafficLightInfo(TrafficLight light, int timer) {
            this.light = light;
            this.timer = timer;
        }
```

```
    public TrafficLight getLight() {
        return light;
    }

    public int getTimer() {
        return timer;
    }

    public void setLight(TrafficLight light) {
        this.light = light;
    }

    public void setTimer(int timer) {
        this.timer = timer;
    }
}

// Function to print the state of the system
private static void printSystemState() {
    System.out.println("The Updated State of the System is as follows");
    for (Direction direction : Direction.values()) {
        TrafficLightInfo info = directionTrafficMap.getOrDefault(direction,
                new TrafficLightInfo(TrafficLight.RED, 0));
        System.out.printf("%-6s %-10s %-5d%n", direction, info.getLight(), info.getTimer());
    }
}

public static void decrementTimers() {
    for (Direction direction : ALL_DIRECTIONS) {
        TrafficLightInfo info = directionTrafficMap.getOrDefault(direction,
                new TrafficLightInfo(TrafficLight.RED, 0));
        int currentTimer = info.getTimer();

        if (info.getLight() == TrafficLight.RED && currentTimer == 0) {
            // If green light timer becomes zero, change light to yellow and set timer to 1
            info.setLight(TrafficLight.YELLOW);
            info.setTimer(1);
        } else if (info.getLight() == TrafficLight.YELLOW && currentTimer == 0) {
            // If yellow light timer becomes zero, change light to red and set timer to 99
            info.setLight(TrafficLight.GREEN);
            info.setTimer(27);
        }
         else if (info.getLight() == TrafficLight.GREEN && currentTimer == 0) {
```

```
            // If yellow light timer becomes zero, change light to red and set timer to 99
            info.setLight(TrafficLight.RED);
            info.setTimer(99);
        } else if (info.getLight() == TrafficLight.RED && currentTimer == 0) {
            // If red light timer becomes zero, change light to green and set timer to 30
            info.setLight(TrafficLight.GREEN);
            info.setTimer(30);
        } else if (currentTimer > 0) {
            // Decrement the timer if it's greater than 0
            info.setTimer(currentTimer - 1);
        }

        directionTrafficMap.put(direction, info);
    }
    printSystemState();
}



public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    do {
        System.out.println("The Initial State of the System is as follows");
        printSystemState();

        System.out.println("\nEnter new information:");

        Direction inputDirection;
        int inputTimer;

        // Loop until valid direction is entered
        while (true) {
            System.out.print("Enter direction (NORTH, EAST, SOUTH, WEST) to Set it to
Green: ");
            String directionInput = scanner.nextLine().toUpperCase();
            try {
                inputDirection = Direction.valueOf(directionInput);
                break; // Exit the loop if the direction is valid
            } catch (IllegalArgumentException e) {
                System.out.println("Invalid direction. Please enter a valid direction.");
            }
        }
```

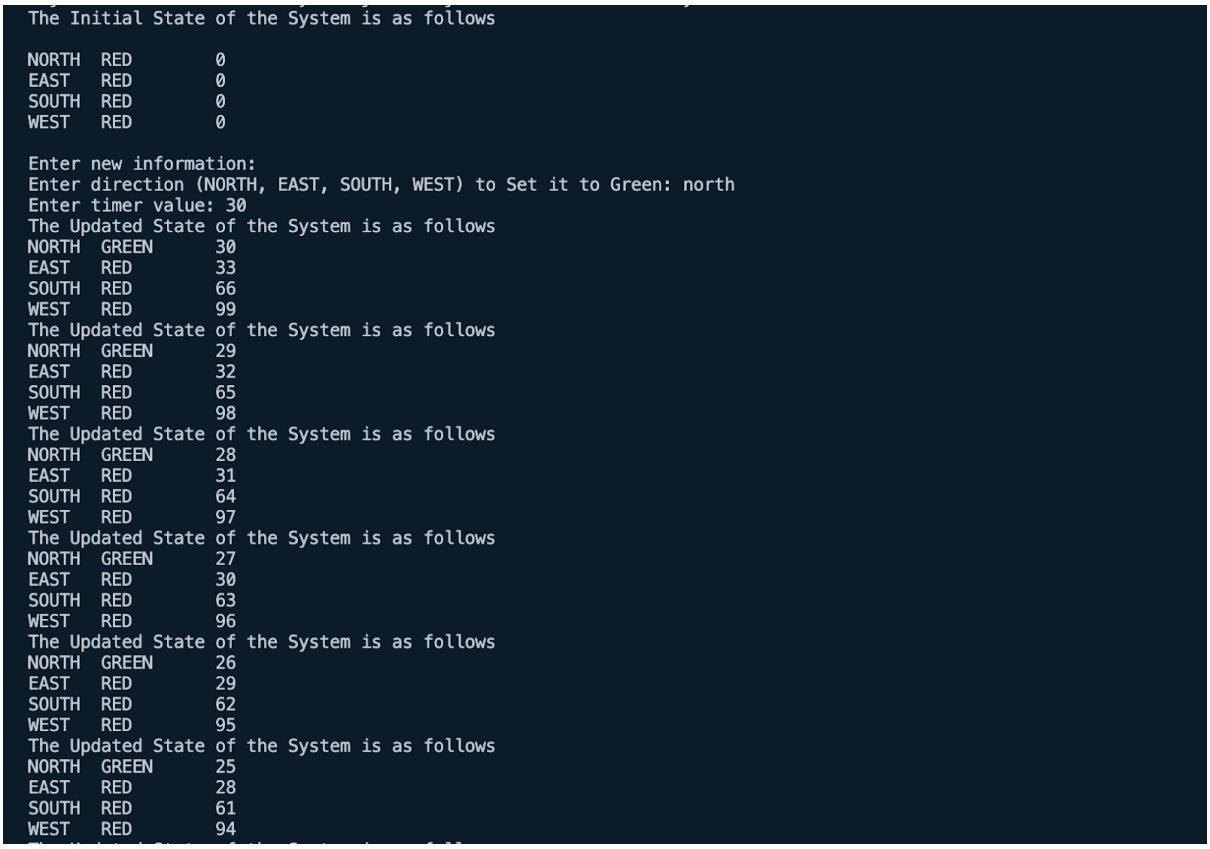
```
        // Loop until valid timer is entered
        while (true) {
            System.out.print("Enter timer value: ");
            try {
                inputTimer = Integer.parseInt(scanner.nextLine());
                if (inputTimer >= MIN_GREEN_TIMER && inputTimer <=
MAX_GREEN_TIMER) {
                    break; // Exit the loop if the timer is valid
                } else {
                    System.out.println(
                            "Timer value should be between " + MIN_GREEN_TIMER + " and " +
MAX_GREEN_TIMER);
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid timer format. Please enter a valid integer.");
            }
        }

