ERS FORMATO BREVE PATRÓN DE USO STATE

Análisis y diseño de software
2. Grado en Ingeniería Informática – 2. semestre

Autor: Oihane Lameirinhas Ortuoste

Contenido

ERS Formato Breve – Patrón de Usos

1. Introducción

Este documento contiene información sobre el patrón de uso “State”, que gestiona en este caso los estados de los pisos que contiene la Asociación Afro situado en Vitoria-Gazteiz. El objetivo de esta breve especificación es definir de manera clara y precisa la funcionalidad del del patrón y ver cómo afecta en el sistema.

2. Significado

Este patrón “State” resulta útil cuando necesitamos que un objeto se comporte de forma diferente dependiendo del estado interno en el que se encuentre en cada momento.

Contendrá una referencia a otro objeto que define los distintos tipos de estado en que se puede encontrar.

3. Uso del patrón “State”

El sistema tiene en cuenta la cantidad de personas que ocupan un piso.

Cuando el trabajador o el voluntario de la asociación busca la información de un piso en concreto, en la pantalla se mostrará la situación de ese piso además de toda la información referente al piso. Aparecerá un indicador a color que nos dará la información del estado de ese piso:

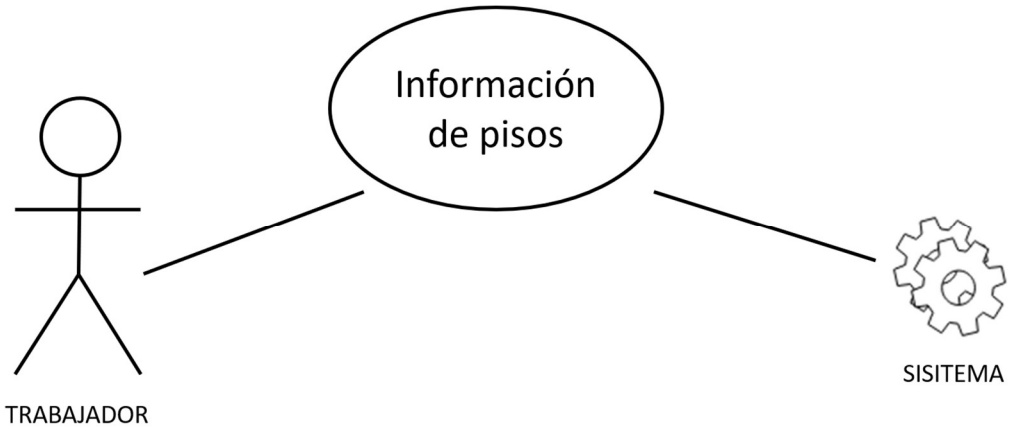
-Verde: El piso está totalmente vacío.
-Amarillo: El piso está medio lleno.
-Rojo: El piso está completo.

De esta manera, según en que situación se encuentre el piso, en pantalla nos aparece un color diferente.

Tabla de Situaciones

PISO	CANTIDAD MÁX	CANTIDAD OCUPADA	COLOR
Piso 1	6	0	
Piso 2	8	4	
Piso 3	4	4	

4. Caso de uso

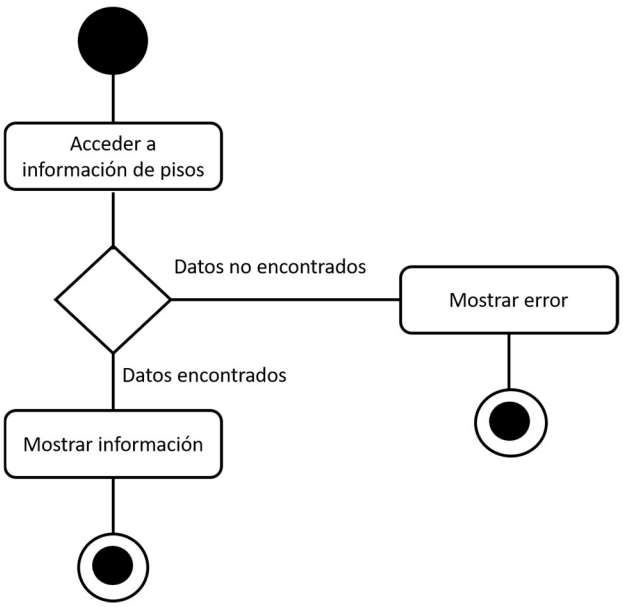


Este patrón se activará en el momento que el trabajador/voluntario acceda a la información de los pisos.

