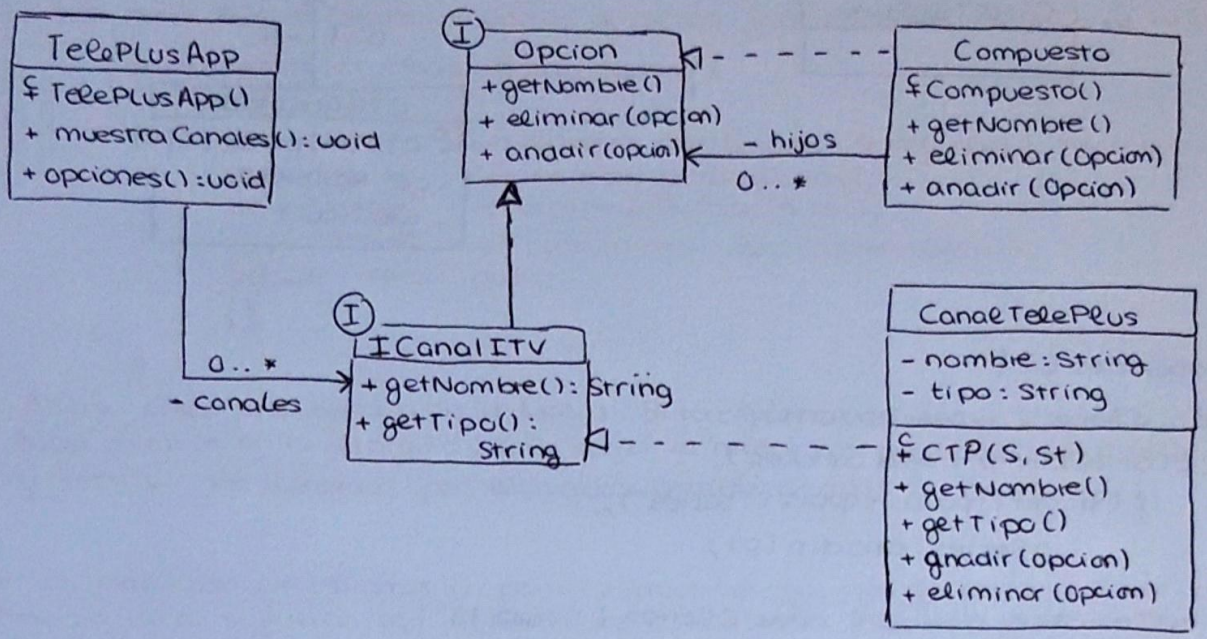
Patrones de diseño - EJ6

En este caso, hay que agrupar los canales y los paquetes de manera jerárquica, por lo que necesitaremos de un patrón estructural para ello, y en este caso es Compose, que nos permite representar en forma de árbol la jerarquía entre objetos simples y agrupaciones.



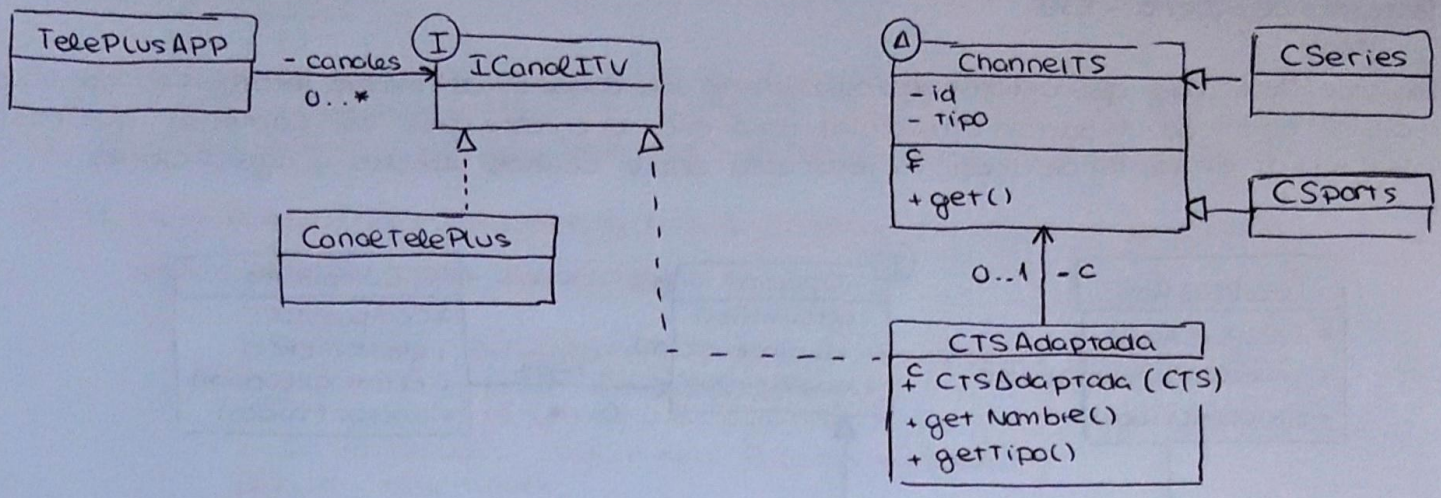

```
public class TeleoPlusApp {
    private List<ICanalITV> canales;
    public void opciones() {
        Opcion oMusica = new Compuesto();
        for (ICanaleITV i : canales)
            if (i.getTipo().equals("musica") oMusica.anadir(i);
        Opcion oSeries = new Compuesto();
        for (ICanaleITV i: canoees)
            if (i.getTipo().equals("series") oSeries.anadir(i);

        Opcion oCombo = new Compuesto();
        oCombo.anadir(oMusica); oCombo.anadir(oSeries);

        ICanaleITV nuevo = new CanaeTeleePlus("40TV", "musica");
        oMusica.(anaair(nuevo);
        oMusica.eliminar(canoles.get(0));

    }
}
```

Para este probeema, debemos usar un patroń estructural, y en este caso el Patroń Adapter, ya que la interfaz ChannelTeleStar no se corresponde con ICanaleITV. Como ambas son clases y no hay múeriple herencia, se usará una clase adaptador por Composición.




```
public void opciones () {
        Opcion oSeries = new Opcion();
        for ( ICanaeITV p : ~~del~~ canales )
                if ( p.getTipo().equals ("series");
                        oSeries.add (p)  ;

        ChannelTeleStar  nueva = new CSeries ( "Calle 13");
        ICanalITV adaptado =  new TeleStarAdapter (nueva);
        oSeries.add (adaptado);
}
```

Para poder usar TelePluss App con el comportamiento expuesto habría que usar un patrón de tipo creacional, y en este caso un Singleton, donde solo se crea un único objeto de la clase común a todas las demás, y siempre se llama al mismo (i, iii). El patrón, tal como indica, hace del constructor privado, con lo que se evita crear instancias fuera de la clase (ii). El Singleton, es una variable global sin valor asignado en un principio (iv)