        // Clear the map before updating it
        directionTrafficMap.clear();

        // Update the traffic light information based on user input
        TrafficLightInfo updatedInfo = new TrafficLightInfo(TrafficLight.GREEN, inputTimer);
        directionTrafficMap.put(inputDirection, updatedInfo);

        // Adjust timers for other directions
        int timerValue = inputTimer + 3;
        for (Direction direction : ALL_DIRECTIONS) {
            if (!direction.equals(inputDirection)) {
                TrafficLightInfo info = directionTrafficMap.getOrDefault(direction,
                        new TrafficLightInfo(TrafficLight.RED, 0));
                info.setLight(TrafficLight.RED); // Set the light to RED for other directions
                info.setTimer(info.getTimer() + timerValue); // Increment the timer by the input
timer value
                directionTrafficMap.put(direction, info);
                timerValue += 33;
            }

        }
        // Call the print function to display the updated state of the system
        printSystemState();
        // After displaying the updated state of the system
```

```
            // decrementTimers();

            ScheduledExecutorService scheduler = Executors.newSingleThreadScheduledExecutor();

            // Schedule the decrementTimers task to run repeatedly with a delay of 1 second
            scheduler.scheduleAtFixedRate(TrafficControllerSystem::decrementTimers, 0, 1,
TimeUnit.SECONDS);

            // ... (rest of your main method)

            // Close the scheduler when the program exits
            Runtime.getRuntime().addShutdownHook(new Thread(scheduler::shutdown));

            // Ask the user if they want to set another direction
            System.out.print("\nDo you want to set another direction? (yes/no): ");
        } while ("yes".equalsIgnoreCase(scanner.nextLine()));

