Prueba Técnica Frontend AXPE Consulting | Manuel Calvo Martín

1. ¿Qué problemas detectas en la operación y razona la respuesta?

Los principales problemas que detecto son que el pseudocódigo no cumple con los principios SOLID y DRY de programación orientada a objetos.

Dentro del método forEach del getTotal, encontramos código duplicado (DRY).

Este código se puede refactorizar a una función separada que tome cada servicio como parámetro y devuelva el total.

Además, respecto a los principios SOLID, RegisteredUser tiene varias responsabilidades, ya que mantiene una lista de servicios y calcula un precio total según estos. Deberíamos separar la gestión de la lista y el cálculo.

Finalmente, en lugar de comprobar los tipos del servicio, se debería comprobar la instancia del servicio (instanceof), al tratarse de una clase personal y no primitiva.

---

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.

Como solución alternativa, deberíamos simplificar RegisteredUser, de forma que solo se ocupe de gestionar los servicios del usuario, y pase el cálculo de los precios a la clase Service.

```
class RegisteredUser {
  constructor(services = []) {
    this.services = services;
  }

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

Por lo tanto, la clase Service añade un nuevo método, llamado getPrice. Este calcula el precio del servicio, según si es una instancia de StreamingService o de DownloadService. Además, si se añaden nuevos tipos de servicio, esto nos permite no tener que modificar al usuario registrado.

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

  getMultimediaContent() {
    return this.multimediaContent;
  }

  getPrice() {
    let price = 0;
    if (this.multimediaContent instanceof MultimediaContent) {
      if (this.multimediaContent instanceof PremiumContent) {
        price += this.multimediaContent.getAdditionalFee();
      }
      if (this instanceof StreamingService) {
        price += this.multimediaContent.getStreamingPrice();
      } else if (this instanceof DownloadService) {
        price += this.multimediaContent.getDownloadPrice();
      }
    }
    return price;
  }
}

class StreamingService extends Service {}

class DownloadService extends Service {}
```

Del mismo modo, comprobamos si el MultimediaContent asociado al servicio es una instancia de PremiumContent, ya que, en este caso, deberemos sumar la additionalFee.

El MultimediaContent y PremiumContent quedan de la misma forma, con los atributos públicos de streamingPrice, downloadPrice y additionalFee para que el método getPrice() de Service pueda acceder a ellos.

De este modo, el código cumple con los principios SOLID, DRY y KISS, permitiendo una mayor robustez del código frente a posibles cambios en escenarios futuros, minimizando el impacto de estos.