Trabajo Práctico 1 — Smalltalk

[7507/9502] Algoritmos y Programación III
Curso 1
Segundo cuatrimestre de 2020

Alumno:	Vilardo, Ezequiel
Número de padrón:	104980
Email:	evilardo@fi.uba.ar

Índice

1. Introducción

El presente informe reune la documentación de la solución del primer trabajo práctico de la materia Algoritmos y Programación III que consiste en desarrollar una aplicación de un sistema de pedidos de productos en Pharo utilizando los conceptos del paradigma de la orientación a objetos vistos hasta ahora en el curso.

2. Supuestos

2.1. Utilización de Cupones

Podrán asignarse múltiples cupones con diferentes porcentajes o valores de descuento, pero el cupón que finalmente se aplique (siempre y cuando se cumplan las condiciones establecidas en la consigna) será el ultimo introducido. Ademas, en el caso del Cupon por valor, si su descuento es mayor al precio total el precio del pedido sera 0, no importa cuanto mayor al precio final sea, no se le pagara al cliente.

2.2. Restriccion a cantidades negativas

No podrán asignarse cantidades, precios o porcentajes negativos a los productos, cupones u objetos.

2.3. Podran asignarse multiples productos con el mismo nombre

Podrán asignarse dos productos con el mismo nombre, estos permanecerán separados y en caso de modificar la cantidad, se modificará la cantidad del primero ingresado, en caso de ser removido esto también afectará al primero de los productos ingresados.

2.4. Podran asignarse multiples menus

Podrán asignarse sin inconvenientes múltiples menús sin inconvenientes, alguien podría solicitar en su pedido el menú del lunes, el del jueves y el del domingo en simultaneo.

2.5. Descuento y recargo por delivery

En los casos de pedidos con delivery en los que se pueda aplicar el cupón de descuento, primero se aplicara el descuento del cupón y luego se aplicara el recargo por el costo del delivery.

3. Diagramas de clase

A continuación inserto el diagrama de clases, en el puede observarse que la clase AlgoPedidos tiene un asociación con la clase abstracta Pedido, la instancia de AlgoPedidos puede no tener ninguna instancia de los hijos de Pedido como puede tener muchas. Además vemos que la clase Pedido tiene 3 asociaciones más, una con la clase abstracta Cupón, de la que siempre contara con una instancia de su hijos y las otras asociaciones son con las clases Producto y Menú de las que tendrá muchas instancias o ninguna.

