Práctica 2. Patrones de Diseño

Sergio Martín Vera https://github.com/smv762e/SistemaLuces

Apartado A:

Un patrón de diseño adecuado para el sistema propuesto sería el de Estado, ya que, permite que un objeto cambie su comportamiento en función del estado interno de este.

La clase Biestable es una solución propuesta al sistema solicitado. Además, implementa la interfaz IEstado.

Atributos:

boolean[] lights // lista para guardar el estado de las luces

static final int numLights = 2 // luces que permite el sistema

String estado // estado de las luces

Métodos:

Biestable() // constructor de la clase

void setSite(int light, boolean site) // método para señalar el estado de las luces manualmente

String getEstado() // método para comprobar el estado de las luces

void abrir() // método para elevar el estado de la luz +1 nivel

void cerrar() // método para bajar el estado de la luz -1 nivel

La clase TestBiestable prueba el funcionamiento de este sistema.

Apartado B:

Reutilizando el código del apartado anterior, se ha conseguido establecer un sistema Triestable.

La clase Triestable es una solución propuesta al sistema solicitado. Sigue implementado la interfaz IEstado.

Atributos:

boolean[] lights // lista para guardar el estado de las luces

static final int numLights = 3 // luces que permite el sistema

String estado // estado de las luces

Métodos:

Triestable() // constructor de la clase

void setSite(int light, boolean site) // método para señalar el estado de las luces manualmente

String getEstado() // método para comprobar el estado de las luces

void abrir() // método para elevar el estado de la luz +1 nivel

void cerrar() // método para bajar el estado de la luz -1 nivel

La clase TestTriestable prueba el funcionamiento de este sistema.

Apartado C:

Reutilizando el código del apartado anterior, se ha conseguido establecer un sistema que comprenda los dos sistemas previos y permita cambiar entre ellos.

La clase SistemaMixto es una solución propuesta al sistema solicitado. Sigue implementando la interfaz IEstado.

Atributos:

boolean[] lights // lista para guardar el estado de las luces

int numLights // luces que permite el sistema

String estado // estado de las luces

Métodos:

SistemaMixto() // constructor de la clase

void setSystem(int num) // método para seleccionar el sistema requerido entre 2 y 3 luces

void setSite(int light, boolean site) // método para señalar el estado de las luces manualmente

String getEstado() // método para comprobar el estado de las luces

void abrir() // método para elevar el estado de la luz +1 nivel

void cerrar() // método para bajar el estado de la luz -1 nivel

void cambio() // método para cambiar de sistema cuando se ha seleccionado uno previamente

La clase TestSistemaMixto prueba el funcionamiento de este sistema.

Si se quiere realizar un cambio de sistema en una situación de un Triestable cuando se encuentra en estado Amarillo, el sistema cambiará a un sistema Biestable y estará en un estado Rojo.