Contenido

Nota: Estaré comentando el programa tal y como lo hice en Kotlin, en el caso de que haya algo distinto en Java lo destacaré, todo lo demás es seguro afirmar que se mantuvo como mínimo, el razonamiento igual. Además, muchas veces he reiniciado el programa para hacer este escrito, por eso ciertas fotos pueden discrepar entre ellas.

Creación de la cesta:

Como se dijo en clase, la creación de los diversos objetos ("cesta", "usuario", "producto" y "listaCesta") será con sus propios métodos de la clase Factory de manera aleatoria, a excepción de "listaCesta" ya que no tiene valores asignables a la hora de ser creado.

La "cesta" tiene un id, una fecha de creación, una "listaCesta" y un "usuario". El "usuario" tiene un id y un nombre. El "producto" tiene un id, un nombre, y un precio unitario. Por último, "listaCesta" tiene 2 arrays de 20 posiciones cada uno, 1 array contiene el objeto "producto" (aunque está inicializado con nulos), el otro array contiene Strings (al principio vacíos) que representan la cantidad de un producto y su precio unitario con el patrón "cantidad-precioUnitario", está hecho de la misma manera tanto en Kotlin como en Java. La posición de un array coincidirá con la posición del otro para sacar información o introducirla.

Nota: Me parece importante destacar, que al principio, en Kotlin, trate de meter todo el contenido de los 2 arrays en 1 solo que fuera un array de Triple<producto, Int, Double>, pero una vez inicializado, no me dejaba cambiar el contenido, por lo que recurrí a la idea hablada arriba.

Una vez iniciamos el programa, se crea una "cesta" aleatoria, con un "usuario" aleatorio y la "listaCesta" con sus 2 arrays vacíos.

Menú principal:

El menú principal con el que trabajará el usuario es el siguiente:

```
*********************************************************
*                          MENÚ                         *
*********************************************************
* [1] Añadir producto a la cesta                        *
* [2] Eliminar producto de la cesta                     *
* [3] Actualizar cantidad de producto en la cesta       *
* [4] Obtener el total de dinero de la cesta            *
* [5] Mostrar la lista de la cesta                      *
* [0] Salir                                             *
*********************************************************
```

Al la hora de introducir la opción deseada en el menú, como con cualquier otro momento en el que debas introducir información en el programa, hay funciones de filtrado que se encargan de evitar que pase algún dato inválido:

```
**************************************************
*                      MENÚ                      *
**************************************************
* [1] Añadir producto a la cesta                 *
* [2] Eliminar producto de la cesta              *
* [3] Actualizar cantidad de producto en la cesta *
* [4] Obtener el total de dinero de la cesta     *
* [5] Mostrar la lista de la cesta               *
* [0] Salir                                      *
**************************************************

La opción introducida no es válida, vuelve a probar:
```

1.Añadir producto:

Lo primero que pasa al seleccionar está opción, es que se busca si hay algún espacio libre en el que se podría meter un nuevo producto, si no lo hay nos saca de la opción con un mensaje diciendo que no queda espacio libre, si queda hueco libre, pasa al siguiente paso. Procede a crear un nuevo objeto de forma aletoria, una vez creado, busca en el array si ya existe un producto con el mismo nombre, si lo hay devuelve la posición donde está en el array, y el producto de esa posición actualiza en +1 su cantidad, lo que se muestra en un mensaje, en caso de no haber ningún producto con el mismo nombre, el producto se almacena en los arrays, en la posición vacía que se encontró anteriormente y con cantidad = 1 y precio unitario = producto.precioUnitario, con un mensaje que lo destaca.

La función de añadir un producto a los arrays, coje la posición vacía y el producto que se creo, mete en el array de “producto” al nuevo producto que se creó en la posición vacía, y en el otro array, en la misma posición cambia el String de vacío a los a cantidad = 1 y el precio unitario del producto con el patrón que ya comente (“cantd-precioU”).

```
**************************************************
* [1] Añadir producto a la cesta                 *
* [2] Eliminar producto de la cesta              *
* [3] Actualizar cantidad de producto en la cesta *
* [4] Obtener el total de dinero de la cesta     *
* [5] Mostrar la lista de la cesta               *
* [0] Salir                                      *
**************************************************
1
Se va a añadir a la cesta el producto: garra
**************************************************
*                      MENÚ                      *
**************************************************
* [1] Añadir producto a la cesta                 *
* [2] Eliminar producto de la cesta              *
* [3] Actualizar cantidad de producto en la cesta *
* [4] Obtener el total de dinero de la cesta     *
* [5] Mostrar la lista de la cesta               *
* [0] Salir                                      *
**************************************************
1
El producto ha añadir a la cesta: gato, ya existe en la lista, por lo que se le va a sumar la cantidad en 1, por lo que ahora es: 2
```

