EXAMEN DE RECUPERACIÓN

Para la realización de este examen, se va a partir del módulo “almacen.py”. Y se deberán crear dos módulos independientes denominados “produccion” y “tienda”.

Una vez sabido esto, las clases se van a distribuir de la siguiente manera:

- Módulo producción: Clase Productos.
- Módulo tienda: Clase Venta, y Clase Cliente.

1) La calse **Productos**: **[5 Puntos]**
   En un módulo independiente llamado “**produccion**” se ha de implementar una clase llamada “**Productos**”, que debe cumplir los siguientes requisitos.
   a) Crear un constructor que tenga como parámetros de entrada, el “**nombre**” y “**articulos**”.
      - **nombre**: nombre que tendrá el producto final, p.e. Mesa.
      - **articulos**: conjunto de artículos y sus cantidades de los que está compuesto el producto.

      Ejemplo para instanciar la clase:

      ```
      Productos("Mesa", {"Tablero": 1, "Pata": 4, "Tornillos": 16})
      ```

      Al instanciar la clase, se almacena el objeto dentro de la clase. Teniendo en cuenta que no pueden existir 2 productos con el mismo nombre.
      En el caso de coincidir, debe mostrar un error,como:

      ```
      ValueError: Este producto ya ha sido dado de alta.
      ```

      Internamente, el constructor tendrá que tener un atributo, “cantidad” que podrá saber el stock del que se dispone y que deberá inicializarse a 0 cuando se instancie.
   b) Todos los setter y getter necesarios.
   c) Debe contener de los siguientes método estático:
      - Método estático “**montar**”:
        Este método debe añadir al stock del producto la cantidad indicada en los argumentos de entrada. Y en el caso de querer montar un producto no dado de alta, lanzará el siguiente error.

        ```
        ValueError: Produccto: Martillo no dado de alta.
        ```

        Argumentos de entrada:
        - **producto**: nombre del producto.
        - **cantidad**: número de unidades a añadir al stock de este producto.

        Ojo: en el caso de no tener stock suficiente, de los artículos de los que está constituido cada producto, debe arrojar un error e incrementar al stock de este componente sólo la cantidad que sea posible.
      - Método estático “**imprimir_producto**”: muestra por pantalla los productos dados de alta con el stock correspondiente a cada uno.
        No tiene argumentos de entrada.
        Ejemplo:

```
In [11]: Productos.imprimir_productos()
**************************************************
Producto número: 1

-> Produccto: Mesa
-> Cantidad en stock: 5
Cada unidad está constituida por:
  * Tablero -> 1 [uds]
  * Pata -> 4 [uds]
  * Tornillos -> 16 [uds]

**************************************************
Producto número: 2

-> Produccto: Martillo
-> Cantidad en stock: 0
Cada unidad está constituida por:
  * Mango -> 1 [uds]
  * Cabeza -> 1 [uds]

**************************************************
```

- Método estático "**retirar_producto**": se ha de comprobar, tanto si hay stock suficiente para retirar dicha cantidad como si existe dicho producto y en caso contrario enviar un error.

```
ValueError: Producto no dado de alta.
```

Si existe el producto y hay stock, se descontará del stock.
Argumentos de entrada:

- **producto**: Nombre del producto.
- **cantidad**: Número de unidades a retirar.

d) Crea los métodos necesarios para su representación normal y legible por el usuario.

* IMPORTANTE: se ha de tener en cuenta que al crear una instancia de esta clase se han de desconectar del stock de artículos la cantidad correspondiente según la composición del producto en cuestión.

2) La clase **Ciente: [2 Puntos]**
Esta clase se encarga sólo de crear el tipo de dato cliente para poder almacenarlo dentro de otras clases. Debe considerarse lo siguiente:

a) Tener un constructor en el que se inicializan los atributos "**nombre**", "**nif**", "**direccion**", "**telefono**".

b) Debe crearse un método mágico para su representación normal y otro para que sea legible por el usuario.
Ejemplo de representación legible:

```
In [6]: print(Cliente("JOSE", "75242152F", "Calle: Luna", 950242526))

- Denominación: JOSE
- NIF: 75242152F
- Dirección: Calle: Luna
- Contacto: 950242526

In [7]:
```

c) Todos los setter y getter necesarios.

3) La clase **Venta: [3 Puntos]**

Dentro del mismo módulo donde se haya definido la clase Cliente, se debe definir una clase llamada "**Ventas**". Esta clase debe cumplir:

a) Que al instanciar esta clase debe almacenar dentro de la misma sus propias instancias. Además, la clase almacenará el número de entradas, que será un número correlativo según el orden cronológico.
   - Se recomienda que el "**id**" se almacene también dentro del estado interno para facilitar su consulta.

b) Debe crearse un constructor que tenga los argumentos de entrada "**producto**", "**cantidad**" y "**cliente**".
   - "**producto**": nombre del producto.
   - "**cantidad**": número de productos.
   - "**cliente**": instancia de la clase cliente.

   Al crearse una instancia de este objeto, se ha de retirar del stock de la clase "Producto" el material vendido.

c) Todos los setter y getter necesarios.

d) Crear el método necesario para una representación legible para el usuario.
   Ejemplo de representación legible:

```
-> ID de venta: 1; Nombre del producto: Mesa; Cantidad: 2 [uds]

Datos del cliente:
- Denominación: Jose
- NIF: 75242653F
- Dirección: Calle: Luna
- Contacto: 950242325
```

   Crear un método estático que muestre por pantalla todas las ventas hasta el momento.

```
In [8]: Venta.imprimir_ventas()
--------------------------------------------------------------------------------
-> ID de venta: 1; Nombre del producto: Mesa; Cantidad: 2 [uds]

Datos del cliente:
- Denominación: Jose
- NIF: 75242653F
- Dirección: Calle: Luna
- Contacto: 950242325
--------------------------------------------------------------------------------
-> ID de venta: 2; Nombre del producto: Mesa; Cantidad: 1 [uds]

Datos del cliente:
- Denominación: Jose
- NIF: 75242653F
- Dirección: Calle: Luna
- Contacto: 950242325
--------------------------------------------------------------------------------
```