Ejercicio 1

1. Que problemas detectas en la operación y razona la respuesta.

Violación del Principio de Abierto/Cerrado: La implementación actual de `getTotal` no es fácilmente extensible. Si se añaden nuevos tipos de servicios en el futuro, se tendrá que modificar `getTotal`, lo cual no es ideal, ya que el código debería ser abierto para extensión, pero cerrado para modificación.

Falta de Abstracción de Tipos: El código utiliza comprobaciones explícitas del tipo (`StreamingService, DownloadService o PremiumContent`) para determinar el precio aplicable. Este enfoque no es bueno, ya que cada vez que se agregue un nuevo tipo de servicio, será necesario modificar este método, lo cual rompe el principio de abierto/cerrado de la programación orientada a objetos.

Acoplamiento con tipos específicos: La función `getTotal` depende directamente de los tipos `StreamingService, DownloadService y PremiumContent`. Si se agregan más tipos de servicios, tendríamos que modificar esta función. Esto también dificulta realizar pruebas unitarias aisladas para cada tipo de servicio. Además, este acoplamiento fuerte dificulta la reutilización y el mantenimiento del código, ya que cualquier cambio en las clases dependientes puede afectar directamente a RegisteredUser.

La lógica de agregar el `streamingPrice, downloadPrice o additionalFee` está repartida en múltiples condiciones, lo que puede hacer que el método sea difícil de mantener y extender. Se podría aprovechar el polimorfismo y delegar la responsabilidad de calcular el precio a las propias clases de servicio o contenido. Esto no solo simplificaría el código, sino que también lo haría más robusto y fácil de mantener.

Posible Violación de Encapsulamiento: `RegisteredUser` es responsable de calcular el precio de cada tipo de servicio, lo cual debería ser responsabilidad de las clases de `Service` o `MultimediaContent`, en lugar de estar en `RegisteredUser`. Esto hace que el código sea más difícil de mantener y menos modular.

Violación del Principio de Responsabilidad Única: La función `getTotal` en `RegisteredUser` está haciendo demasiadas cosas. No solo suma el total, sino que también decide cómo calcular el precio para cada tipo de servicio. Esto hace que la clase tenga una responsabilidad que no debería tener.

2. Propón una solución alternativa (también en pseudocódigo del mismo estilo) que corrija los problemas de la operación `getTotal` de `RegisteredUser` que has detectado en la pregunta anterior. Realiza todos los cambios que consideres necesarios en cualquiera de las clases del modelo del enunciado.

```
class RegisteredUser {
    constructor(email, password, registration, adult, services = []) {
        this.email = email;
        this.password = password;
        this.registration = registration;
        this.adult = adult;
        this.services = services;
    }

    getTotal() {
        let total = 0;
        this.services.forEach(service => {
            total += service.getPrice();
        });
        return total;
    }
}


class Service {
    constructor(timestamp, multimediaContent) {
        this.timestamp = timestamp;
        this.multimediaContent = multimediaContent;
    }

    getMultimediaContent() {
        return this.multimediaContent;
    }

    getPrice() {
        return this.multimediaContent.getPrice();
    }
}
```

```
class StreamingService extends Service {
    getPrice() {
        return this.multimediaContent.getStreamingPrice();
    }
}


class DownloadService extends Service {
    getPrice() {
        return this.multimediaContent.getDownloadPrice();
    }
}


class MultimediaContent {
    constructor(title, streamingPrice, downloadPrice, duration, adult, size)
{
        this.title = title;
        this.streamingPrice = streamingPrice;
        this.downloadPrice = downloadPrice;
        this.duration = duration;
        this.adult = adult;
        this.size = size;
    }

    getStreamingPrice() {
        return this.streamingPrice;
    }

    getDownloadPrice() {
        return this.downloadPrice;
    }
}
```

```
class PremiumContent extends MultimediaContent {
    constructor(title, streamingPrice, downloadPrice, duration, adult, size,
additionalFee) {
        super(title, streamingPrice, downloadPrice, duration, adult, size);
        this.additionalFee = additionalFee;
    }

    calculatePrice(basePrice) {
        return basePrice + this.additionalFee;
    }

    getStreamingPrice() {
        return this.calculatePrice(super.getStreamingPrice());
    }

    getDownloadPrice() {
        return this.calculatePrice(super.getDownloadPrice());
    }
}
```