Importante:
- NO SE PUEDEN CREAR CLASES ADICIONALES.
- NO SE PUEDEN AÑADIR MÁS MIEMBROS DE LOS QUE SE PIDEN.
- En algunos ejercicios, disponemos de varios caminos para resolverlos. Para que la solución se considere correcta, no bastará con que 'funcione' desde el punto de vista del usuario; deberá ser la más adecuada.

1. Tenemos artículos que se identifican por un id automático numérico único (¡¡empieza en 0!!), y tienen un nombre y un precio. El nombre y el precio sí podrían ser los mismos en diferentes artículos. Ya depende del usuario que no enrede las cosas.
En la clase '*Articulo*':

1.1. Añade un miembro dinámico '*catalogo*' con todos los artículos existentes (ten en cuenta el punto 1.2). En él, se irán añadiendo los artículos a medida que se crean. Añade también un método 'muestraCatalogo' que muestre por pantalla su contenido, de esta forma **(1p)**

```
Catálogo:
      (0) Mesa de comedor 187,25€
      (1) Estantería 15,00€
      (2) Banqueta de baño 45,20€
      (3) Estantería 7,50€
      ...
```

Para facilitar las cosas, no se contempla la posibilidad de eliminar artículos del catálogo.

1.2. Completa el método '*recuperaArticulo(id)*'. Debe devolver el artículo asociado a ese id. Si el id solicitado no es válido debe devolver 'null'.**(1p)**

2. Queremos disponer de almacenes. Cada '*Almacen*' se identifica por el nombre de la ciudad en la que está. No puede haber más de un almacén en la misma ciudad.
En la clase '*Almacen*':

2.1. Añade a cada almacén un componente dinámico '*reservas*' que permita almacenar la cantidad que tiene de cada artículo. Cuando se crea un almaceń, todavía no tiene reservas. Las reservas de un almacén se deben mantener siempre ordenadas por el id de artículo.**(1p)**

2.2. Añade un componente llamado '*red*' (red de almacenes) para que contenga todos los almacenes que se vayan creando sin permitir más de un almacén por cada ciudad, manteniéndolos siempre ordenados por la secuencia de creación.**(1p)**

2.3. Completa el constructor de '*Almacen*' y añade un pseudoconstructor '*nuevoAlmacen*' que devuelva true/false, y haz los cambios necesarios en la clase para que si intentamos crear un almacén (mediante el pseudoconstructor) en una ciudad que ya tiene, entonces no lo añada a la red y devuelva false. Si lo puede añadir, que devuelva true.**(1p)**

2.4. Crea el método de clase '*muestraRed()*', que muestra la red de almacenes: **(1p)**

```
Red de almacenes:
      1-Sevilla
      2-Huelva
      …
      7-Madrid
```

El número corresponde al orden de inserción.

2.5. Completa el método de instancia '*recibe(id,cantidad)*'. Debe rechazar (devolver false) cantidades no positivas y también id que no se correspondan con ningún artículo. Si el id y la cantidad son correctas, deberá incrementar las reservas de ese artículo en el almacén y devolver true.**(1p)**

2.6. Crea el método de instancia '*muestraExistencias()*' que mostrará las reservas del almacén: **(1p)**

```
Reservas de Huelva:
      (0) Mesa de comedor 187,25€ → 20ud
      (3) Estantería 7,50€ → 3ud
      ...
```

3. Completa la clase genérica '*Contenedor*' para que nos permita almacenar cualquier tipo de datos (homogéneo, claro; cada contenedor almacenará un tipo diferente). Completa también las declaraciones de las variables *contenido* y *contenido2* al final del método *pruebasContenedor* en la clase *Main*.**(2p)**
Los contenedores tendrán un nombre y un contenido. Almacenarán su contenido sin orden y sin permitir objetos repetidos (es decir no se puede almacenar en un contenedor dos veces el mismo objeto). Obviamente, al construirlos se crean sin contenido alguno.
Permitirán las acciones:
- guardar. Permite guardar un objeto del tipo para el que se ha definido ese contenedor. Devuelve false si se intenta guardar un elemento null o repetido. Devuelve true en caso contrario.
- sacar. Devuelve un elemento al azar de entre el contenido (y lo elimina del contenedor).
- sacar todo. Devuelve todo el contenido (y lo elimina del contenedor), sin orden concreto.
Los tres son métodos de instancia.

EJEMPLO DE EJECUCIÓN

```
.......Creo unos cuantos artículos y muestro el catálogo........
Catálogo:
       (0)Mesa de comedor - 187,25€
       (1)Estantería - 15,00€
       (2)Banqueta de baño - 45,20€
       (3)Estantería - 7,50€

.......Recupero un artículo válido y lo muestro por pantalla........
(2)Banqueta de baño - 45,20€

.......Recupero un artículo NO válido y lo muestro por pantalla........
null

.......Creo unos cuantos almacenes y muestro la red........
true  |  false  |  true  |  true  |  true  |  false
Red de almacenes:
      1-Sevilla
      2-Huelva
      3-Málaga
      4-Cáceres

.......Recupero un almacén, recepciono artículos y muestro almacén........
true  |  false  |  false  |  true  |  true
Reservas de Huelva:
       (0)Mesa de comedor - 187,25€ -> 4ud
       (2)Banqueta de baño - 45,20€ -> 2ud

.......Creo un contenedor de Artículos........
.......Guardo dentro varios artículos........
true  |  false  |  true  |  false  |  true  |  true
En el contenedor c1 hay 4 elementos
       (6)Arroz - 3,00€
       (4)Piruleta - 1,00€
       (7)Gel - 4,00€
       (5)Vino - 2,00€
.......Y saco uno de ellos........
(6)Arroz - 3,00€
.......Esto es lo que queda dentro........
En el contenedor c1 hay 3 elementos
       (4)Piruleta - 1,00€
       (7)Gel - 4,00€
       (5)Vino - 2,00€
.......Saco los restantes y muestro lo que he sacado........
[(4)Piruleta - 1,00€, (7)Gel - 4,00€, (5)Vino - 2,00€]
.......Saco los restantes(NADA) y muestro lo que he sacado(NADA)........
[]

Process finished with exit code 0
```