Práctica

Descripción de la Práctica

El objetivo de esta práctica es que el alumno sea capaz de construir una aplicación simplificada para la gestión de ventas de una tienda on-line de componentes informáticos. A lo largo de esta memoria se llamará indistintamente “componente” o “producto” a cada uno de los componentes informáticos que se venden en la tienda.

La aplicación que desarrollaremos deberá ser capaz de:

- Gestionar un stock (o inventario de existencias) de componentes informáticos. El stock constituye el conjunto y cantidad de cada tipo de producto disponible para la venta en nuestra tienda.

- Manejar un conjunto de “carritos de la compra”, es decir, disponer de un mecanismo que nos permita recoger los pedidos de nuestros clientes antes de formalizar la compra mediante el pago.

- Guardar el conjunto de ventas realizadas por la tienda, es decir, almacenar los procesos de compra completados mediante pago, ya se haya realizado éste por medio de transferencia bancaria o mediante tarjeta.

Cada venta deberá contener información sobre los productos vendidos, la fecha en que se realizó la venta, el cliente y el modo de pago.

En nuestra tienda habrá diferentes tipos de componentes como memorias, placas base y periféricos.

El sistema, además de almacenar información sobre el stock, las ventas en proceso y las ventas completadas de la tienda, deberá permitir realizar una serie de acciones como son la modificación de la cantidad disponible de un determinado producto en el stock; la creación y borrado de productos, inserción, modificación y eliminación de productos dentro de los “carritos de la compra”, la consulta de ventas, la creación de nuevos componentes o tipos de componentes, etc.

La forma de desarrollar la práctica será partiendo del enfoque más básico, avanzando de forma iterativa. Para ello planteamos dos etapas. En la segunda etapa se partirá del diseño y desarrollo de la etapa anterior para añadir nuevas características y funcionalidades, poniendo en juego los conceptos del temario que se van estudiando progresivamente.

Etapa 1b

Se deberá añadir a la fecha del CarritoDeLaCompra: la hora, minutos y segundos en que se ha creado el carrito. Para ello se puede sustituir el tipo utilizado en la primera etapa para la fecha. La fecha tomará el valor de la hora del sistema en que se crea el carrito.

La clase Venta también debe contener hora, minutos y segundos en el atributo de fecha. Recordemos que la fecha-hora de la venta no tiene por qué coincidir con la del carrito.

Etapa 2

A continuación, se debe crear el sistema para gestionar las ventas y llevar un control del stock de productos disponibles en nuestra tienda. Para ello deberá crearse una clase ProductoEnStock con las siguientes características:

1. Deberá ser capaz de guardar información relativa al componente, al número de unidades disponible para su venta en un momento determinado y el número de unidades "reservadas" por otros carritos.

2. Se ha de permitir, aumentar y disminuir el número de unidades disponibles para la venta.

Por otro lado, deberá crearse una clase TiendaOnLine que incluya:

1. Una lista de clientes. La lista no podrá contener 2 clientes con el mismo código.
2. Una lista de fabricantes. La lista no podrá contener 2 clientes con el mismo código.
3. Un inventario formado por un conjunto de productos en stock. Lógicamente, en este stock no puede haber dos productos correspondientes con un mismo componente. Es decir, que cada vez que la tienda recibe un nuevo producto para su venta (cada vez que compra un producto a sus proveedores) creará una nueva instancia de ProductoEnStock con una referencia a ese producto y el número de unidades que haya adquirido para su venta. Si ese producto ya existe no se podrá añadir al inventario puesto que ya existe.
4. Una lista de los carritos de la compra creados. Un mismo cliente no puede disponer de 2 carritos de la compra.
5. El conjunto de ventas realizadas por la tienda. Hay que recordar que un carrito se convierte en una venta cuando se realiza el pago de los productos indicando la forma de pago. Debe poderse proporcionar por pantalla un listado de las ventas realizadas por un cliente, indicando para cada una de las ventas su fecha, el identificador de cliente, la lista de códigos internos de los productos y el precio total de la venta.

La clase TiendaOnLine debe realizar todas las acciones correspondientes a este tipo de negocio:

1. Debe permitir añadir clientes y fabricantes a las listas correspondientes. También se permitirá mostrar dichas listas.

2. Debe permitirse aumentar o disminuir la cantidad de cada uno de los componentes disponibles en el stock. Esta funcionalidad permitirá actualizar el stock cada vez que se realice una venta. Dado un código de componente y las unidades (positivas o negativas) se actualizará el stock de dicho producto.

3. Permitirá añadir un producto al inventario según lo dicho anteriormente.

4. Permitirá añadir un carrito a la lista de carritos para un cliente dado y con la fecha del sistema.

5. Permitirá añadir un producto al carro y eliminar un producto del carro.

6. Deberá permitir mostrar la lista de carritos que se tengan en un momento determinado (con la ayuda de la funcionalidad correspondiente de CarritodelaCompra). También se permitirá mostrar la lista de ventas.

7. Debe permitir eliminar un carrito (o una venta) dado un cliente (y una fecha en el caso de la venta), eliminándose tanto el carrito (o la venta) como su contenido. La eliminación del carrito podrá venir dada porque un usuario anule un proceso de compra o en el momento en el que confirme una compra (momento en el que se crea una nueva Venta). La eliminación de la compra podría venir dada por una devolución del total de una venta.

8. Permitirá comprar un carro determinado, dada la posición del carro dentro de la lista de carros y una forma de pago. Esta funcionalidad debe actualizar el stock.

