Prueba técnica Axpe Consulting

Ejercicio 1

Respuestas:

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

Por convenio se suele poner “capitalizados” los constructores de clases. En el código dentro de los condicionales *if* lo que hace es comparar el *typeof* con tres elementos con la primera letra en mayúsculas y que da a presuponer que son constructores. Lo que parece que intenta hacer es comprobar si esas cosas que esta comparando son ese tipo de clase u objeto.

El problema es que *typeof* devuelve un *string* y no se puede comparar con una clase, es mas el resultado sera ”object” ya que lo que haria seria reconocerlo como un objeto.

Por lo tanto existen dos fallos:

1. Esta intentando comparar un *string* con una clase
2. No daría el resultado que se espera ya que en lugar de dar la clase en *string* como por ejemplo ”StreamingService” devolvería ”object”.

Como inciso me gustaría recalcar que typeof esta puesto dos veces, cuando se podría poner una vez arriba

Para comprobar si un objeto es de una clase se utilizaría *instanceof* con eso podría utilizarse de un modo muy similar a como se utiliza en el código

```
service instanceof StreamingService
```

Con eso solucionaríamos lo que considero es el error mas grande que tiene el código y que impediría que funcionase.

Ahora ciertos detalles que he encontrado diagrama del modelo.

- Se busca manejar precios usando *float*, JavaScript no siempre maneja bien las operaciones con f*loat*. Utilizaría números enteros, es decir en vez de usar dólar o euro como base monetaria usaría céntimos.

- No se contempla la posibilidad de que exista un servicio que no haya sido contemplado entre los existentes, se debería contemplar como mínimo la posibilidad de que haya un error.

- No existe ningún identificador, supongo que es para no complicar el caso, pero no existe ningún atributo tipo id para ser usado como key en React

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.

Actualmente solo hay dos servicios *streaming* y descargas pero ¿que pasa si se añaden mas? Tendríamos que cambiar el código y agregar un nuevo *else if* cada vez que exista un servicio nuevo.

Mi opción de cambio es tratar el servicio como un carrito de la compra sumando el precio de los mismos tanto si son descargas como *streaming* y tratando el contenido *premium* como *addons*.

De esta manera se calcularía el precio de los servicios contemplando que se puedan añadir nuevos en el futuro.

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

    addService(service) {
        this._services.push(service);
    }

    getTotal() {
        return this._services.reduce((total, service) => {
            const serviceTotal = service.getMultimediaContents().reduce((serviceTotal, multimediaContent) => {
                const addonsTotal = multimediaContent.getAddons().reduce((addonsTotal, addon) => {
                    return addonsTotal + addon.additionalFee;
                }, 0);
                return serviceTotal + service.getPrice(multimediaContent) + addonsTotal;
            }, 0);

            return total + serviceTotal;
        }, 0);
    }
}
```

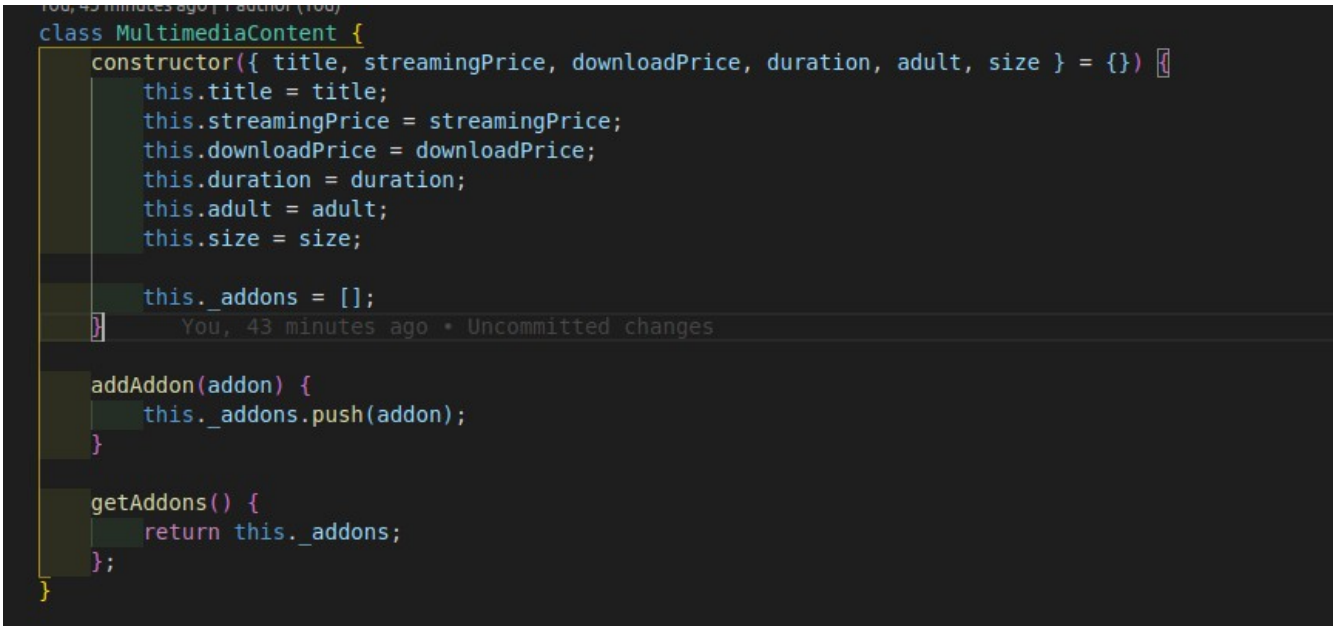
La clase *Service* incluiría la función *getPrice*() para obtener el precio del servicio según se ha elegido llamándolo como expongo en la siguiente imagen a una clase heredada.

