Prueba Técnica AXPE - Parte teórica

Siguiendo los principios SOLID, en el código presentado veo un problema de **mantenibilidad y escalabilidad** en caso de que los servicios aumenten en un futuro. Siendo esta la estructura actual:

```
getTotal (){
    let total = 0;

    this.services.forEach(service, index => {
        let multimediaContent = service.getMultimediaContent ();

        if (typeof service == StreamingService) {
            total += multimediaContent.streamingPrice;
        }else if (typeof service == DownloadService) {
            total += multimediaContent.downloadPrice;
        }

        if (typeof multimediaContent == PremiumContent) {
            total += multimediaContent.additionalFee;
        }

    })

    return total;
}
```

Imaginamos que escalamos la aplicación y ampliamos sus servicios. Esto con la estructura if else if actual puede llevar a tener una estructura tipo:

if ():
else if ():
else if ():
else if ():
else if ():
else if ():
...

Esto programáticamente puede llevar a introducir errores y en el caso en que esto ocurra, el resultado de getTotal() sería 0, con el riesgo para la plataforma que esto podría generar respecto a sus ingresos.

Si hiciéramos una estructura que mapease el tipo de servicio al método que devuelve el precio, podríamos usar un ErrorHandler, de tal manera que cada vez que se introdujese un error, la estructura de cálculo de precio total seguiría en pie, tan solo el servicio fallido quedaría excluído. En el ErrorHandler definiríamos que tipo de respuesta esperamos frente a ese error. Ejemplo muy simplificado:

```
SERVICE_PRICE_GETTER_MAP = {
   StreamingService: MultimediaContent.streamingPrice,
   DownloadService: MultimediaCOntent.downloadPrice,
   ...
}
```

if typeof service not in SERVICE_PRICE_GETTER_MAP:

```
    ErrorHandler
else:
   total += SERVICE_PRICE_GETTER_MAP(typeof service)()
```

Otro modo de implementar una solución basándonos en el *Single-responsibility principle* es crear tres funciones:

```
getStreamingPrice()
getDownloadPrice()
getAdditionalFeePrice()
```

Cada una con su correspondiente ErrorHandling encapsuladas en getTotal(). De este modo mejoraríamos la legibilidad sin perder la escalabilidad y poner en riesgo el cálculo del total. Además, desde el punto de vista de la refactorización disponemos de nombres más descriptivos que en la solución anterior.