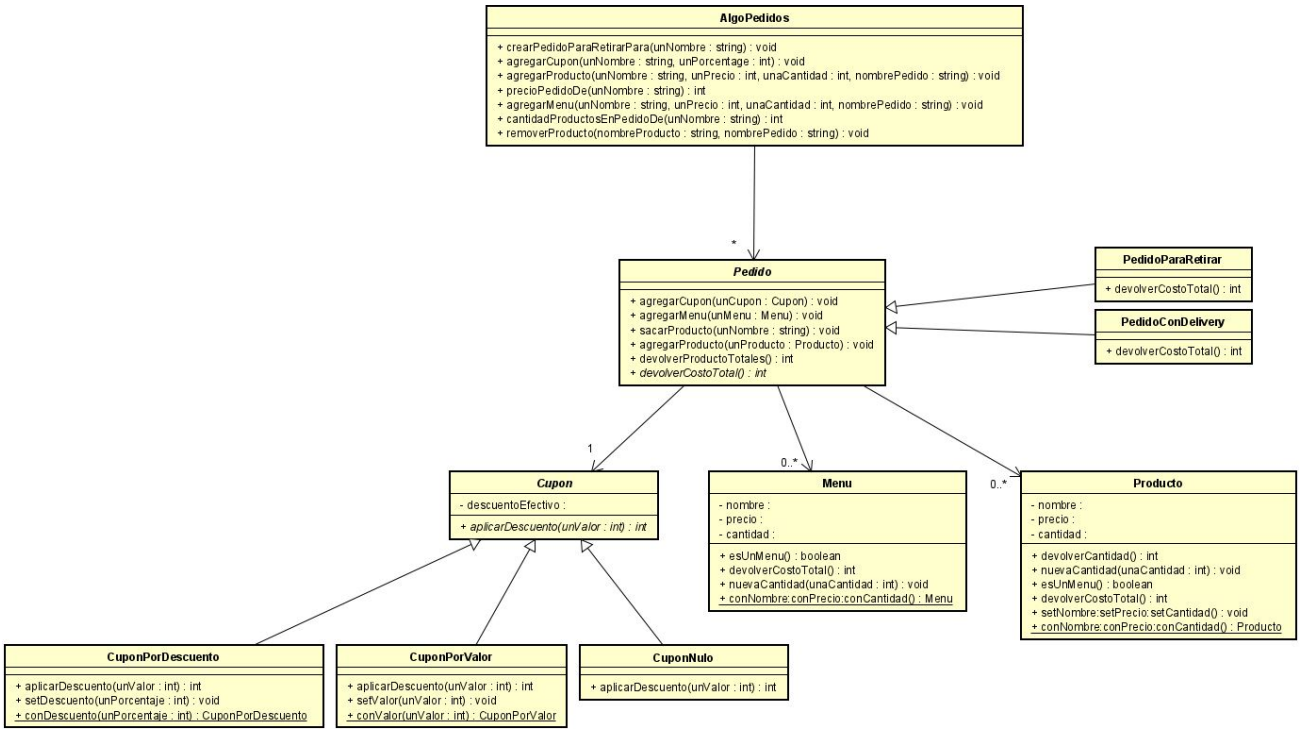

Figura 1: Diagrama de clases 1.

Además agrego otro diagrama de clases donde muestro, las relaciones de dependencia que tiene cada excepción con las clases mostradas en el diagrama anterior, estas excepciones serán explicadas mas adelante en el informe.

