Dado un diagrama de clases y una breve explicación, se solicita:

1. Identificar los problemas en la operación
2. Proponer una solución alternativa (incluyendo pseudocódigo)

### Problemas identificados

En la función getTotal() de la clase RegisteredUser se observa que:

1. En función de la clase hija de Service, se obtiene el precio en función de si es StreamingService o DownloadService, es decir, está restringido a los tipos que se han declarado, con lo que se salta el principio abierto-cerrado (O), además, depende de clases concretas, con lo que también violaría el principio de inversión de dependencias (I).
2. De la misma manera, se utiliza PremiumContent para determinar el precio, y está restringido exclusivamente a esta clase, principio (O).
3. En la parte lógica, no tan relacionado con las clases si no con la funcionalidad, nos encontramos que el precio del contenido multimedia está definido en el contenido multimedia, la función getTotal() se define como que devuelve el importe total pagado por los servicios (es un hecho pasado). Si no registramos cuánto se pagó en su momento, cualquier variación en el precio alteraría el cálculo anterior.

Por ejemplo, suponga que para Navidad, los contenidos navideños gozan de un precio menor, un usuario que hubiera adquirido el producto antes de la campaña de Navidad, obtendría, durante la campaña, el precio actual del producto y vería un total inferior al que verdaderamente ha pagado. De la misma manera sucedería al contrario, si un usuario se ha beneficiado de un descuento de Black Friday, a día de hoy, su getTotal() refleja un importe superior al realmente pagado. Con lo que perderíamos la información histórica del usuario.

En términos de escenarios, quizás nos gustaría poder reflejar algunos aspectos tales como:

1. Regirse al máximo posible por los principios SOLID para obtener un código que goce de solidez y escalabilidad.
2. Gestionar de forma sencilla la tarifa premium, a mi entender, es una tarifa fija que se aplica de forma general a todos los contenidos si son premium.
3. Conocer el precio pagado por el usuario por la adquisición del servicio en un determinado momento del tiempo
4. Que un contenido multimedia pudiera estar disponible exclusivamente para streaming o para descarga
5. En una lista de deseos, aplicar de forma sencilla descuentos a contenidos específicos, tanto a nivel de contenido como a nivel de tipo de consumo (streaming o descarga), no se implementa de en la solución propuesta.

Una posible solución pasaría por:

1. Introducir un nuevo elemento Option para referirse a las opciones de compra (streaming y descarga), y que tiene un contenido multimedia
2. Cada servicio, por tanto, pasaría a tener una opción y un coste, que es obtenido a partir del precio en cada momento

Por tanto:

1. Un usuario tendría una colección de servicios, cada uno consta de la opción elegida y el coste.
2. Cada opción consta de los parámetros de precio (precio base y tarifa premium) y del contenido multimedia
3. El contenido multimedia se responsabiliza únicamente de describirse a sí mismo, entre otros, describiendo si es un contenido premium o no.

<https://docs.google.com/drawings/d/1_KkQkszs8LSe0adRLqgI7X24QvgQeGJ5Z9-OQ9CsJio/edit?usp=sharing>

El objetivo principal es simplificar la función getTotal() aprovechando el nuevo diseño:

/\*\*

\* @class RegisteredUser

\*/

class RegisteredUser {

/\*\*

\*

\* @param {string} email

\* @param {boolean} adult

\* @param {Service[]} services

\*/

constructor(email, adult = false, services = []) {

this.email = email;

this.adult = adult;

this.services = services;

}

/\*\*

\*

\* @returns {number}

\*/

**getTotal() {**

**let total = 0;**

**this.services.forEach(service => {**

**total += service.getCost();**

**});**

**return total;**

**}**

// (...)

}

El código completo funcional <https://jsfiddle.net/8k1tgnpj/>