# Programación Orientada a Objetos – Curso 2023-24

# Práctica 7 – La Clase Market

Últimos ejercicios para nuestro marketplace.
En moodle encontrarás el ficheros los tests (`market-test.cc`) que tendrás que descargar al directorio marketplace/tests.
Crea el directorio poo/p7 con una copia del directorio poo/p6.
Adapta y añade los ficheros CMakeLists.txt que sea necesario.

**EJERCICIO 1.** La clase Market
La clase *Market* representa la tienda completa, es decir, simulará el funcionamiento de toda la tienda online con clientes, vendedores y sus transacciones, por tanto, tendrá una lista de clientes (client_list_) y una lista de vendedores (seller_list_). Un *Market* además tiene una cadena ('slogan_') con el eslogan de la tienda.
Esta clase hará el papel de 'Application Factory', un **patrón de diseño** que veremos en clase de teoría en el tema 'Patrones de Diseño'.

Codifica los siguientes métodos para la clase Market:

1. Constructor de la clase Market que recibe como parámetro el eslogan de la tienda.
2. GetSlogan(): observador que devuelve el slogan del Market.
3. SetSlogan(): recibe un std::string con el nuevo slogan del Market.
4. AddClient() de tipo bool que recibe como parámetro un cliente (objeto de la clase *Client*). Si ya existe en el Market un cliente con ese id, no lo añade y devuelve false. Si no existe un cliente con ese id, lo añade y devuelve true.
5. AddSeller() de tipo bool que recibe como parámetro un vendedor (objeto de la clase *Seller*). Si ya existe en el Market un vendedor con ese id, no lo añade y devuelve false. Si no existe un vendedor con ese id, lo añade y devuelve true.
6. NClients() y NSellers() que devuelven el número de clientes y vendedores respectivamente en Market.
7. DeleteClient() de tipo bool que recibe como parámetro un cliente. Si existe un cliente con ese id, lo borra y devuelve true. Si no existe, devuelve false. No tener en cuenta la devolución de los productos a la cesta de los vendedores.
8. DeleteSeller() de tipo bool que recibe como parámetro un vendedor. Si existe un vendedor con ese id, lo borra y devuelve true. Si no existe, devuelve false. No tener en cuenta si hay clientes con sus productos.
9. AddProductSeller(). Recibe un producto 'p' y el id de un vendedor (id_seller) y añade el producto 'p' al stock del vendedor id_seller. Si no existe el vendedor devuelve false. En caso contrario devuelve true.
10. AddProductClient(). Recibe un producto 'p' y el id de un cliente (id_client). Añade el producto 'p' a la cesta de la compra del cliente id_client. Si no existe a la venta un producto con ese mismo id por parte de ninguno de los vendedores, o no existe el cliente, devuelve false. En caso contrario devuelve true.
11. DeleteProductSeller(): Recibe un producto y el id del vendedor. Borra ese producto de la cesta del vendedor y devuelve true. Devuelve false en caso de algún error. No tiene en cuenta si el producto está en la cesta de algún cliente.
12. DeleteProductClient(): Recibe un producto y el id del cliente. Borra ese producto de la cesta del cliente y devuelve true. Devuelve false en caso de algún error. No tener en cuenta la devolución del producto a la cesta del vendedor.
13. GetMoneyInBasket(): devuelve la suma de euros que hay en todas las cestas en ese momento.

14. DumpMarket(): recibe un dato de tipo int (outmode) que si es igual a 1 vuelca a un fichero texto de salida (ventas.txt) la siguiente salida por cada cliente y devuelve true:

```
|-------------------------------------|
|CLIENT ID  |PRODUCT ID |PRODUCT QTY  |
|C1         |P1         |2            |
|C1         |P2         |1            |
|...        |...        |...          |
|C2         |P3         |1            |
|C2         |P4         |1            |
|...        |...        |...          |
|...        |...        |...          |
|-------------------------------------|
|TOTAL: 110€                          |
|-------------------------------------|
```

    Si outmode es igual a 0, saca la tabla por pantalla y devuelve true. Y si outmode es <0 o >1, la función debe devolver false.

    Para este fichero de salida utiliza exclusivamente las nuevas utilidadas que trae C++ para ficheros (#include <fstream>).

    Debajo de “CLIENT ID” debe aparecer el ID de cada cliente del marketplace y repetirse para cada uno de los productos en su cesta.

    Detrás de ‘TOTAL’ debe aparecer el total de todas las cestas que es el valor que devuelve la función GetMoneyInBasket().

    Debes conseguir que la salida coincida con el formato de la tabla mostrada.

Crear esta clase en el directorio src/market y hacer un pequeño programa principal market-main.cc que declara un objeto de tipo Market y muestra en pantalla el eslogan.

**EJERCICIO 2.** Codifica una aplicación en el fichero src/app/app-main.cc con un programa principal con una función main() que cree un marketplace con un eslogan, inserte 5 clientes, y 5 vendedores cada uno con 5 productos. A continuación mostrará un menú con las siguientes opciones:

1. Añadir cliente.
2. Añadir vendedor.
3. Añadir producto en la cesta de un cliente pidiendo su id de cliente y el id del producto.
4. Borrar producto de la cesta de un cliente pidiendo su id de cliente y el id del producto.
5. Volcar datos al fichero de salida ventas.txt.
6. Volcar datos a pantalla.
7. Salir del programa.