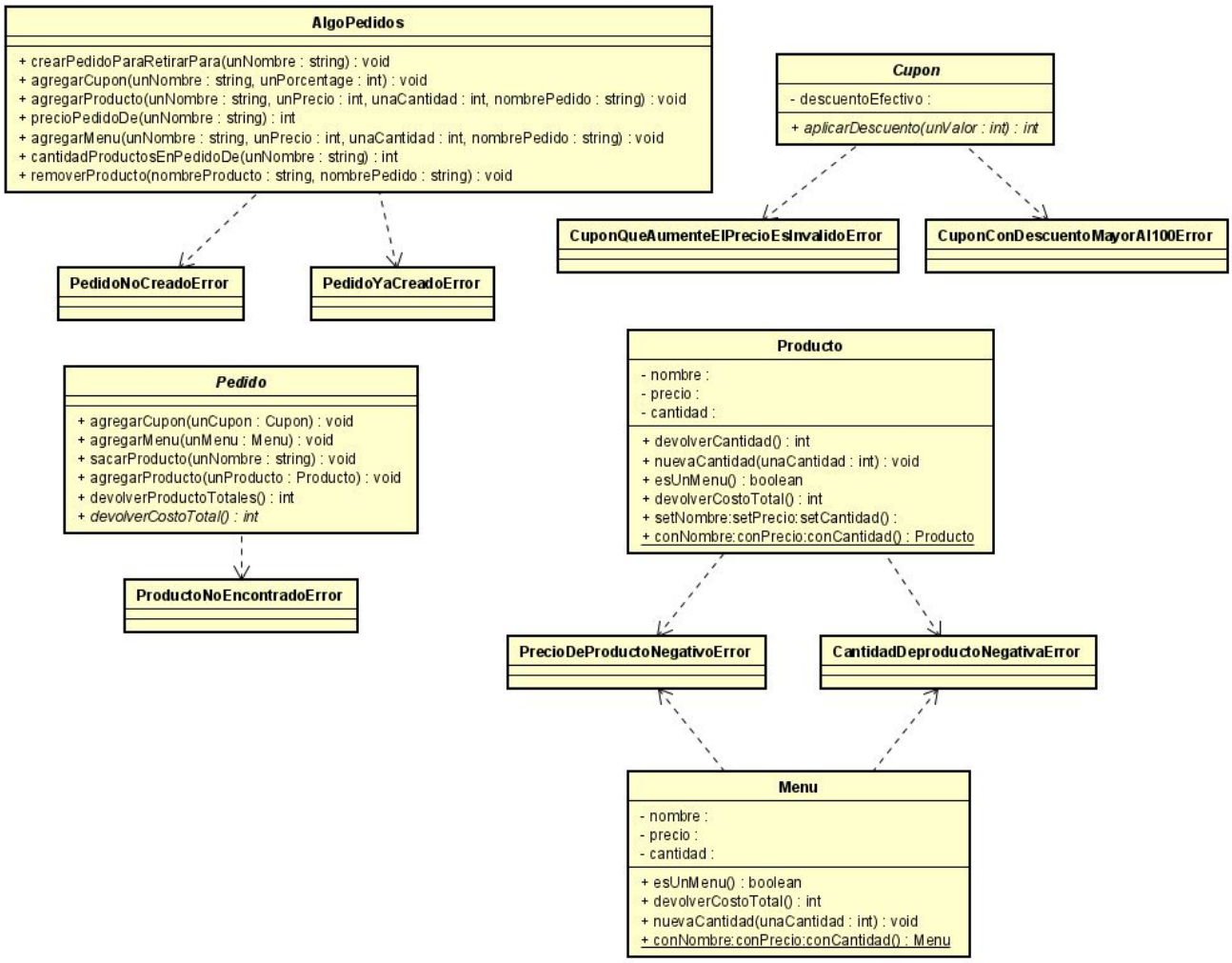

Figura 2: Diagrama de clases (Excepciones que dependen de cada clase).

4. Detalles de implementación

4.1. La Clase Pedido y sus hijos

Podía observarse en el diagrama de clases como PedidoParaRetirar y PedidoConDelivery heredaban de la clases abstracta Pedido, con esto además de buscar la solución polimórfica al problema que se generaba al devolver el costo del pedido, en el caso que el pedido tuviera delivery se aplicaba un recargo, se aplica herencia con la finalidad de reaprovechar el código, esto es posible gracias a que se cumple la condición 'es un'.

Además como detalle, puede observarse las delegaciones que hace la clase, al pasarle los problemas a otras clase, por ejemplo a la hora de hacer el descuento esta le transfiere el valor a alguno de los hijos de Cupón y que esta se encargue del problema, otro ejemplo es a la hora de modificar las cantidades de un Producto esta le pasa la nueva cantidad y este decide como procederá, estos serian ejemplos de la presencia de encapsulamiento dentro de esta clase.

4.2. Cupon, CuponPordescuento, CuponPorvalor y ¿CuponNulo?

Al igual que en el caso anterior, además de buscar la solución polimórfica ya que todos los tipos de cupones que existen en el problema buscaban lo mismo, generar un descuento, en esta situación la herencia tiene un objetivo mas visual que funcional, la herencia no suma ninguna ventaja sobre la opción secundaria de crear directamente las 3 clases hijos sin herencia, no se evita codigo repetido, fue pensado como una especie de interfaz.

Con el CuponNulo buscaba solucionar el problema de los casos en los que no se aplicara un cupón o en caso de que no se cumplieran las condiciones necesarias, el hecho de que existiera un cupón falso que devolviera el valor que se le pasaba sin realizarle cambios simplificaba el problema original que implicaba hacer mas verificaciones dentro de los cupones para saber si deberia o no hacer el descuento.

