Practica 1

Juárez Ubaldo Juan Aurelio - 421095568
Sergio Mejía Caballero - 31511149

1 Teoría

1. Menciona los principios de diseño esenciales del patrón Strategy y Observer.
   Menciona una desventaja de cada patrón

   - Strategy.
     El patrón Strategy nos permite encapsular algoritmos que definen el comportamiento de un mismo objeto en clases separadas. De esta manera, agregar o quitar comportamientos no afecta el codigo ya escrito, y podemos cambiar el comportamiento del objeto en tiempo de ejecucion. Una desventaja que encontramos, es que si el programa tuviera pocos algoritmos que raramente cambian, usar Strategy complicaria el codigo, ya que estamos agregando clases e interfaces en exceso que complicarian mas el codigo. Por lo que creemos que es mejor usarlo cuando tenemos algoritmos que cambian con frecuencia.
   - Observer.
     El principio fundamental de Observer es establecer una relacion de "uno a muchos", de tal manera que cuando el objeto Subject, cambia de estado, todos sus observadores son notificados de este cambio de estado y puedan actualizarse de acuerdo a sus metodos definidos. Una desventaja que encontramos en el patròn es que los objetos dependen del cambio de estado del sujeto para tomar alguna accion.

2 README

En esta practica, usamos los patrones Strategy y Observer para simular un entorno en el cual Clientes se pueden suscribir, cambiar de plan, pagar y anular suscripciones de uno o varios Servicios de Streaming, donde cada servicio tiene un metodo de pago distinto. Usamos Observer para poder conectar a los servicios con los clientes que esten suscritos a ellos, de esta manera, nos aseguramos que cada cliente reciba una recomendacion del servicio y le notificamos que tiene que pagar cada que pase un mes. Para implementar el pago, usamos el patron Strategy, ya que cada cliente tiene su cuenta de banco para pagar sus servicios, usando el patron Strategy podemos cambiar el comportamiento del pago del cliente (dependiendo del servicio que cobre) usando el mismo objeto (payment en nuestro caso) y cambiarlo en tiempo de ejecucion.

3 Anotaciones

- Para imprimir el paso de los meses, decidimos imprimir los eventos que pasan por cada servicio, es decir, cada mes que pasa, se imprime por servicio a que clientes se les cobro que servicio. Esto porque nuestro programa tiene una lista de servicio e iteramos sobre ella cada mes. Un efecto que lo anterior ocasiono, es que si un cliente no puede pagar el servicio, no podiamos removerlo mientras iterabamos sobre la lista de clientes del servicio ya que arrojaba un error de concurrencia. Para solucionar esto, si determinamos que un cliente no puede pagar, imprimimos un mensaje de que el pago no se realizo, lo añadimos a una lista de deudores y al final del metodo recorremos la lista de deudores para removerlos del servicio.

- Para llevar un registro de que plan tienen los clientes, usamos un HashMap que guarda al objeto cliente y un objeto de tipo InfoCliente para guardar la informacion relevante del cliente (mes en el que se suscribe, plan, meses suscrito), para poder facilitar el cobro cada mes.