5. Escenario

Caso de uso: Información pisos	
Actor: Trabajador/Voluntario	
Curso normal	**Curso alternativo**
1.Se inicia el sistema	
2. El usuario accede a la información de pisos	
3. El sistema muestra la información, incluyendo los colores de estados de los pisos.	3.1. El sistema no puede cargar la información
4. Fin del caso de uso	

6. Diagrama de actividad



7. Implementación del patrón “State”

En las siguientes imágenes se muestra el código de “State”:

Estas 3 imágenes corresponden al código que permite crear la ventana de los pisos.

```
private Component crearPanelDatos() {
    JPanel panel = new JPanel(new GridLayout(12, 1));

    identificador = new JLabel(String.valueOf(this.piso.getId()));
    nomPrograma = new JLabel(this.piso.getNomPrograma());
    tipo = new JLabel(this.piso.getTipo());
    intensidad = new JLabel(this.piso.getIntensidad());
    direccion = new JLabel(this.piso.getDireccion());
    telefono = new JLabel(String.valueOf(this.piso.getTelefono()));
    aforo = new JLabel(String.valueOf(this.piso.getAforo()));
    subven = new JLabel(String.valueOf(this.piso.getSubvencion()));
    // Crea el cuadrado que cambiara de color
    Cuadrada = new JLabel();
    // Añade las medidas al cuadrado
    Cuadrada.setPreferredSize(new Dimension(2, 2));
    // Añade color al cuadrado
    setBackColor();

    panel.add(crearComponentBorder(identificador, "Identificador"));
    panel.add(crearComponentBorder(nomPrograma, "Nombre del programa al que pertonece"));
    panel.add(crearComponentBorder(tipo, "Tipo de piso"));
    panel.add(crearComponentBorder(intensidad, "Intensidad"));
    panel.add(crearComponentBorder(aforo, "Aforo del piso"));
    panel.add(crearComponentBorder(direccion, "Domicilio"));
    panel.add(crearComponentBorder(telefono, "Teléfono"));
    panel.add(crearComponentBorder(subven, "Subvencion recivida"));
    panel.add(crearComponentBorder(Cuadrada, "Estado de ocupacion del piso"));

    return panel;
}
```

ç

```
// Funcion que usa State
public void setBackColor() {
    PisoEsta objPiso = new PisoEsta();
    // Comprueba la cantidad de personas que viven en el piso
    if (Integer.parseInt(String.valueOf(this.piso.getInquilino())) == 0) {
        //Añade el estado
        objPiso.setEstado(new EstadoVerde());
        //Añade el color al Cuadrado
        Cuadrada = objPiso.mostrar(Cuadrada);
    } else if (Integer.parseInt(String.valueOf(this.piso.getInquilino())) > 0 && (Integer.parseInt(
            String.valueOf(this.piso.getInquilino())) < (Integer.parseInt(String.valueOf(this.piso.getAforo()))))) {
        //Añade el estado
        objPiso.setEstado(new EstadoAmarillo());
        //Añade el color al Cuadrado
        Cuadrada = objPiso.mostrar(Cuadrada);
    } else if (Integer.parseInt(
            String.valueOf(this.piso.getInquilino())) == (Integer.parseInt(String.valueOf(this.piso.getAforo())))) {
        //Añade el estado
        objPiso.setEstado(new EstadoRojo());
        //Añade el color al Cuadrado
        Cuadrada = objPiso.mostrar(Cuadrada);
    }
}
```

```
package PatronesDeDiseño;

import javax.swing.JLabel;

//Clase Abstracta de el Patron de Diseño de State, aqui se determinan las funciones comunes para todas las clases
public abstract class EstadoPiso {
    public EstadoPiso() {
    }

    //Funcion comun para todas las clases
    public abstract JLabel mostrar(JLabel Cuadrada);
}
```

Las siguientes imágenes en cambio, corresponden a la estructura interna del “State”.

```
package PatronesDeDiseño;

import javax.swing.JLabel;

public class PisoEsta extends EstadoPiso  {
    private EstadoPiso objEstadoPiso;

    // Constructor de la clase
    public PisoEsta() {
    }

    // Añade el estado donde tiene que estar el cuadrado
    public void setEstado(EstadoPiso objEstadoPiso) {
        this.objEstadoPiso = objEstadoPiso;
    }

    // Funcion que cambia de color al cuadrado
    @Override
    public JLabel mostrar(JLabel Cuadrada) {
        return Cuadrada;
    }
}
```

```java
package PatronesDeDiseño;

import java.awt.Color;

import javax.swing.JLabel;

public class EstadoAmarillo extends EstadoPiso {
    public EstadoAmarillo() {
    }

    //Cambia el color a Amarillo
    @Override
    public JLabel mostrar(JLabel Cuadrada) {
        Cuadrada.setBackground(Color.RED);
        return Cuadrada;
    }
}
```

```java
package PatronesDeDiseño;

import java.awt.Color;

import javax.swing.JLabel;

public class EstadoAmarillo extends EstadoPiso {
    public EstadoAmarillo() {
    }

    //Cambia el color a Amarillo
    @Override
    public JLabel mostrar(JLabel Cuadrada) {
        Cuadrada.setBackground(Color.YELLOW);
        return Cuadrada;
    }
}
```

```java
package PatronesDeDiseño;

import java.awt.Color;

import javax.swing.JLabel;

public class EstadoRojo extends EstadoPiso {
    public EstadoRojo() {
    }

    // Cambia el color a Rojo
    @Override
    public JLabel mostrar(JLabel Cuadrada) {
        Cuadrada.setBackground(Color.RED);
        return Cuadrada;
    }
}
```

```java
package PatronesDeDiseño;

import java.awt.Color;

import javax.swing.JLabel;

public class EstadoVerde extends EstadoPiso {
    public EstadoVerde() {
    }

    // Cambia el color a Verde
    @Override
    public JLabel mostrar(JLabel Cuadrada) {
        Cuadrada.setBackground(Color.GREEN);
        return Cuadrada;
    }
}
```