Entrega 4.3: Tienda

1) Desarrolla un sistema simple para gestionar una tienda, que incluirá las siguientes tres clases principales:

1. **Clase Producto**
   Representa un producto que se vende en la tienda.
   Atributos:
   - **nombre**: El nombre del producto.
   - **precio**: El precio del producto.
   - **stock**: Cantidad disponible en la tienda

   Métodos:

   - **mostrar_producto():** Muestra el nombre, precio y el stock del producto.
   - **devolver_precio()**: Devuelve el precio del producto
   - **actualizar_stock()**: Recibe una cantidad de stock (puede ser positiva o negativa) y lo añade al stock actual.
   - **hay_disponible():** Recibe una cantidad y verifica si dicha cantidad es suficiente con el stock actual.

2. **Clase Cliente**
   Representa a un cliente de la tienda.
   Atributos:
   - **nombre:** El nombre del cliente.
   - **carrito:** Una lista de productos y cantidades que el cliente desea comprar (formato: lista de tuplas (Producto, cantidad)). No estará en el constructor, sino que por defecto será una lista vacía.
   - **ticketsCompra()**: un diccionario de los tickets de los productos que el cliente ha comprado. No estará en el constructor, sino que por defecto será un diccionario vacío. El diccionario tendrá la estructura siguiente: {“Ticket 1” : [(producto, cantidad), (producto,cantidad)]}
   - **membresiaVIP:** Será True o False. Por defecto será False, por lo que no estará en el constructor.

   Métodos:

   - **agregar_producto()**: Recibe un producto y una cantidad de dicho producto para añadirlo al carrito, verificando si hay suficiente stock.
   - **mostrar_carrito():** Muestra todos los productos y cantidades en el carrito del cliente. (Mostrará los nombres de los productos)
   - **calcular_total_carrito():** Devuelve el costo total de los productos en el carrito.
   - **vaciar_carrito():** vacía el carrito actual del cliente.

- **comprar():** el cliente comprará lo que hay en el carrito. Añadirá un nuevo ticket a ticketsCompra. “Ticket (número del ticket)” será la clave, y el valor será el carrito actual, con la diferencia de que hay que aplicarle a cada producto un descuento del 10% en caso de que el cliente sea VIP. Posteriormente, tendremos que actualizar el stock de cada producto comprado, y por último, vaciar el carrito.
- **darse_alta_baja_vip():** el cliente se da de alta como VIP en caso de que no lo sea o se da de baja en caso de que si lo sea.
- **total_gastado():** calculará cuánto dinero ha gastado el cliente en base a los tickets
- **mostrar_cliente**(): mostrará información acerca del cliente. Se mostrará su nombre, si es VIP y el dinero total gastado en la tienda. (Lo que hay en el carrito no lo ha gastado todavía).

3. **<u>Clase Tienda</u>**
   Representa la tienda en general.
   Atributos:
   - nombre: El nombre de la tienda.
   - clientes: Una lista de los clientes registrados en la tienda.
   - productos: Una lista de los productos registrados en la tienda.

   Métodos:

   - **agregar_cliente():** Recibe un cliente y lo añade a la lista de clientes de la tienda.
   - **agregar_producto**(): recibe un producto y lo añade a la lista de productos de la tienda
   - **mostrar_clientes():** Muestra los nombres de todos los clientes de la tienda.
   - **buscar_cliente()**: Recibirá el nombre de un cliente y lo buscará en la lista de clientes y lo devolverá en caso de existir. En caso de no existir, devolverá False. (En caso de existir devolverá el cliente como tal).
   - **buscar_producto():** Recibirá el nombre de un producto y lo buscará en la lista de productos, y lo devolverá en caso de existir.
   - **cliente_top():** Mostrará el cliente que ha gastado más dinero en la tienda.
   - **procesar_compra():** Recibe el nombre de un cliente, la tienda lo buscará y la tienda hará la acción de comprar por el cliente.
   - **mostrar_productos():** Muestra una lista de tuplas con todos los productos de la tienda con el formato [(nombre, precio, stock)]

Además, el ejercicio contendrá las siguientes funciones (en ninguna clase):

- **MostrarMenu()**: función que mostrará el menú de opciones al usuario para que elija lo que quiere hacer (Solo lo mostrará una vez, y solo hará eso): Las opciones del menú son:
  1. Registrar Producto

2. Registrar Cliente
3. Agregar Producto al carrito de un cliente
4. Mostrar el carrito de un cliente
5. Mostrar información de la tienda
6. Procesar compra de un cliente
7. Modificar información de un cliente
8. Salir

- **Registar_Producto()** Recibe una tienda. Permite añadir un nuevo producto a la tienda preguntando al usuario los datos de nombre, precio y stock inicial. Comprobará si dicho producto existe ya. (Ayudarse de la función buscar_producto).
- **Registrar_Cliente()**: Recibe una tienda y pedirá al usuario que introduzca un nombre de cliente. Comprobará si dicho cliente existe ya. Una vez creado, dará la opción de darse de alta como vip.
- **Agregar_Producto_Carrito_Cliente()**: Recibirá una tienda y pedirá al usuario que introduzca el nombre de un producto y el nombre de un cliente. En caso de existir ambos (y en caso de haber stock), lo añadirá al carrito del cliente.
- **Mostrar_Carrito_Cliente()**: Recibe una tienda y pedirá al usuario que introduzca el nombre de un cliente. En caso de existir, mostrará el contenido actual del carrito de un cliente.
- **Mostrar_informacion_tienda():** Recibe una tienda y mostrará los productos, los clientes y el cliente top. (Utilizará las funciones mostrar_productos, mostrar_clientes y cliente_top).
- **Procesar_Compra_Cliente()**: Recibe una tienda y pedirá al usuario que introduzca el nombre de un cliente. En caso de existir, realizará la acción de comprar el carrito actual del cliente.
- **Modificar_informacion_cliente():** Recibe una tienda y pedirá al usuario que introduzca el nombre de un cliente. En caso de existir, preguntará al usuario si desea cambiar el nombre del cliente o si desea cambiar la membresía VIP del cliente.
- **Menu():** función que generará un bucle infinito, el cuál mostrará las opciones de Menú y permitirá al usuario introducir una de las opciones disponibles. Esté método llamará al resto de métodos para realizar dichas funciones. El método recibirá una Tienda como argumento de entrada.