4.3. Pilares del POO

Con lo mencionado en los párrafos anteriores, pueden observarse la presencia de los principales pilares del paradigma que son: Herencia, Polimorfismo, Abstracción y Encapsulamiento.

El uso de herencia principalmente en Pedido con la finalidad de reutilizar código, siendo que ambos pedidos funcionan igual con una única diferencia en su comportamiento, la ventaja en el uso de la herencia se centra principalmente en la reutilizacion de codigo entre clases.

El de polimorfismo en los cupones ya que no me importaba su tipo solo quería que apliquen su descuento o con Producto y Menú, con cada uno respondiendo a los mensajes a su forma, la principal ventaja del soluciones polimorficas es que al querer ampliar la implementacion se vuelve muchisimo mas sencillo, ya que "solo"hay que agregar un modulo mas.

La abstracción en Producto o Menú, donde solo me interesa su nombre, precio y cantidad, dejando otros aspectos que no tienen relevancia en el problema fuera de discusión y simplificandolo.

Y el Encapsulamiento, puede verse en mucho lugares, en Pedido transfiriendo responsabilidades a Cupón o a Producto evitando así las responsabilidades de aplicar el descuento o de saber cuando vale el producto, así también como AlgoPedidos derivando todo a la instancia especifica de Pedido en la que se busca interactuar, ventajas de esta caracteristica es que a la hora de modificar las implementaciones solo hay que modificar la parte en la que se busca cambiar el comportamiento, al no haber codigo repartido por todos lados esto se vuelve una tarea mucho mas sencilla..

5. Excepciones

5.1. CantidadDeProductoNegativaError

Como el nombre indica, esta excepción es lanzada cuando en el proceso de inicialización de un producto/menu se detecta que la cantidad mandada es negativa, con esta excepción se busca evitar romper la lógica de la implementación, se busca evitar que sea el cliente el que termine vendiéndole al local.

5.2. PrecioDeProductoNegativoError

Al igual que la excepción anterior esta será lanzada durante la inicialización del producto/menu pero en caso de que el precio sea un monto negativo, ya que un precio negativo significaría que el local le paga al cliente por venderle algo.

5.3. PedidoNoCreadoError

Esta excepción aparece cuando clase AlgoPedidos esta buscando si tiene el pedido sobre el cual se quiere trabajar y este no se encuentra dentro del diccionario de pedidos.

5.4. PedidoYaCreadoError

Como la anterior esta aparece en AlgoPedidos, durante el proceso de creación de pedidos se buscar que no exista otro pedido reservado con el mismo nombre, en el caso que si exista la excepción aparece, evitando así que el pedido anterior sea pisado por el nuevo.

5.5. ProductoNoEncontradoError

Esta excepción se lanza en los casos que se esta buscando un producto/menu para eliminarlo o para modificar su cantidad y este no se encuentra en el pedido.

5.6. CuponConDescuentoMayorAl100Error

Durante el proceso de inicialización del cupón, al detectarse un porcentaje de descuento mayor al 100 que implicaría que el local le pagase al cliente hace saltar esta excepción.

5.7. CuponQueAumenteElPrecioEsInvalidoError

Durante el proceso de inicialización del cupón un descuento inferior al 0 porciento o un valor negativo, condiciones que provocarían que el cupón aumentara el precio final en ves de disminuirlo, harían saltar la excepción.

6. Diagramas de secuencia

El diagrama a continuación representa una secuencia en la cual se añade un pedido para retirar, y al se agregan un producto, un menú y un cupón por descuento.

