```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class TrafficLightController extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel lightPanel;
    private JButton startButton, stopButton, applyButton, peakHourButton, ambulanceButton,
nightModeButton;
    private JTextField redTimeField, yellowTimeField, greenTimeField;
    private int currentLight = 0; // 0: Red, 1: Yellow, 2: Green
    private final Color[] lightColors = {Color.RED, Color.YELLOW, Color.GREEN};
    private int[] lightTimings = {5000, 2000, 5000}; // Default timings in milliseconds
    private Timer lightCycleTimer, nightModeTimer;
    private boolean isPeakHour = false;
    private boolean isNightMode = false;

    public TrafficLightController() {
        setTitle("Traffic Light Controller");
        setSize(500, 750);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Light Panel
        lightPanel = new JPanel() {
            private static final long serialVersionUID = 1L;

            @Override
```

```
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        for (int i = 0; i < 3; i++) {
            g.setColor(i == currentLight ? lightColors[i] : Color.GRAY);
            g.fillOval(200, 50 + (150 * i), 100, 100);
        }
    }
};
lightPanel.setBackground(Color.BLACK);
add(lightPanel, BorderLayout.CENTER);

// Control Panel
JPanel controlPanel = new JPanel();
controlPanel.setLayout(new GridLayout(4, 2, 10, 10));

startButton = new JButton("Start");
stopButton = new JButton("Stop");
peakHourButton = new JButton("Toggle Peak Hour");
ambulanceButton = new JButton("Ambulance Priority");
nightModeButton = new JButton("Toggle Night Mode");
applyButton = new JButton("Apply Timings");

stopButton.setEnabled(false);

controlPanel.add(startButton);
controlPanel.add(stopButton);
controlPanel.add(peakHourButton);
controlPanel.add(ambulanceButton);
controlPanel.add(nightModeButton);
```

```
controlPanel.add(applyButton);

add(controlPanel, BorderLayout.SOUTH);

// Input Panel
JPanel inputPanel = new JPanel();
inputPanel.setLayout(new GridLayout(3, 2, 10, 10));
inputPanel.setBorder(BorderFactory.createTitledBorder("Set Light Timings (ms)"));

redTimeField = new JTextField(String.valueOf(lightTimings[0]));
yellowTimeField = new JTextField(String.valueOf(lightTimings[1]));
greenTimeField = new JTextField(String.valueOf(lightTimings[2]));

inputPanel.add(new JLabel("Red Light:"));
inputPanel.add(redTimeField);
inputPanel.add(new JLabel("Yellow Light:"));
inputPanel.add(yellowTimeField);
inputPanel.add(new JLabel("Green Light:"));
inputPanel.add(greenTimeField);

add(inputPanel, BorderLayout.EAST);

// Timer Setup
lightCycleTimer = new Timer(lightTimings[currentLight], e -> {
   currentLight = (currentLight + 1) % 3; // Cycle lights
   lightCycleTimer.setDelay(lightTimings[currentLight]); // Update delay
   lightPanel.repaint();
});
```

```
// Night Mode Timer for blinking yellow light
nightModeTimer = new Timer(500, e -> {
    currentLight = currentLight == 1 ? -1 : 1; // Toggle yellow light
    lightPanel.repaint();
});

// Button Listeners
startButton.addActionListener(e -> {
    lightCycleTimer.start();
    startButton.setEnabled(false);
    stopButton.setEnabled(true);
});

stopButton.addActionListener(e -> {
    lightCycleTimer.stop();
    nightModeTimer.stop();
    startButton.setEnabled(true);
    stopButton.setEnabled(false);
});

peakHourButton.addActionListener(e -> {
    isPeakHour = !isPeakHour;
    if (isPeakHour) {
        lightTimings[0] = 10000; // Increase Red light duration
        lightTimings[2] = 3000;  // Decrease Green light duration
        JOptionPane.showMessageDialog(this, "Peak Hour Mode Activated!");
    } else {
        lightTimings[0] = Integer.parseInt(redTimeField.getText());
        lightTimings[2] = Integer.parseInt(greenTimeField.getText());
```

```
        JOptionPane.showMessageDialog(this, "Peak Hour Mode Deactivated!");
    }
    if (lightCycleTimer.isRunning()) {
        lightCycleTimer.setDelay(lightTimings[currentLight]); // Update delay immediately
    }
});

ambulanceButton.addActionListener(e -> {
    lightCycleTimer.stop();
    currentLight = 2; // Green light
    lightPanel.repaint();
    JOptionPane.showMessageDialog(this, "Ambulance Priority: Green Light Activated!");
    lightCycleTimer.start();
});

nightModeButton.addActionListener(e -> {
    isNightMode = !isNightMode;
    if (isNightMode) {
        lightCycleTimer.stop(); // Stop normal cycle
        currentLight = 1; // Set to yellow light
        nightModeTimer.start(); // Start blinking
        JOptionPane.showMessageDialog(this, "Night Mode Activated: Blinking Yellow Light");
    } else {
        nightModeTimer.stop(); // Stop blinking
        currentLight = 0; // Reset to red light
        lightPanel.repaint();
        JOptionPane.showMessageDialog(this, "Night Mode Deactivated");
    }
});
```

```
        applyButton.addActionListener(e -> {
            try {
                int redTime = Integer.parseInt(redTimeField.getText());
                int yellowTime = Integer.parseInt(yellowTimeField.getText());
                int greenTime = Integer.parseInt(greenTimeField.getText());

                if (redTime > 0 && yellowTime > 0 && greenTime > 0) {
                    lightTimings[0] = redTime;
                    lightTimings[1] = yellowTime;
                    lightTimings[2] = greenTime;
                    lightCycleTimer.setDelay(lightTimings[currentLight]); // Update delay immediately
                    JOptionPane.showMessageDialog(this, "Timings Updated Successfully!");
                } else {
                    JOptionPane.showMessageDialog(this, "Invalid Input: Timings must be positive integers.");
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(this, "Invalid Input: Please enter valid numbers.");
            }
        });
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            TrafficLightController trafficLightController = new TrafficLightController();
            trafficLightController.setVisible(true);
        });
    }
}
```