2.Eliminar producto:

Empieza calculando si hay algún producto en la lista de la cesta, en caso negativo te saca directamente al menú:

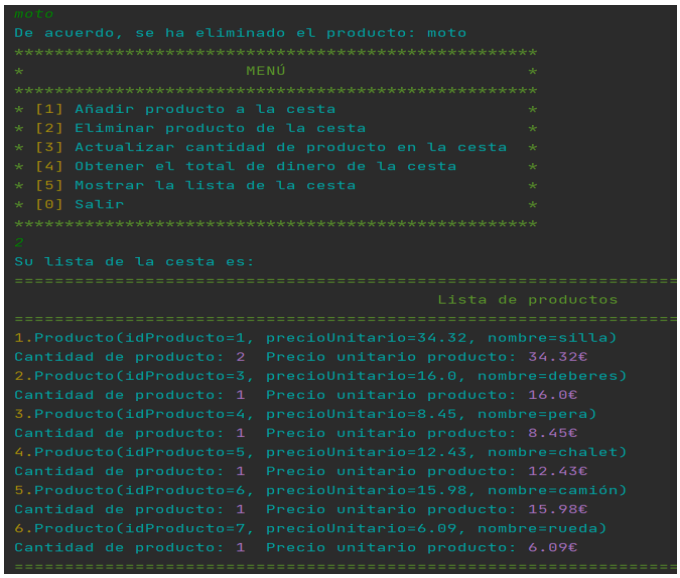
```
**************************************************
*                    MENÚ                        *
**************************************************
* [1] Añadir producto a la cesta                 *
* [2] Eliminar producto de la cesta              *
* [3] Actualizar cantidad de producto en la cesta *
* [4] Obtener el total de dinero de la cesta     *
* [5] Mostrar la lista de la cesta               *
* [0] Salir                                      *
**************************************************
2
Su lista de la cesta es:
Aun no hay nada en el carrito
Como está vacía, no hay productos a eliminar
**************************************************
*                    MENÚ                        *
**************************************************
* [1] Añadir producto a la cesta                 *
* [2] Eliminar producto de la cesta              *
* [3] Actualizar cantidad de producto en la cesta *
* [4] Obtener el total de dinero de la cesta     *
* [5] Mostrar la lista de la cesta               *
* [0] Salir                                      *
**************************************************
```

En caso contrario, prosigue, primero mostrando un listado no ordenado de los productos de la cesta y pidiendo al usuario que introduzca el nombre del producto que desea eliminar de la cesta. Una vez se introduce un nombre válido (o “quit”, lo que te devuelve al menú), se busca alguna coincidencia entre el nombre introducido y el de algún producto, si no la hay, la función que busca la posición según el nombre devuelve un -1 y se te pide volver a introducir un nombre:

```
2
Su lista de la cesta es:
================================================================================
                                    Lista de productos
================================================================================
1.Producto(idProducto=1, precioUnitario=34.32, nombre=silla)
Cantidad de producto: 2  Precio unitario producto: 34.32€
2.Producto(idProducto=3, precioUnitario=16.0, nombre=deberes)
Cantidad de producto: 1  Precio unitario producto: 16.0€
3.Producto(idProducto=4, precioUnitario=8.45, nombre=pera)
Cantidad de producto: 1  Precio unitario producto: 8.45€
4.Producto(idProducto=5, precioUnitario=12.43, nombre=chalet)
Cantidad de producto: 1  Precio unitario producto: 12.43€
5.Producto(idProducto=6, precioUnitario=15.98, nombre=camión)
Cantidad de producto: 1  Precio unitario producto: 15.98€
6.Producto(idProducto=7, precioUnitario=6.09, nombre=rueda)
Cantidad de producto: 1  Precio unitario producto: 6.09€
7.Producto(idProducto=8, precioUnitario=8.45, nombre=moto)
Cantidad de producto: 1  Precio unitario producto: 8.45€
================================================================================
Introduce el nombre del producto que quieras eliminar, o "quit" si quieres salir:
agua
No Existe ningún producto con ese nombre, vuelve a probar:
```

En caso de si encontrar una coincidencia, devuelve la posición del producto a eliminar, el cual es completamente borrado de los registros.