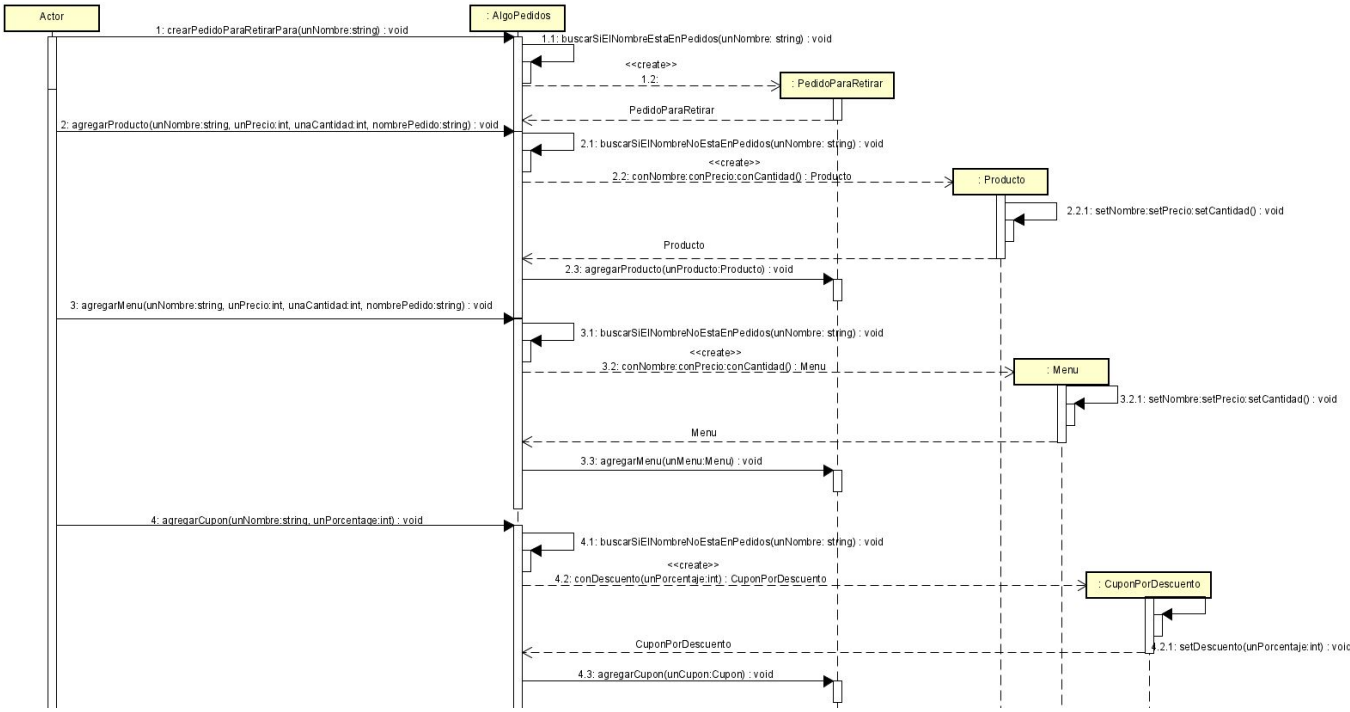

Figura 3: Diagrama 1 parte 1.

