```java
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Point;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;
import javax.imageio.ImageIO;

public class SimpleTrafficLight implements Paintable {
    private Point position; // Posição do semáforo
    private Dimension dimension; // Dimensão do semáforo
    private SpotLight yellow; // Luz amarela
    private SpotLight green; // Luz verde
    private SpotLight red; // Luz vermelha
    private BufferedImage mask; // Imagem do semáforo


    public SimpleTrafficLight() throws IOException {
        this(new Point(0, 0), new Dimension(70, 180));
    }


    public SimpleTrafficLight(Point position, Dimension dimension) throws IOException {
        this.position = position;
        this.dimension = dimension;
        create(); // Inicializa as luzes
        configurePositions();
    }


    private void create() throws IOException {
        this.green = createSpot("green");
        this.yellow = createSpot("yellow");
        this.red = createSpot("red");


        String path = this.currentRelativePath();
        URL url = this.getClass().getResource(path + "/img/trafficLight.png");
        this.mask = ImageIO.read(url);
    }


    private SpotLight createSpot(String color) {
        SpotLight spotLight = new SpotLight(color);
        spotLight.setLight(new E27LightBulb());
        return spotLight;
    }


    private void configurePositions() {
        final int WIDTH = this.dimension.width - 20; //
        final int HEIGHT = (this.dimension.height - 30) / 3;

        int xLeft = this.position.x + 10; // Posição X com margem
        int yTop = this.position.y + 10;   // Posição Y com margem


        this.green.setPosition(xLeft, yTop);
        this.green.setDimension(new Dimension(WIDTH, HEIGHT));
        yTop += (5 + HEIGHT);
        this.yellow.setPosition(xLeft, yTop);
        this.yellow.setDimension(new Dimension(WIDTH, HEIGHT));

        yTop += (5 + HEIGHT); // Espaçamento entre as luzes
        this.red.setPosition(xLeft, yTop);
        this.red.setDimension(new Dimension(WIDTH, HEIGHT));
    }


    @Override
    public void paint(Graphics g) {
        int xLeft = this.position.x;
        int yTop = this.position.y;
        int width = this.dimension.width;
        int height = this.dimension.height;

        // Desenha a imagem do semáforo
        g.drawImage(mask, xLeft, yTop, width, height, null);

        // Desenha as luzes do semáforo, se estiverem ligadas
        this.green.paint(g);
        this.yellow.paint(g);
        this.red.paint(g);
    }


    private String currentRelativePath() {
        return "/" + this.getClass().getPackageName().replace('.', '/'); // Converte o nome do pacote
para caminho relativo
    }


    private class SpotLight implements Paintable {
        private String color;
        private Dimension dimension;
        private Point position;
        private E27LightBulb light; // Exemplo de lâmpada
        private boolean isOn; // Estado da luz

        public SpotLight(String color) {
            this.color = color;
            this.isOn = false; // Inicialmente a luz está desligada
        }

        public void setPosition(int x, int y) {
            this.position = new Point(x, y);
        }

        public void setDimension(Dimension dimension) {
            this.dimension = dimension;
        }

        public void setLight(E27LightBulb light) {
            this.light = light;
        }

        public void setOn(boolean isOn) {
            this.isOn = isOn; // Define o estado da luz
        }
        public boolean isOn() {
            return this.isOn; // Retorna o estado da luz
        }

        @Override
        public void paint(Graphics g) {
            if (isOn) { // Desenha apenas se a luz estiver ligada
                g.setColor(getColor());
                g.fillOval(position.x, position.y, dimension.width, dimension.height);
            }
        }

        private java.awt.Color getColor() {
            switch (color) {
                case "green":
                    return java.awt.Color.GREEN;
                case "yellow":
                    return java.awt.Color.YELLOW;
                case "red":
                    return java.awt.Color.RED;
                default:
                    return java.awt.Color.BLACK;
            }
        }

        @Override
        public String toString() {
            return color + " at " + position + " with dimension " + dimension;
        }
    }

    // Dummy class for E27LightBulb (to be implemented)
    private class E27LightBulb {
        // Implementação da lâmpada E27
    }

    // Método para visualizar as luzes
    public void displayLights() {
        System.out.println(green);
        System.out.println(yellow);
        System.out.println(red);
    }


    public void setLightStates(boolean greenOn, boolean yellowOn, boolean redOn) {
        this.green.setOn(greenOn);
        this.yellow.setOn(yellowOn);
        this.red.setOn(redOn);
    }


    public static void main(String[] args) {
        try {
            SimpleTrafficLight trafficLight = new SimpleTrafficLight();
            trafficLight.setLightStates(true, false, false); // Liga a luz verde
            trafficLight.displayLights();
        } catch (IOException e) {
            System.err.println("Erro ao carregar a imagem do semáforo: " + e.getMessage());
        }
    }
}
```