Guillem Escriba Molto - u188331

David Pérez Carrasco - u188332

Programación Orientada a Objetos

Informe Práctica 5

En esta última práctica cambiamos completamente de tesitura, dejamos de lado el *World* con el que llevábamos trabajando hasta ahora y empezamos a crear un nuevo programa, una tienda online de venta de libros. El objetivo principal era gestionar los stocks de cada libro, permitir al usuario que añada y quite libros, que los compre, y que todo se actualice en tiempo real, descontando las unidades del stock de la tienda y añadiéndolas al carrito. En esta práctica, ya nos daban bastantes clases hechas, por lo que en sí las únicas clases que hemos tenido que crear han sido *Catalog*, *Book, Stock* y *ShoppingCart*. A pesar de solo tener que implementar estas, se nos daba libertad de hacerlo como creyésemos conveniente de la manera más optimizada posible.

La clase Book simplemente tenía unos atributos de tipo String para el título, el autor y el lugar de publicación de cada libro, un atributo de tipo Date para la fecha de publicación y un atributo Long para el ISBN. El constructor de dicha clase simplemente recibía como argumentos los valores iniciales de todos los atributos y se los atribuía y los métodos de esta clase simplemente eran los getters de los distintos atributos.

En cuanto a la clase Stock, los atributos son un Book, un entero para el número de copias, un double para el precio y un Currency para el tipo de moneda, teniendo el constructor el valor de inicial de cada uno de ellos como argumento. Los métodos eran el getBook, el getBookTitle y el numberOfcopies, que simplemente retornaban su respectivo

atributo, el método addCopies que añadía 1 copies y removecopies que lo restaba y finalmente totalPrice que retornaba el precio.

La clase de *Catalog* básicamente es la encargada de guardar los stocks de todos los libros de la tienda, en un inicio, pensamos que estaba vacía en el sentido de que los métodos y el constructor eran tal y como los heredaba, pero tras bastantes problemas al ver que no se inicializaba bien, no compilaba correctamente ni se ejecutaba nos dimos cuenta de que se tenían que modificar el constructor para que inicializase correctamente collection, un HashSet de Stocks de los Books de la tienda que almacena las copias de cada libro y sus datos y es el encargado de mostrar al cliente todos los libros que hay disponibles en cada momento descontando los que se han añadido al carrito. En sí, una vez entendimos lo que hacía esta clase no fue muy difícil de implementar, pero nos costó llegar a ese punto.

Por otro lado está *ShoppingCart* que es, de lejos, la clase más problemática de toda, la práctica, al principio pensamos que era una clase sencilla, quita copias de un sitio y las añade a otro, pero en el momento en el que la empezamos a implementar nos dimos cuenta rápidamente de que no era tan fácil como parecía en un primer momento. El primer problema con el que nos encontramos es que desconocíamos como encontrar el stock de un libro concreto, pero se solucionó cuando vimos que en la superclase, había una función llamada *getStock* que se precisamente de hacer eso pero estaba con visibilidad privada y al desconocer el motivo de esto, la pusimos en pública para que así se pudiese acceder. Una alternativa pudo haber sido crear otro método que hiciese lo mismo, pero consideramos que no tenía sentido pudiendo heredarlo de su superclase evitando así duplicar código. El segundo problema fue que no sabíamos como se definía exactamente la clase en el sentido de que no llegábamos a comprender qué almacenaba ni cómo, luego entendimos que era otro HashSet de Stocks de los Books solo que a diferencia del catálogo, este estaba vacío en un inicio. Y aquí es donde surgió el tercer problema, cuando ya se nos ejecutaba correctamente la interfaz gráfica y

veíamos que se podía interactuar correctamente con ella, nos dimos cuenta de que el stock se compartía ya que por ignorancia copiamos la referencia en vez de duplicar el carrito, pero para solucionar esto de manera eficaz lo que hicimos fue crear un nuevo catálogo, inicializar las copias en 0 y luego añadir cada Stock a collection del *ShoppingCart* y de esta forma lo solucionamos independizando ambos Stocks.

En conclusión, esta práctica posiblemente, a pesar de que no era muy extensa, ha sido la que más nos ha costado debido a su dificultad y la gran libertad que nos daban. Si que es cierto que tener la libertad de implementación es algo muy positivo a nivel de desarrollo de nuestras habilidades ya que no se nos da todo tan pautado y tenemos que pensar nosotros mismos en la solución, pero el problema es que el momento en el que la teníamos que hacer por los globales, se nos complicó bastante la cosa. A pesar de ello, conseguimos sacarla a delante y con un resultado acorde a lo demandado, la aplicación funciona tal y como debería, se pueden añadir y quitar libros y tanto el stock como el precio varían en función de estos, si que es cierto que tal vez el código no está completamente optimizado pero con los recursos proporcionados, nuestros conocimientos y el tiempo disponible, creemos que el resultado es más que satisfactorio, de tener más tiempo tal vez pudiésemos haber encontrado alguna que otra forma de hacerlo, pero aún así, el objetivo de esta práctica se ha alcanzado.