Practica01_ModeladoYProgramacion

Equipo: SQLazo

- César Becerra Valencia (322064287)
- Victor Abraham Sánchez Morgado (322606003)
- Cortes Nava José Luis (322115437)

Versión de Java utilizada: 17

Descripción de las clases:

- Main: simplemente manda a llamar la simulación con un objeto simulador
- Simulator: es aquella que corre la simulación, definiendo a los usuarios concretos y cómo interactúan
- HvoMax, Memeflix, Momazon, Spootify, ThisneyPlus: son los sujetos concretos, es decir, los servicios de Streaming para los usuarios
- Plan: clase que representa un plan de subscripción, este plan puede ser de cualquier tipo
- Service: clase abstracta que implementa Subject, sirve para definir las acciones que podrán realizar los sujetos
- User: clase que implementa las acciones que podrán realizar los usuarios, o sea, observadores concretos
- Observer: interfaz que representa a los observadores con el método update
- Subject: interfaz que representa a los sujetos, que serán servicios de streaming
- FixedPayment: clase para los pagos acordados desde un inicio
- FreePayment: representa a los pagos gratuitos de servicios de streaming
- PaymentPromotion: clase que representa a los tipos de pago que cambian después de un tiempo
- PaymentStrategy: interfaz con los métodos de pago y descripción, es la base del patrón Strategy

El proyecto consiste en una simulación de cobros y funcionamiento de plataformas de streaming y cómo interactúan estas con los usuarios. El programa nos permite tener observadores de uno o varios sujetos. En este caso específico los sujetos que son servicios de Streaming proveen a los observadores y a cambio les cobran, sin embargo los sujetos se pueden subscribir y desubscribir en cualquier momento que deseen. El programa al ser finalizado genera un archivo.txt en el cuál se imprime la secuencia del programa, con los cobros, acciones hechas por los usuarios y las recomendaciones del mes proporcionadas por los sujetos.

Los patrones utilizados fueron los siguientes:

- Strategy: Dada las instrucciones de la práctica se tiene que cobrar de distintas maneras a los usuarios de las plataformas de streaming, por lo que implementamos el patrón Strategy para que se pueda cobrar en estas modalidades. Esto nos permite evitar tener un código lleno de if's, por lo que solamente es necesario indicar qué estilo de cobranza tendrá cada plataforma de streaming.
- Observer: Resulta muy simple identificar la razón por la que se usa el patrón observer. Tenemos sujetos que son las plataformas de streaming, las cuales van a notificar a su lista de observadores (que en este caso serán los usuarios de la plataforma). De esta forma podemos crear una relación usuario-plataforma que nos va a servir para cobrar y notificar a los usuarios. El comportamiento del observador no debe influir de ninguna manera en el comportamiento del sujeto, lo cuál nos sirve ya que no queremos que la plataforma de streamig se vea afectada por el usuario.