Desafío - Membresía

En este desafío validaremos nuestros conocimientos de herencia de clases y sobreescritura de métodos. Para lograrlo, necesitarás utilizar el archivo ***Apoyo Desafío guiado - Membresía.zip***.

Lee todo el documento antes de comenzar el desarrollo **individual/grupal**, para asegurarte de tener el máximo de puntaje y enfocar bien los esfuerzos.

Descripción

Has sido contratado como programador/a backend por una empresa que se encuentra desarrollando una aplicación de streaming de videos de películas y series chilenas. En esta ocasión, se te ha solicitado desarrollar la estructura de clases que permita crear membresías de los usuarios suscriptores. Existen 5 tipos de membresía: ***Gratis***, ***Básica***, ***Familiar***, ***Sin Conexión*** y ***Pro***.

Como el foco actual es el diseño, solo se le ha pedido considerar que para crear cada membresía se debe solicitar el **correo electrónico** y el **número de tarjeta** (como texto) del suscriptor. Adicionalmente, algunos tipos de membresía reciben **días de regalo** al crearse (reciben un valor fijo, pero que en el futuro podría se decidir modificar). En cuanto a las funcionalidades, todos los tipos de membresía *deben* permitir **cambiar la suscripción**, lo que genera una nueva membresía (según el tipo solicitado) conservando el correo electrónico y el número de tarjeta de la membresía original.

Para el **cambio de membresía**, debe utilizar un identificador numérico para identificar el tipo de membresía solicitada, según lo siguiente:

- 1: Básica
- 2: Familiar
- 3: Sin Conexión
- 4: Pro

NOTA: Puedes utilizar el método "`_crear_nueva_membresia`" (dentro de la(s) clase(s) que estimes adecuado) que se encuentra en el archivo `apoyo_desafio.py`. Este método tiene un nivel de acceso de tipo "***protected***" (dado por el underscore "_" al principio del nombre). Un método de tipo "protected" solo puede ser accedido por la misma clase donde se define, y por todas las clases que le hereden.

En cuanto al ***detalle de los tipos de membresía***, debes tener en cuenta las siguientes consideraciones para el desarrollo solicitado:

- No puede existir una **"Membresía"** como tal, sino que siempre debe ser de algún tipo específico, que se detalla en los siguientes puntos.

- Todas las membresías de tipo "**Gratis**" tienen un *costo de $0, y una cantidad máxima de 1 dispositivo*. Comportamiento:
  - Si alguien con este tipo de membresía desea ***cambiar su suscripción***, debe validar que la nueva membresía solicitada sea un número entre *1 y 4 inclusive*, y luego generar la nueva membresía. En caso contrario, debe retornar la membresía actual.

- Todas las membresías de tipo "**Básica**" tienen un costo de *$3000, y una cantidad máxima de 2 dispositivos*. Comportamiento:
  - Si alguien con este tipo de membresía desea ***cambiar su suscripción***, debe validar que la nueva membresía solicitada sea un número entre *2 y 4 inclusive*, y luego generar la nueva membresía. En caso contrario, debe retornar la membresía actual.

- Todas las membresías de tipo "**Familiar**" tienen un costo de *$5000, y una cantidad máxima de 5 dispositivos*. Al momento de crear una membresía Familiar, se debe asignar *7* a un atributo de instancia que almacene los *días de regalo*. Comportamiento:
  - Si alguien con este tipo de membresía desea ***cambiar su suscripción***, debe validar que la nueva membresía solicitada sea un *1, 3 ó 4*, y luego generar la nueva membresía. En caso contrario, debe retornar la membresía actual.
  - Debe permitir modificar el control parental. La lógica de este comportamiento aún no se ha definido, pero debe declararlo (sin implementación).

- Todas las membresías de tipo "**Sin Conexión**" tienen un costo de *$3500, y una cantidad máxima de 2 dispositivos*. Al momento de crear una membresía Sin Conexión, se debe asignar *7* a un atributo de instancia que almacene los *días de regalo*. Comportamiento:
  - Si alguien con este tipo de membresía desea ***cambiar su suscripción***, debe validar que la nueva membresía solicitada sea un *1, 2 ó 4* , y luego generar la nueva membresía. En caso contrario, debe retornar la membresía actual.
  - Debe permitir incrementar la cantidad máxima de contenido disponible para ver sin conexión. La lógica de este comportamiento aún no se ha definido, pero debe declararlo (sin implementación).