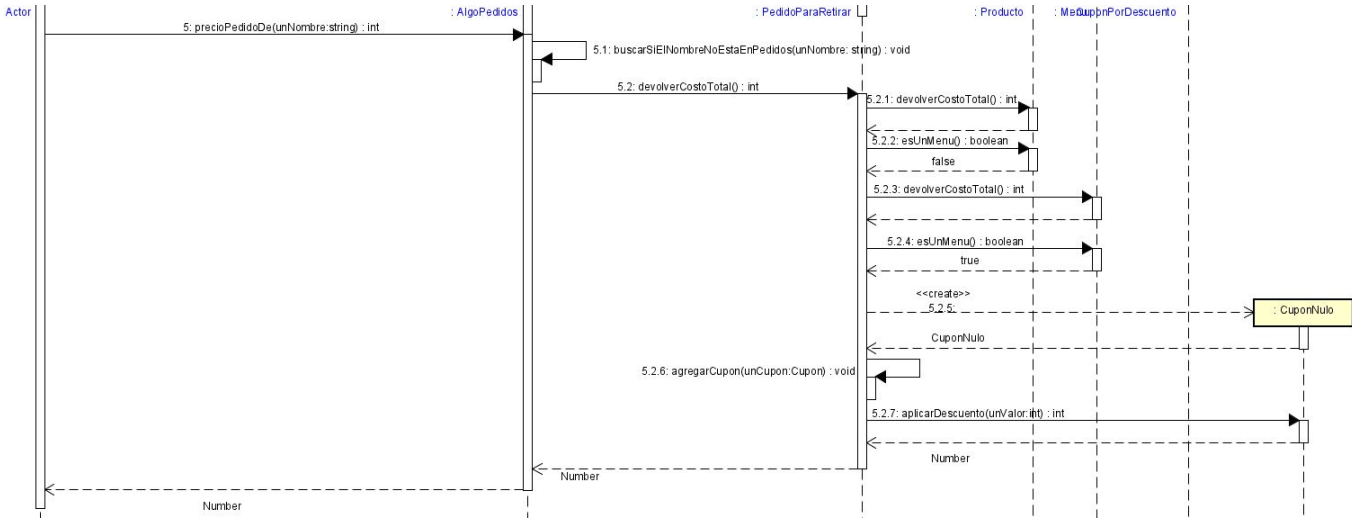

Figura 4: Diagrama 1 parte 2.

En la segunda parte se observa el proceso por el cual se obtiene el precio total del pedido, AlgoPedidos deriva a la instancia especifica de Pedido de la se quiere saber el precio, esta se encargara luego de preguntarle al producto y al menú su costo total y si son un menú, en este caso al encontrase un menú dentro del pedido se crea una nueva instancia de un CuponNulo que se agrega al Pedido y pisa al anterior cupón, luego se le pasa al cupón el precio acumulado y este aplica el descuento (en realidad al ser un CuponNulo devuelve el mismo valor que se le paso) y devuelve el valor a Pedido, este se lo devuelve a AlgoPedidos y este lo devuelve al actor.

