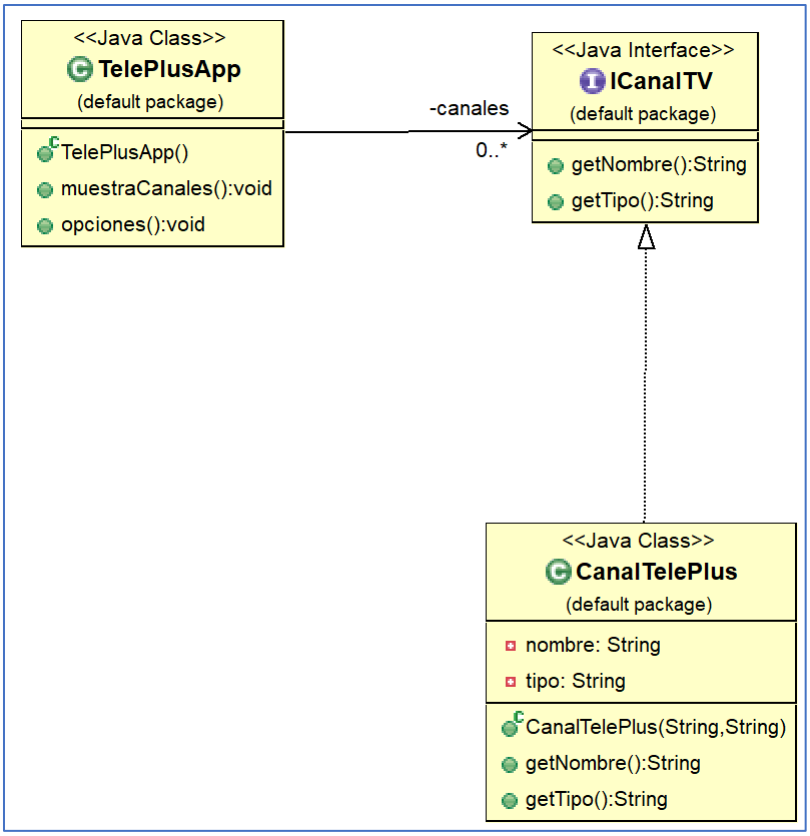
Nombre y Apellidos:	
UO:	**Firma:**

La operadora de televisión por cable TelePlus ofrece a sus clientes una aplicación **TelePlusApp** que ofrece información acerca de los distintos canales de TV que dispone.

Como se puede ver en el siguiente diagrama, la aplicación dispone de un atributo **canales** que es una lista de objetos que implementan la interface **ICanalTV**: cada canal debe tener un nombre que lo identifica, y un tipo que representa la temática del canal (ej: música, series, deportes, etc.).



Cada canal viene modelado por un objeto de la clase **CanalTelePlus** que implementa dicha Interface. El código fuente de estos tipos de datos está al final del enunciado.

La salida generada por la ejecución del método **TelePlusApp.muestraCanales()** será la siguiente:

```
Canales:
FoxLife series
AXN series
HIT-TV música
MTV música
```

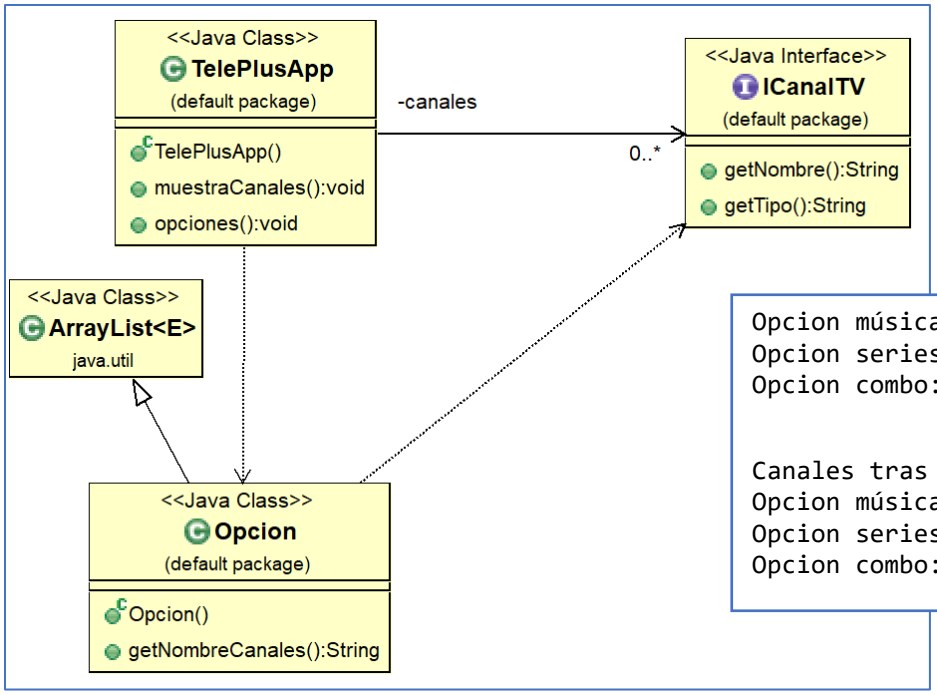
La empresa dispone además de varios paquetes de canales para sus clientes, en concreto:

- OpcionMusica: agrupa los canales de música.
- OpcionSeries: agrupa los canales de series.
- OpcionCombo: agrupa los canales del resto de opciones, actualmente OpcionMusica y OpcionSeries.

El diseño empleado por los programadores actuales de la empresa para los paquetes es demasiado ineficiente. Se ha limitado a crear una simple subclase de **ArrayList<ICanalTV>** denominada **Opcion** que agrupa los canales de un determinado paquete. El método **TelePlusApp.opciones()** ilustra la forma de crear cada paquete y los muestra por consola.

