

(SIMULACIÓN CESTA) PROGRAMACIÓN ORIENTADA A OBJETOS BÁSICA

Angel Maroto Chivite
23/01/2023

Índice

Contenido

Índice	1
1. Java & Kotlin	2
2. Ejecución de la simulación	2
2.1. Instanciar objetos	2
2.2 Menú	2
2.2.1 Listar Nombre Ascendente	2
2.2.2 Listar Precio Descendente	3
2.2.3 Nueva Cesta	3
2.2.4 Modificar Productos	3
2.3 Lógica Cesta	3
3. Pruebas unitarias	4
4. Diseño implementado	5
4.1 Factories Class	5
4.2 Models Class	5

1. Java & Kotlin

Primero realicé la práctica en Java.

Para la versión de Kotlin me he apoyado del conversor del IntelliJ, a la par que debía ajustar diversos errores que iban surgiendo.

2. Ejecución de la simulación

2.1. Instanciar objetos

Instanciamos la Cesta Final gracias a mis factories, está compuesta de una Línea de Cesta intermedia para una asistencia al Usuario, si en el futuro se cambia el precioUnitario de los productos que dispongamos en nuestras bases de datos y cada Línea de Cesta, está formada de todos los productos que podrá ir manejando nuestro usuario limitadamente.

Posteriormente se ejecutará un menú:

2.2 Menú

Disponemos de dos formas de listar:

2.2.1 Listar Nombre Ascendente

Con un orden de nombre ascendente, es decir, de la A a la Z:

```
=====
===== |CESTA 1| |CREADA = 2023-01-23| =====
=====
Product{idProduct=26, name='Azúcar', unitaryPrice=1.81, quantity
Product{idProduct=30, name='Canónigos', unitaryPrice=1.5, quanti
Product{idProduct=23, name='Cereal', unitaryPrice=0.61, quantity
Product{idProduct=21, name='Chocolate', unitaryPrice=0.7, quanti
Product{idProduct=25, name='Harina', unitaryPrice=0.98, quantity
Product{idProduct=29, name='Manzana', unitaryPrice=0.43, quantit
Product{idProduct=28, name='Miel', unitaryPrice=0.47, quantity=2
Product{idProduct=27, name='Sal', unitaryPrice=0.87, quantity=2,
Product{idProduct=22, name='Zanahoria', unitaryPrice=1.29, quant
Product{idProduct=24, name='Zumo', unitaryPrice=0.51, quantity=4
=====
=====
```

2.2.2 Listar Precio Descendente

Con un orden de precio descendente, es decir, de 9-0:

```
=====
===== |CESTA 1| |CREADA = 2023-01-23| =====
=====
Product{idProduct=30, name='Canónigos', unitaryPrice=1.5, quantity=4, Total= 6.0}
Product{idProduct=22, name='Zanahoria', unitaryPrice=1.29, quantity=4, Total= 5.16}
Product{idProduct=25, name='Harina', unitaryPrice=0.98, quantity=4, Total= 3.92}
Product{idProduct=26, name='Azúcar', unitaryPrice=1.81, quantity=2, Total= 3.62}
Product{idProduct=21, name='Chocolate', unitaryPrice=0.7, quantity=5, Total= 3.5}
Product{idProduct=29, name='Manzana', unitaryPrice=0.43, quantity=5, Total= 2.15}
Product{idProduct=24, name='Zumo', unitaryPrice=0.51, quantity=4, Total= 2.04}
Product{idProduct=23, name='Cereal', unitaryPrice=0.61, quantity=3, Total= 1.83}
Product{idProduct=27, name='Sal', unitaryPrice=0.87, quantity=2, Total= 1.74}
Product{idProduct=28, name='Miel', unitaryPrice=0.47, quantity=2, Total= 0.94}
=====
Cantidad de productos a comprar: 35
```

2.2.3 Nueva Cesta

Se aumentará un contador para generar una nueva cesta, desechando la anterior y empezar la simulación con la nueva.

2.2.4 Modificar Productos

Podemos seleccionar un producto y elegir añadir cantidad, restar cantidad o eliminarlo, estando las entradas filtradas sin error de ejecución

2.3 Lógica Cesta


La Cesta calculará en todo momento por cualquier cambio que se realice a los productos, el total que nos contará la cesta actual y la cantidad de productos que disponemos:

```
Product{idProduct=28, name='Miel', unitaryPrice=0.47, quantity=2,
=====
Cantidad de productos a comprar: 35
Por un precio de: 30.9 €
=====
```

Por cada acción se imprimirá la cesta para tener control sobre ella.

3. Pruebas unitarias

El código de impresión en pantalla no se encuentra probado, y tampoco alguna función puntual que se me ha complicado de más, el resto del código está probado:

Element ^	Class, %	Method, %	Line, %
▼  simulacionCesta	88% (8/9)	76% (26/34)	48% (103/214)
▼  factories	100% (3/3)	100% (4/4)	100% (18/18)
 CestListFactory	100% (1/1)	100% (2/2)	100% (11/11)
 FinalCestFactory	100% (1/1)	100% (1/1)	100% (3/3)
 ProductFactory	100% (1/1)	100% (1/1)	100% (4/4)
▼  models	100% (3/3)	100% (14/14)	100% (42/42)
 CestList	100% (1/1)	100% (3/3)	100% (6/6)
 FinalCest	100% (1/1)	100% (6/6)	100% (25/25)
 Product	100% (1/1)	100% (5/5)	100% (11/11)
▼  utils	100% (2/2)	53% (8/15)	28% (43/152)
 funcionesMenuCesta	100% (1/1)	41% (5/12)	22% (32/141)
 randomData	100% (1/1)	100% (3/3)	100% (11/11)

Para la versión en Java he utilizado el gestor de dependencias Maven mediante la dependencia de Test Junit5.

Para la versión de Kotlin he utilizado el gestor de dependencias Gradle mediante la dependencia de Test Junit5.

4. Diseño implementado

4.1 Factories Class

Creación de objetos aleatorios mediante factories, que se agregan a una Lista de Cesta, siendo esta una entidad intermedia, para posteriormente introducirla en nuestra Cesta Final.

Esta implementación de la Línea de Cesta intermedia es debida, por si se realiza algún cambio en el precio unitario de un producto, el cliente sigue manteniendo el precio con el que realizaba su compra.

4.2 Models Class

Objetos que instanciaré a lo largo de la ejecución del programa.