- Todas las membresías de tipo "**Pro**" tienen un costo de *$7000, y una cantidad máxima de 6 dispositivos*. Al momento de crear una membresía Pro, se debe asignar *15* a un atributo de instancia que almacene los *días de regalo*. Comportamiento:

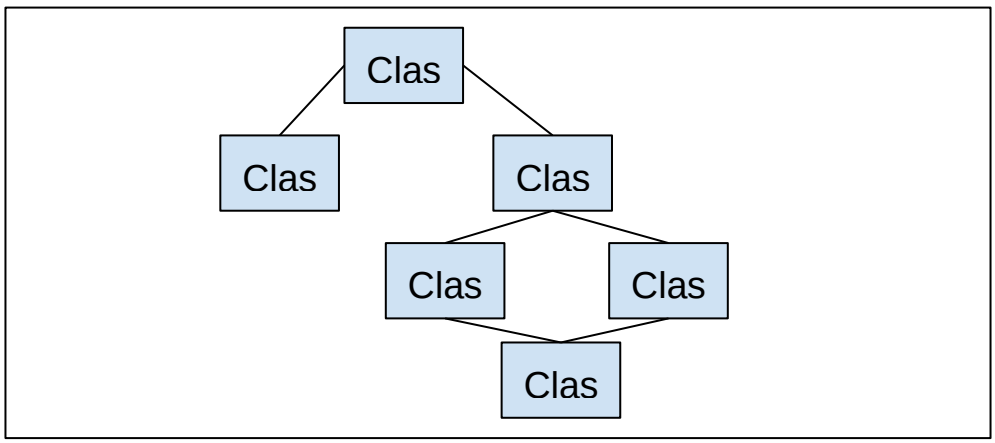
- Si alguien con este tipo de membresía desea ***cambiar su suscripción***, debe validar que la nueva membresía solicitada sea un número entre *1 y 3 inclusive*, y luego generar la nueva membresía. En caso contrario, debe retornar la membresía actual.
- Debe permitir modificar el control parental (igual a membresía Familiar)
- Debe permitir incrementar la cantidad máxima de contenido disponible para ver sin conexión (Igual a membresía Sin Conexión).

- Las membresías de tipo "**Básica**", "**Familiar**", "**Sin Conexión**" y "**Familiar**" deben tener además el siguiente comportamiento:
  - Deben permitir ***cancelar la suscripción***. Al realizar esta acción, se debe generar una membresía de tipo Gratis con el mismo correo y número de tarjeta de la membresía original.

Tips:

- Para resolver este desafío con menos líneas de código, puede aplicar herencia híbrida, donde:
  - Dos de las clases son padre de más de una clase (herencia jerárquica).
  - Tres de las clases son hija y padre a la vez (herencia multinivel).
  - Una de las clases hereda de más de una clase (herencia múltiple).

- Considerando lo anterior, el ordenamiento de las clases puede ser el siguiente:



Requerimientos

1. En un archivo membresia.py, crear la clase que permita definir los atributos de instancia y comportamiento de ***todas las membresías***. Considere:
   (2 Puntos)
   a. Utilice abstracción para definir el o los comportamientos requeridos (puede definir también métodos no abstractos).
   b. Utilice encapsulamiento para los atributos de instancia. Declare las propiedades que estime necesarias según lo solicitado.

2. En el mismo archivo, crear la clase que permita crear instancias de membresías de tipo ***Gratis***. Considere:
   (2 Puntos)
   a. Heredar la o las clases necesarias para evitar repetir implementaciones.
   b. Definir los atributos de clase necesarios.
   c. Definir los métodos necesarios (en caso de que se justifique).
   d. Sobrescribir los métodos necesarios (en caso de que se justifique).

3. En el mismo archivo, crear la clase que permita crear instancias de membresías de tipo ***Básica***. Considere:
   (2 Puntos)
   a. Heredar la o las clases necesarias para evitar repetir implementaciones.
   b. Definir los atributos de clase necesarios.
   c. Definir los métodos necesarios (en caso de que se justifique).
   d. Sobrescribir los métodos necesarios (en caso de que se justifique).

 ***Tip:*** Puede ser útil el uso de `isinstance` en para establecer los días de regalo.

4. En el mismo archivo, crear la clase que permita crear instancias de membresías de tipo ***Familiar***. Considere:
   (2 Puntos)
   a. Heredar la o las clases necesarias para evitar repetir implementaciones.
   b. Definir los atributos de clase necesarios.
   c. Definir los métodos necesarios (en caso de que se justifique).
   d. Sobrescribir los métodos necesarios (en caso de que se justifique).

5. En el mismo archivo, crear la clase que permita crear instancias de membresías de tipo ***Sin Conexión***. Considere:
   (1 Punto)
   a. Heredar la o las clases necesarias para evitar repetir implementaciones.

   b. Definir los atributos de clase necesarios.
   c. Definir los métodos necesarios (en caso de que se justifique).
   d. Sobrescribir los métodos necesarios (en caso de que se justifique).

6. En el mismo archivo, crear la clase que permita crear instancias de membresías de tipo ***Pro***. Considere:
   (1 Punto)
   a. Heredar la o las clases necesarias para evitar repetir implementaciones.
   b. Definir los atributos de clase necesarios.
   c. Definir los métodos necesarios (en caso de que se justifique).
   d. Sobrescribir los métodos necesarios (en caso de que se justifique).

 ¡Mucho éxito!