Informe e1, Sistema de compra online

Introducción

El ejercicio consiste en crear el sistema de una tienda online para realizar compas de productos de un almacén. El código se organizó en tres partes, una con los productos y el almacén, otra con los pedidos y otra con los estados en los que puede estar el pedido.

- **Product y Store**: Son dos clases pequeñas que una tiene la utilidad de crear productos y la otra la de crear un almacén en el que almacenar los productos.
- **Order**: Es como la clase principal, en la que se guarda toda la información relacionada con los pedidos, identificador, productos en el carrito, si está pagado o no, si ya se pagó se almacena también la fecha...
  También tiene métodos para añadir, quitar y modificar la lista de productos del carrito.
- **StateOrder**: Es la interfaz que implementa a los cinco posibles estados del pedido, cada estado tiene sus propias definiciones de las interfaces que los implementas.

Planificación

Patrón State

Para iniciar el diseño pareció ideal utilizar el **patrón State**, que nos permite ir de un estado a otro y hacer cosas específicas para cada uno. Por lo que se creo una interfaz **StateOrder** que implementará a las clases de los estados y que la será usada por la clase **Order**. Luego, las posibles acciones de añadir, eliminar y modificar productos serán implementadas en en Order y se podrán hacer o no según en que estado esté.

También se aprovechó la interfaz creada para esta patrón, para hacer que sea extendida por otra interfaz llamada **ScreenInfo**, que mostrará la información del pedido en el estado en el que esté. Esta interfaz implementa a **Order** para que pueda ser usada por la clase de los pedidos.

Patrón Singleton

También se utilizó el **patrón Singleton**, ya que al principio para cambiar de clase siempre se creaba una clase nueva aunque se pasáramos por ese estado (Ejemplo: Ir de ShoppingCart a CheckOut, volver ShoppingCart y de nuevo a CheckOut, creaba cuatro instacias dos para cada estado). Al usar este patrón, **limitamos** la creación de instancias por clase a una, por lo que con el ejemplo anterior solo se crearían dos instancias.

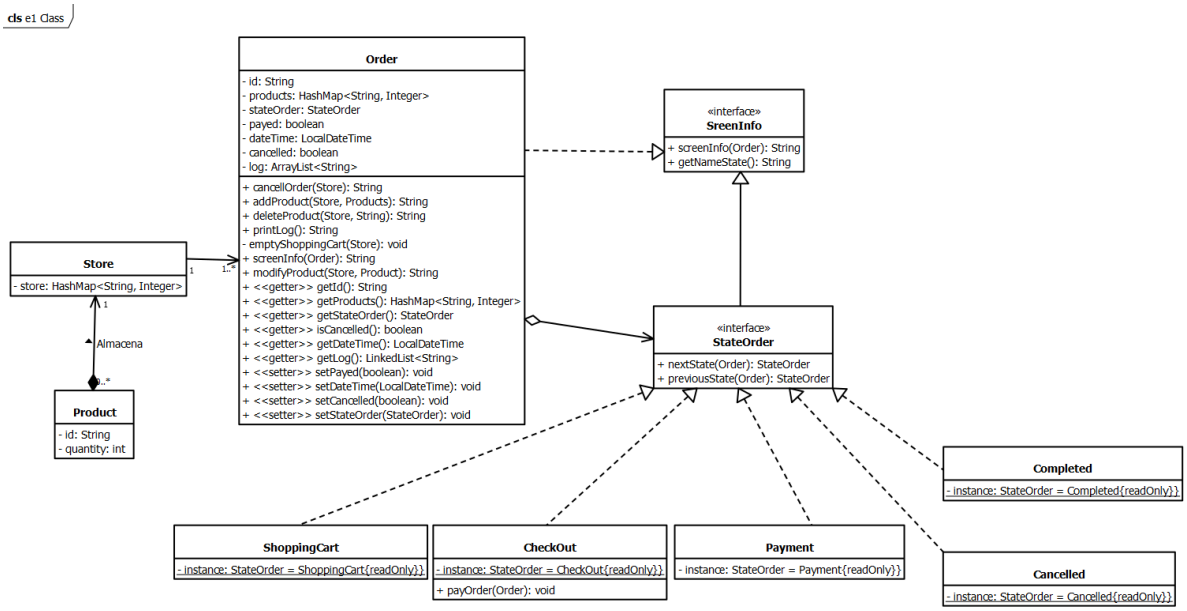
Principios SOLID

- **Principio abierto-cerrado**: Se utilizó al hacer del patrón State, donde en función de en que estado estemos, podremos hacer unas acciones u otras.
- **Principio de inversión de dependencia**: Se hace uso de este en prácticamente todas las clases, excepto Product y Store, el resto todas tienen implementaciones con sus respectivas sobrescrituras.
- **Principio de segregación de interfaces**: Al usar el patrón State de la forma en la que está implementada, hacemos uso de este principio, ya que la otra forma de implementar este patrón sería por medio de condicionales con un enum. Esto lo haría menos escalable, algo necesario ya que en el enunciado del ejercicio se indica que habrá más estados.

El resto de principios de SOLID no se utilizan realmente porque en este ejercicio no hubo ninguna relación de herencia.

Diagramas

De clases



De estados