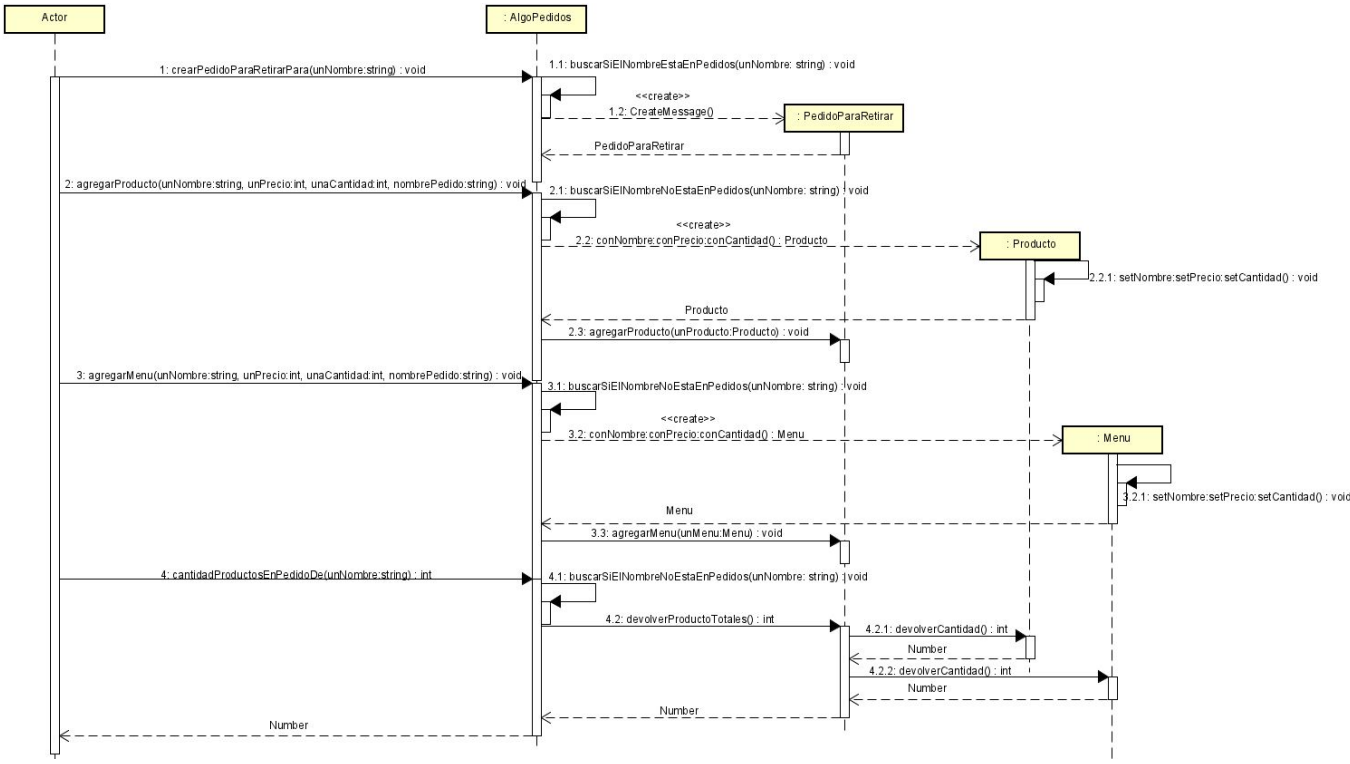

Figura 5: Diagrama 2.

En este segundo diagrama puede observarse nuevamente el proceso por el cual se añade un nuevo pedido para retirar, un producto y un menú, pero a continuación de estos se muestra la secuencia para identificar la cantidad de productos presentes en el pedido. Algopedidos le transfiere el trabajo de preguntarle a los productos su cantidad y una vez le pregunta a todos le devuelve el valor a Algopedidos y este se lo devuelve al actor.

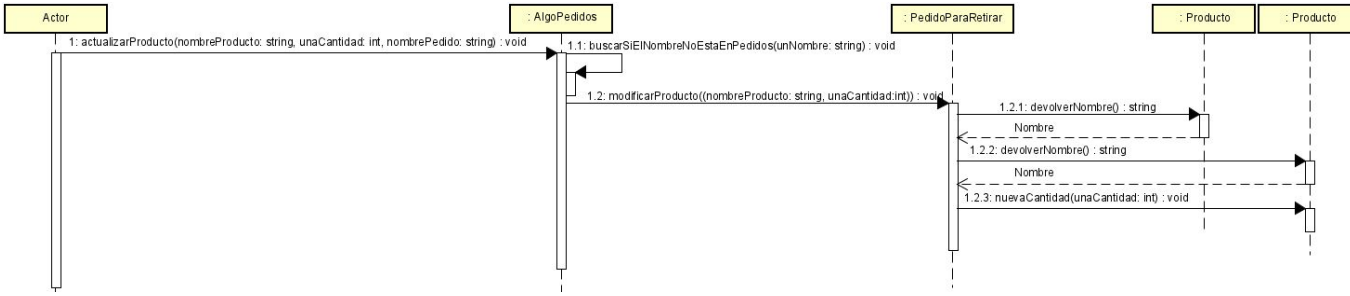

Figura 6: Diagrama 3.

En este ultimo diagrama se representa la secuencia de modificación de un producto, pero se parte de un punto donde ya se cuenta con la instancia de pedido y de dos productos dentro del pedido, estas secuencias ya las había representando en los diagramas anteriores las omito para evitar secuencias repetidas, al igual que en secuencias anteriores Algopedidos deriva a Pedido que se encarga de preguntarle a cada producto su nombre, cuando encuentra el buscado le deriva a este el proceso para que modifique su cantidad.