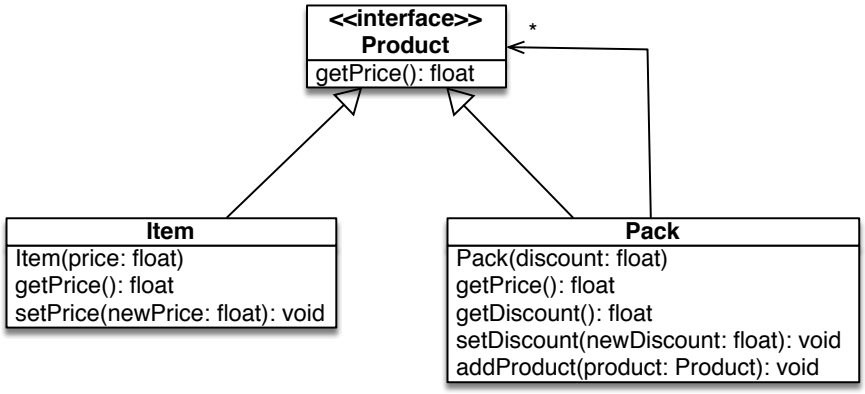
Problema 1. (2 puntos)

¿Qué son los métodos por defecto de una interfaz y por qué se tuvieron que añadir en Java 8?

Problema 2. (8 puntos)

Dado el siguiente diseño de partida:



Donde la clase `Item`:

- representa un producto individual que, en su constructor, recibe el valor del precio que, por simplificar, será un `float`.
- tiene el método `setPrice` para cambiar el precio del producto
- tiene el método `getPrice` que retorna el precio del producto

Y la clase `Pack`

- representa un grupo de productos (que pueden ser tanto `Items` como `Packs`) que, en su constructor decibe el valor del tanto per uno de descuento a aplicar
- tiene el método `setDiscount` para cambiar el tanto por uno de descuento
- tiene el método `addProduct` para añadir un nuevo producto al pack
- tiene el método `getDiscount` que retorna el tant per uno de descuento
- tiene el método `getPrice` que retorna el precio del pack: suma de los precios de los productos descontado sgún el tanto por uno de descuento. Por ejemplo: si los productos tienen precios 10.0 i 20.0 i el tanto per no es 0.2, el preu es (1.0 – 0.2) * (10.0 + 20.0) = 24.0

¿Qué patrón sigue el diseño de estas clases? Implementa las clases involucradas, de manera que

- si en algún método se recibe un precion negativo se lanza la excepción no comprobada `IllegalArgumentException`
- si el descuento no está entre 0.0 i 1.0 se lanza también `IllegalArgumentException`
- **no es necesario comprobar que no hay ciclos de productos** (i.e. que un pack forma un arbre)

Se quieren monitorizar los precios, de manera que es vol saber si un producto, ya sea un `Item` o un `Pack`, ha cambiado de precio.

Para indicar el cambio de precios producido, se dispone de la clase `PriceChanged`

```
1 public final class PriceChanged {
2   private final float oldPrice;
3   private final float newPrice;
4   public PriceChanged(float oldPrice, float newPrice) {
5     this.oldPrice = oldPrice; this.newPrice = newPrice;
6   }
7   public float getOldPrice() { return oldPrice; }
8   public float getNewPrice() { return newPrice; }
9 }
```

Añadid el mecanismo de notificación de cambios de precio, de manera que, cuando el preu cambia, se envía una instancia de la clase `PriceChanged` para indicar el cambio, teniendo en cuanta que **solamente se ha de notificar si el precio ha cambiado realmente.** Mostrad los cambios en las implementacions de `Product`, `Item` i `Pack`.