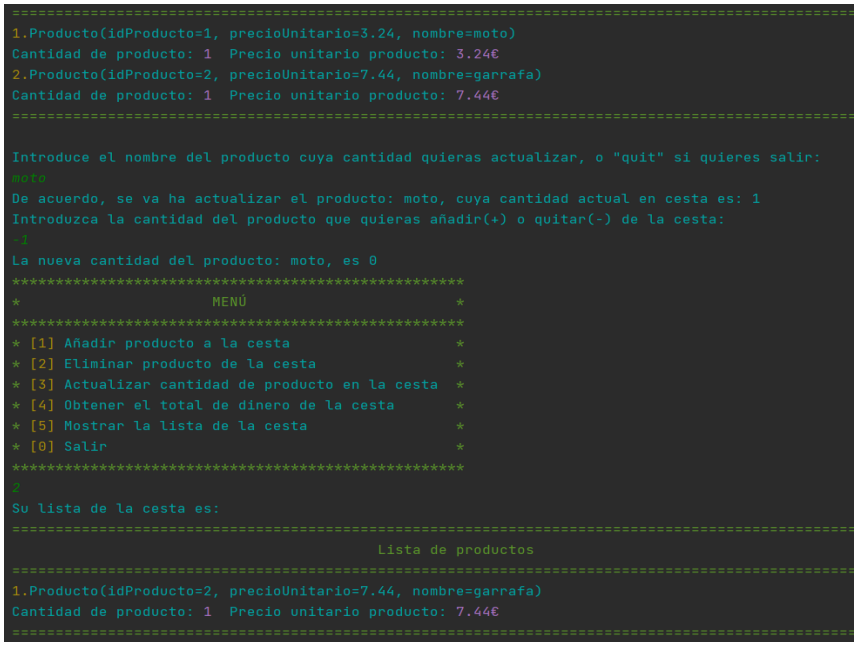
La función de eliminar un producto, pone a nulo y con un mensaje vacío la posicione dadas por parámetro en el array correspondiente. La posición dada por parámetro es la posición del producto buscada según el nombre.



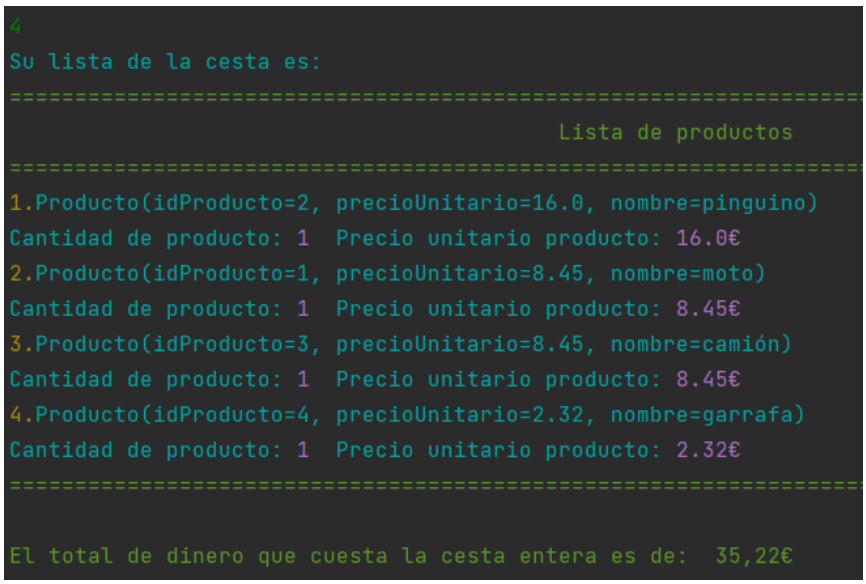
3.Actualizar producto:

La actualización (de la cantidad de un producto) funciona exactamente igual que la función de eliminar, excepto por que la función actualizar no elimina un producto sino que cambia un dato asociado al mismo, pero todo lo de mostrar la lista de productos, y seleccionar según el nombre es igual.