        scanner.close(); // Close the Scanner object when done
    }
}
```

OUTPUTS:

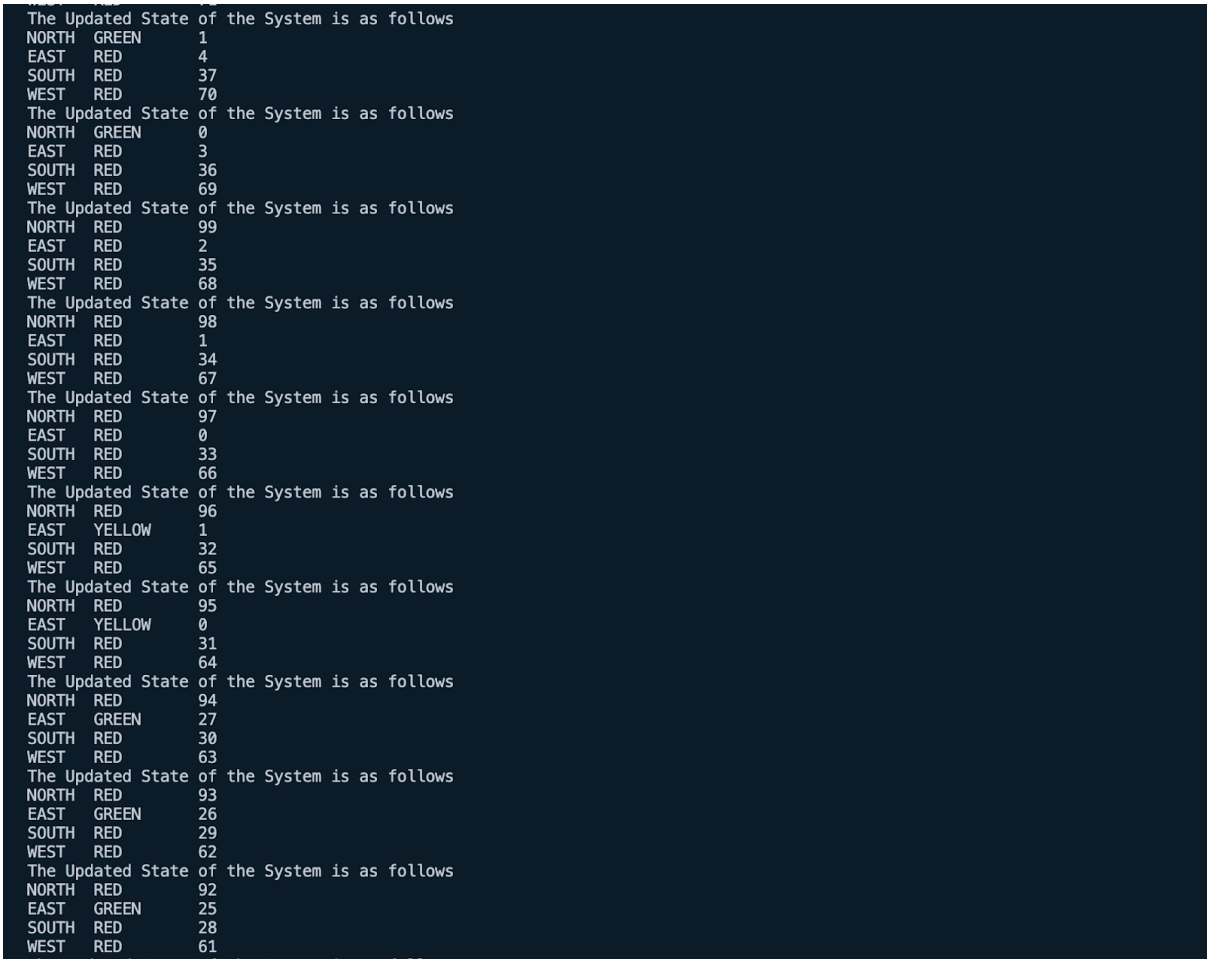
OUTPUT 1:

User input direction and time to initiate the system, system automatically changes its state after one second.



OUTPUT 2:

System state when the light of one direction gets changes when its timer become zero from green to red with the new timer as defined, and the light which is in state of red becomes yellow form 2 seconds then get green.



TESTING

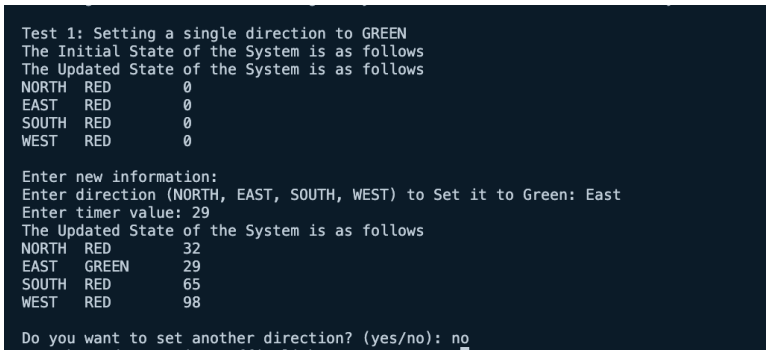
EventTicketingTester.java:

```
public class TrafficControllerSystemTester {
    public static void main(String[] args) {
        testSingleDirection(); // Test setting a single direction to GREEN
        testInvalidDirection(); // Test entering an invalid direction
        testInvalidTimer(); // Test entering an invalid timer value
        testMultipleDirections(); // Test setting multiple directions
    }
    private static void testSingleDirection() {
        System.out.println("\nTest 1: Setting a single direction to GREEN");
        TrafficControllerSystem.main(new String[]{"NORTH", "5"});
    }
    private static void testInvalidDirection() {
        System.out.println("\nTest 2: Entering an invalid direction");
        TrafficControllerSystem.main(new String[]{"INVALID_DIRECTION", "10"});
    }
    private static void testInvalidTimer() {
        System.out.println("\nTest 3: Entering an invalid timer value");
        TrafficControllerSystem.main(new String[]{"SOUTH", "40"});
    }
```

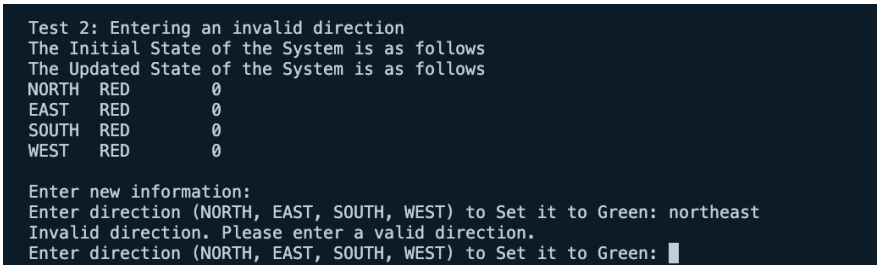
```
    private static void testMultipleDirections() {
        System.out.println("\nTest 4: Setting multiple directions");
        TrafficControllerSystem.main(new String[]{"EAST", "15", "yes", "WEST", "7", "no"});
    }
}
```

OUTPUT

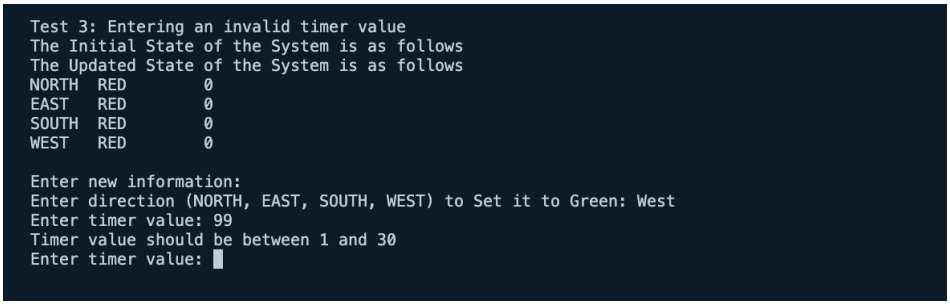
Test Case 1:



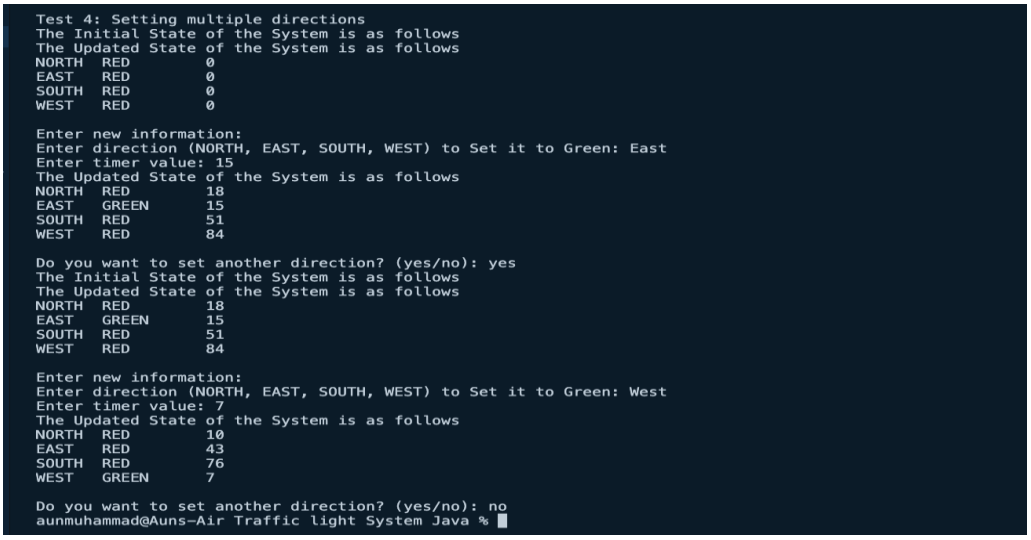
Test Case 2:



Test Case 3:



Test Case 4:



Repository:

https://github.com/AunMuhammad110/TrafficLightController