También ilustra lo engorroso que resulta actualizar los canales de una opción al poder ésta formar parte de otra. Así, por ejemplo, al desaparecer el canal FoxLife (de OpcionSeries) y aparecer el canal 40TV (en OpcionMusica), es necesario borrar y añadir estos canales también en OpcionCombo.

A continuación, se muestra el diagrama de clases referido a la clase Opcion y a la salida del método **TelePlusApp.opciones()**.



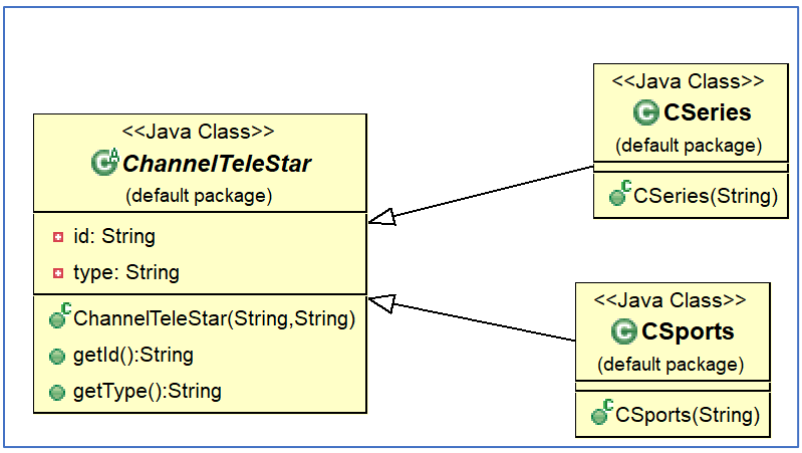

```
Opcion música: HIT-TV MTV
Opcion series: FoxLife AXN
Opcion combo: HIT-TV MTV FoxLife AXN


Canales tras añadir 40TV y eliminar FoxLife
Opcion música: HIT-TV MTV 40TV
Opcion series: AXN
Opcion combo: HIT-TV MTV AXN 40TV
```

1. ¿Qué patrón habría que utilizar para simplificar el mantenimiento de los diferentes paquetes de canales, sabiendo que los paquetes pueden contener canales o bien otros paquetes ya existentes (como es el caso de OpcionCombo que está formado por los canales de OpcionMusica y OpcionSeries), y que en un futuro se pueden añadir y/o eliminar canales en los distintos paquetes, y que esos cambios deberían reflejarse automáticamente en el resto de los paquetes que los contiene? **Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta. Indica además el tipo de patrón, y si el patrón utilizado tiene varias versiones justifica cual utilizarías. (1,5 puntos)**

2. Define las clases/interfaces necesarias para aplicar dicho patrón. Representa y justifica como quedaría el diagrama de clases. **(2 puntos)**

3. Reimplementa el método **TelePlusApp.opciones()** de acuerdo con el nuevo diseño. **(1 punto)**

TelePlus acaba de llegar a un convenio de colaboración con la plataforma TeleStar, lo que les permite incorporar sus canales a su parrilla de canales propios. El problema está en que TeleStar dispone de su propia representación de los canales, tal y como muestra el siguiente diagrama de clases:



Como se puede ver la representación no es igual, aunque sí bastante similar. En este caso existe una clase abstracta **ChannelTeleStar** que modela la parte común de cada canal: su **id** (el nombre) y su **type** (su tipo). Sin embargo, por cada tipo de canal existe una subclase concreta. Por ejemplo, un canal de "series" como "Calle13" será un objeto de la clase **CSeries** y se creará vía: **new CSeries("Calle13")**. Sabemos además que TeleStar **no permite modificar la definición de ninguna de las clases que proporciona** a TelePlus.

(Estas son algunas de las clases de TeleStar, habría más subclases con cada tipo de canal)

4. ¿Qué patrón habría que utilizar para poder utilizar las distintas subclases de **ChannelTeleStar** como otro canal más de TelePlus, es decir, que implemente ICanalTV? Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta. Indica el tipo de patrón y si el patrón utilizado tiene varias versiones justifica cual utilizarías. **(1,5 puntos)**

5. Define las clases/interfaces necesarias para aplicar dicho patrón. Representa **y justifica** como quedaría el diagrama de clases. A modo de ejemplo de uso, añade el canal "Calle13" anterior al paquete OpcionSeries de TelePlus en el método **TelePlusApp.opciones()**. **(2 puntos)**

La empresa TelePlus pretende integrar la aplicación **TelePlusApp** en un formato compatible con las Smart TVs actuales, sin embargo, para poder hacerlo los fabricantes de televisiones le imponen las siguientes restricciones:

i. A la parte "inteligente" de la TV le resulta muy costoso, computacionalmente hablando, la creación de objetos tipo **TelePlusApp**, así que si fuera posible se debería reutilizar siempre que se pueda el mismo objeto aplicación.
ii. Como consecuencia del punto anterior, y para evitar errores de programación en la integración, se debería impedir la creación directa de nuevos objetos de la clase **TelePlusApp**.
iii. La interfaz de usuario de cada fabricante de TVs es diferente, así que se debe proporcionar un acceso global a la aplicación desde el código del proyecto.
iv. La parte inteligente de la TV tiene escasa memoria, no tiene sentido crear la aplicación en memoria hasta el momento en que realmente la active un usuario de la TV.

6. ¿Qué patrón habría que utilizar para poder utilizar **TelePlusApp** con este comportamiento? Selecciona el patrón más adecuado para esta tarea. Justifica tu respuesta ilustrando de qué manera cumple los requisitos expuestos en el párrafo anterior. Indica el tipo de patrón y si el patrón utilizado tiene varias versiones justifica cual utilizarías. **(2 puntos)**