9. Debe permitir además, eliminar un componente de una venta ya realizada. Dada la fecha de una venta, el cliente que la realizó y el código interno de un componente, se eliminará el componente y se actualizará el stock del inventario. Con esta funcionalidad cubriremos las devoluciones de productos ya vendidos. Cada vez que se realice esta acción habrá que actualizar el stock. Por simplicidad no se considerarán aspectos relacionados con el posible cambio en la facturación, ya que se trata de un aspecto no contemplado.

10. Incluirá un método que muestre por pantalla el stock de la tienda. Debe imprimirse la cantidad de cada uno de los componentes de los que se dispone en stock, indicando el código interno y la descripción del producto.

Se pide además añadir al método main(), que se creó en la Etapa 1, el código necesario para:

1. Crear una nueva tienda on-line.

2. Crear varios clientes y fabricantes.

3. Crear varios componentes y, a continuación, los productos en stock asociados, fijando diferente número de unidades disponibles.

4. Añadir los productos en stock a la tienda virtual.

5. Crear un carrito de la compra añadiendo una serie de productos.

6. Realizar la compra; es decir, se deberá crear una venta a partir de un carrito de la compra previamente creado (con todo su contenido) y un modo de pago. En este momento habrá que modificar el stock de la tienda actualizando las cantidades de cada producto disponible, es decir, que si en una venta se incluye un determinado producto habrá que reducir la cantidad disponible de ese producto en el stock.
7. Almacenar la venta en la tienda virtual.
8. Mostrar por pantalla el stock de la tienda.
9. Dado que nuestra tienda admite devoluciones, probaremos ahora dicha funcionalidad. Deberá borrarse algún producto dentro de una venta ya realizada, actualizando el stock como sea necesario. Pruebe también el borrado de una venta, simulando una devolución completa de una venta realizada previamente.
10. Mostrar de nuevo por pantalla el stock de la tienda.
11. Buscar por código interno un producto que no exista en la tienda, produciendo un mensaje informativo al respecto.
12. Siempre que se utilicen códigos de clientes o fabricantes se verificará su existencia.

En nuestro sistema hasta ahora sólo se ha modelado un único tipo de componentes. Ahora se desean incluir nuevos tipos de componentes, que llamaremos Memoria (y que será de tipo Flash, RAM o disco duro), Placas base y Periféricos (que podrán ser Teclados, Ratones e Impresoras).

Estos componentes tienen las mismas características de los componentes originales, con las siguientes novedades (no se han de modificar las clases que ya teníamos para incluir estos nuevos tipos de componentes en el sistema):

- Toda Memoria incluye un tamaño. Además, queremos ofertar diferentes tipos de memoria, para los que hay que describir la siguiente información:
  - En el caso de memoria RAM el tipo (DDR, DDR2 o DDR3).
  - En el caso de memoria Flash la velocidad de lectura.
  - En el caso de disco duro el tipo (externo o interno).

- Un Placa base incluye un tipo de procesador, un tamaño expresado en ancho y largo, y un tipo de ranura de memoria permitida.
- Un Periférico tiene un tipo de puerto de comunicación. Para cada tipo de periférico se almacenarán datos adicionales.
  - En el caso de teclados se incluye un idioma.

◦ En el caso de ratones incluye el número de botones.

◦ En el caso de impresoras se incluye el tipo de impresión: por tinta o láser.

Haciendo uso del mecanismo del polimorfismo, cada tipo de componente debe mostrar por pantalla sus características particulares, además de la información básica antes considerada.

Se pide además añadir al método main() las sentencias necesarias para :

1. Añadir los nuevos componentes creados al stock de la tienda.

2. Añadir algunos componentes de los nuevos tipos a diferentes carritos de la compra. A continuación, se deberá proceder a realizar las ventas especificando diferentes métodos de pago.

3. El código necesario para mostrar en cada uno de los diferentes tipos de componentes su información básica, de modo que se compruebe que efectivamente aparecen todos los nuevos datos de los nuevos productos y no sólo la información básica implementada en la clase padre Componente.

Opcional:

1. Modifique las clases que sean necesarias para incluir la posibilidad de listar los componentes según su tipo. Es suficiente con que exista un método para listar cada tipo de componente. Añadir al método main() el código necesario para que, dada una venta, se listen por separado las Memorias, Placas base y Periféricos contenidos en ella utilizando los métodos que se han creado a tal efecto. Para identificar la clase a la que pertenece un objeto se puede hacer uso del operador instaceof, que dado un objeto nos permite comprobar si implementa una clase concreta, por ejemplo:

   ```
   if (n instanceof Periferico) {
   ...
   }
   ```

2. El objetivo de este apartado es que las ventas de la tienda se muestren ordenadas alfabéticamente por cliente. Para mostrar las ventas de forma ordenada por pantalla tenemos dos opciones: almacenarlas con cierto orden u ordenarlas en el momento de presentarlas por pantalla. En este último caso, cada vez que queramos visualizar las ventas por pantalla habría que reordenarlas. Como sólo queremos ordenarlas por cliente, no necesitamos reordenarlas alfabéticamente cada vez, por lo que elegiremos la primera alternativa que nos permite reordenar únicamente cuando una nueva venta sea realizada. Por tanto, ahora se ha de modificar el código de inserción de ventas en nuestra tienda de forma que se inserten ordenadas alfabéticamente por cliente. No es necesario usar estructuras de datos que proporcionen ordenación; se pide utilizar la misma estructura de lista que ya teníamos, pero modificando la manera en que se guarda una venta. Ahora en lugar de simplemente añadirla, tendremos que buscar previamente la posición en la lista donde corresponde insertarla según el cliente e insertarla en dicha posición.

Se pide también incluir en el método main() el código necesario para comprobar el correcto funcionamiento de las modificaciones realizadas, de forma que se presenten por pantalla las ventas ordenadas alfabéticamente por cliente.

Observar si esta nueva inserción supone algún cambio en otros métodos ya implementados.