```
class Service {
    constructor({ timestamp }) {
        this.timestamp = timestamp;
        this._multimediaContents = [];
    }

    addMultimediaContent(multimediaContent) {
        this._multimediaContents.push(multimediaContent);
    }

    getMultimediaContents() {
        return this._multimediaContents;
    }      You, 22 minutes ago • Uncommitted changes

    getPrice() {
        throw new Error('This method should be called on a child class');
    }
}
```

Ahora las clases antes mencionadas *StreamingService* y *DownloadService* extienden la clase *Service* llaman a la función *getprice()* incluyendo los precios de dichos servicios y agregándolos al carrito. De esa manera si se incluyese un servicio nuevo en el futuro bastaría con hacer una nueva clase de la misma manera.

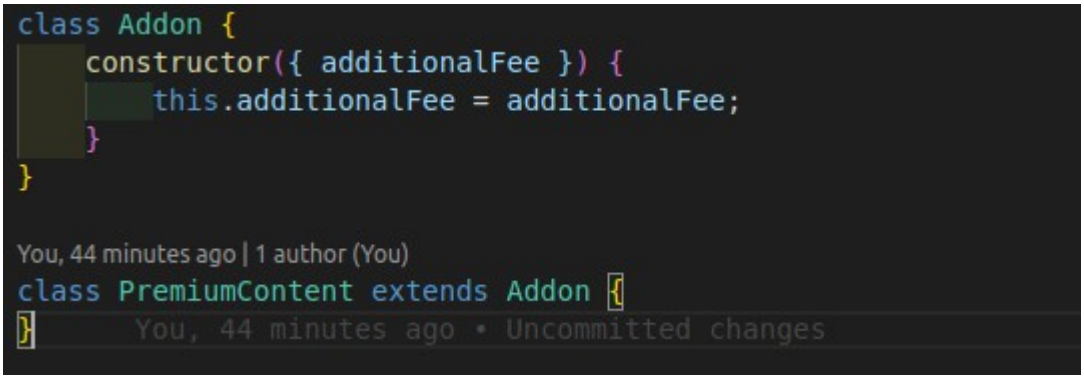
```
You, 27 minutes ago | 1 author (You)
class StreamingService extends Service {
    getPrice(multimediaContent) {
        return multimediaContent.streamingPrice;
    }
}

You, 27 minutes ago | 1 author (You)
class DownloadService extends Service {
    getPrice(multimediaContent) {
        return multimediaContent.downloadPrice;
    }
}
```

La clase *MultimediaContent* mencionada antes se modificaría agregando el *array* de *addons* y las funciones *addAddon*() y *getAddons*() de manera que se añadirían las tarifas cualquier elemento adicional como los servicios *premium*.



```
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;

        this._addons = [];
    }       You, 43 minutes ago • Uncommitted changes

    addAddon(addon) {
        this._addons.push(addon);
    }

    getAddons() {
        return this._addons;
    };
}
```

La clase *Addon* englobaría cualquier elemento extra que pudiese existir y en este caso *PremiumContent* seria una extensión de *Addon*.



```
class Addon {
    constructor({ additionalFee }) {
        this.additionalFee = additionalFee;
    }
}

You, 44 minutes ago | 1 author (You)
class PremiumContent extends Addon {
}       You, 44 minutes ago • Uncommitted changes
```

Poniendo un caso de ejemplo usare un usuario registrado que busque ver una película en streaming y descargar una película para adultos ambas con contenido premium.

```
const user = new RegisteredUser({
    email: 'rob@gmail.com',
    hash: '$2a$12$Xew28iGju7.v1LS7G7H56.OIqfFDolgyfIMXWbHrENahsa/pFshd.',
    registration: new Date(),
    adult: true,
});
```

He pedido ver una pelicula en streaming con contenido premium y sin anuncios (por probar otro caso de *Addon*. Ambos establecidos con anterioridad.

```
const bambiMovie = new MultimediaContent({
    title: 'Bambi',
    streamingPrice: 5,
    downloadPrice: 7,      You, seconds ago • Uncommitted changes
    duration: 120,
    adult: false,
    size: 4_000_000,
});
bambiMovie.addAddon(watchIn4K);
bambiMovie.addAddon(noAds);
```

Los contenidos añadidos serian:

```
const watchIn4K = new PremiumContent({
    additionalFee: 1,
});

const noAds = new Addon({
    additionalFee: 1,
});
```

Con esto el producto se añadiría a un servicio el cual le he dado nombre:

```
const childsNightService = new StreamingService({
    timestamp: new Date(),
});
childsNightService.addMultimediaContent(bambiMovie);
```

Ahora por añadir nuestro contenido de descarga:

```
const coolWorldMovie = new MultimediaContent({
    title: 'Cool World',
    streamingPrice: 5,      You, seconds ago • Uncommitted changes
    downloadPrice: 7,
    duration: 120,
    adult: true,
    size: 4_000_000,
});
coolWorldMovie.addAddon(watchIn4K);

const adultsEveningService = new DownloadService({
    timestamp: new Date(),
});
adultsEveningService.addMultimediaContent(coolWorldMovie);
```

usando la función *addservice*() que aparece en la primera captura de pantalla añadiría los servicios al usuario y luego buscaría calcular el total

```
user.addService(childsNightService);
user.addService(adultsEveningService)
const total = user.getTotal();      You, 2 minutes ago • Unc
console.log('el total es', total);
```

Con esto nos devolvería que el total de ambos servicios es 16