La función de actualizar la cantidad funciona de manera que si una cantidad es la máxima posible (es decir, 10) podrás solamente quitar cantidad del producto, en caso de que la cantidad este entre 1 y 9, podrás añadir como máximo más 10 o menos 10, de manera que si te pasas de cantidad 10, se podrá automáticamente a 10, y se le pones menos de 0 se pondrá automáticamente a 0. Un dato curioso es que tras actualizar tengo una función que elimina cualquier producto cuya cantidad sea 0:



4.Obtener total de dinero:

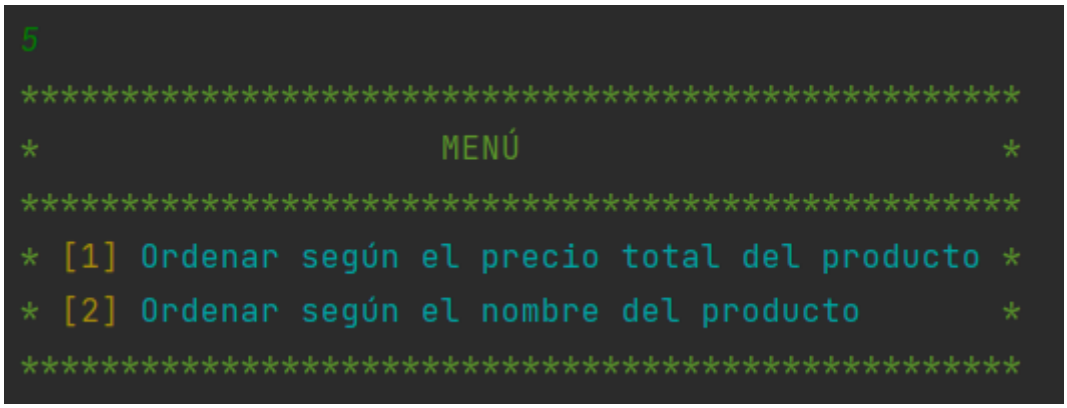


Como vemos en la foto, muestra la lista de productos (si no hubiera productos, saltaría al menú directamente), y un mensaje con el total de la cesta que es un número con dos decimales, por lo que use un String.format() para representarlo.

El dinero se calcula, sumando el resultado de la cantidad de un producto por su precio unitario.

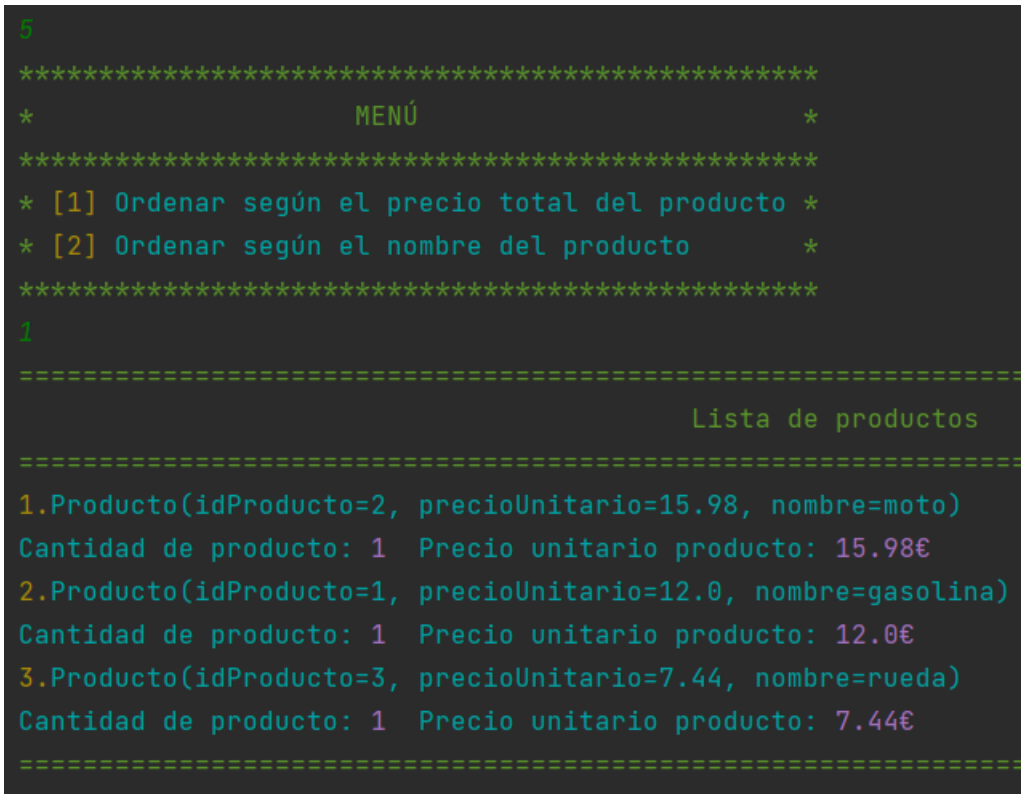
5.Mostrar la lista de la cesta:

Por último, tenemos la función de representar la lista, pero está cuenta con dos opciones posibles (en caso de que no hubiera productos, nos llevaría directamente al menú principal):

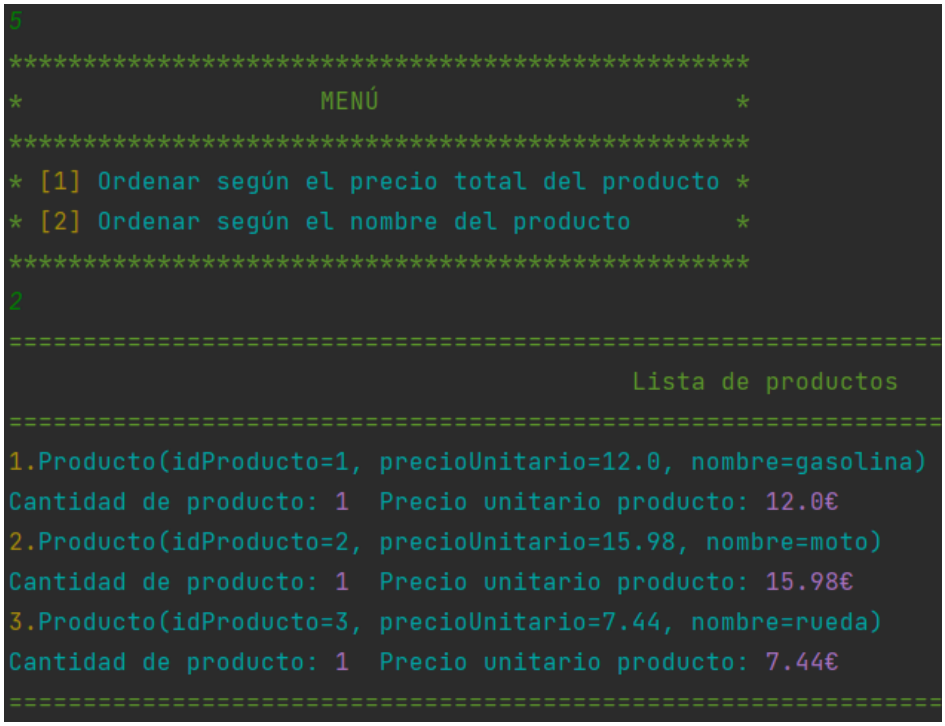


Como vemos en el menú, hay dos opciones, o lo ordenamos de mayor a menor según el resultado de la cantidad por precio unitario, o lo ordenamos de la ‘a’ a la ‘z’ (para lo que use un compareTo).

Primero, como se vería con la primera opción:



Ahora como se vería la misma lista, con la segunda opción:

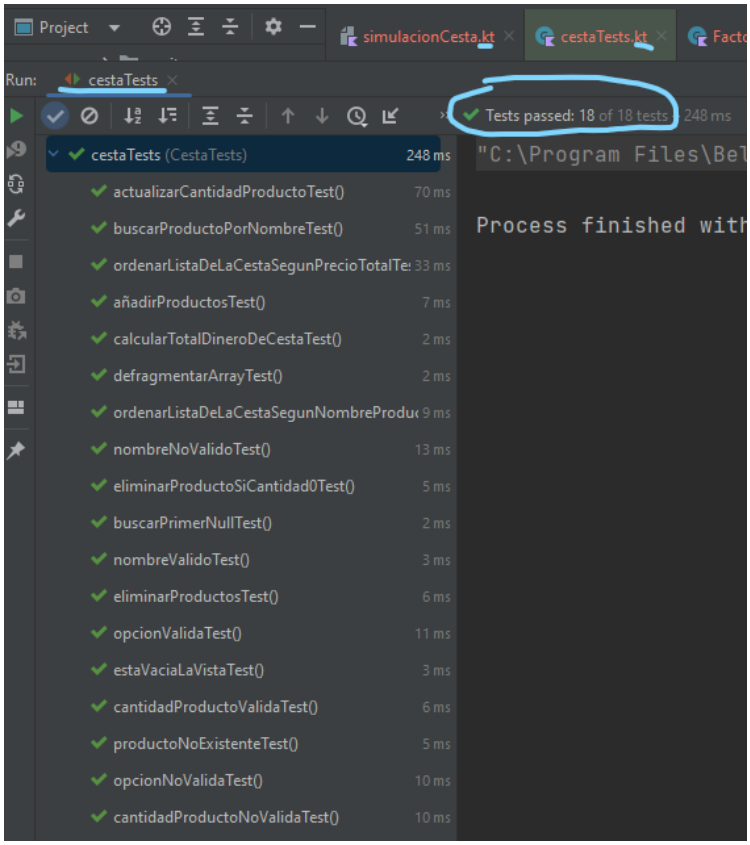


Nota: En todos los momentos donde represento la lista de productos, siempre le paso primero la función de defragmentar los arrays, para que los índices quedén mejor, la prueba es que los números puestos en amarillo al inicio de cada producto, es el índice+1.

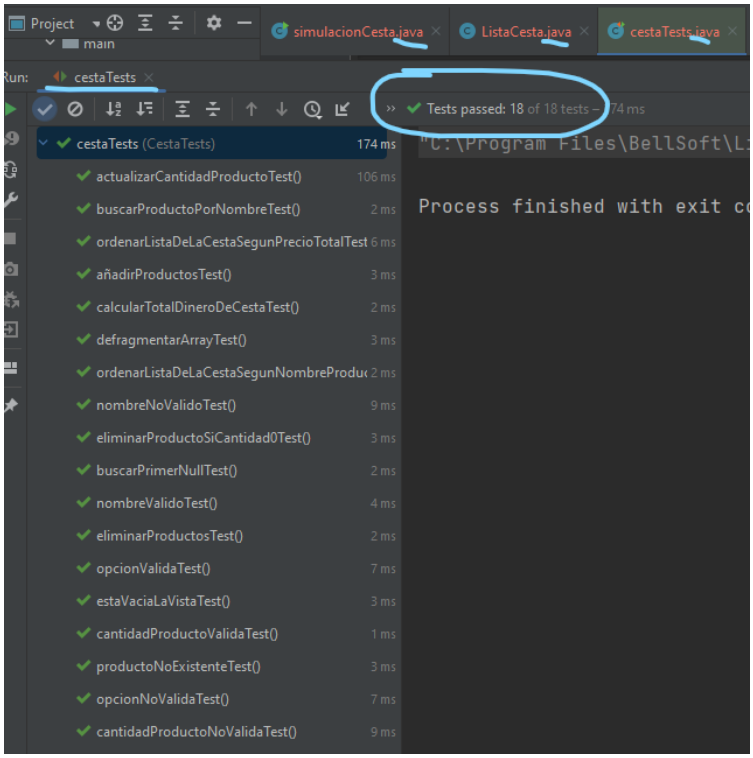
Los Tests:

Lo primero es lo primero, hay que mostrar que los tests pasan:

Kotlin:



Java:



Hay pocos, pero hay algún test que sirve para válidar si el dato introducido está bien, o si no lo está, en cuyo caso se debe de comprobar si se está dando el mensaje adecuado al haber errado.

También tenemos los test que comprueban las funciones de ordenar los array, y el test de defragmentar los arrays.

Por otro lado, podemos destacar los tests que comprueban que los productos se añaden, eliminan y actualizan correctamente.

Además están los test de buscar posición según nombre del producto, buscar la primera posición vacía, si los arrays están vacíos.

Y por último, también está el tests de la función de calcular el dinero total de la cesta.