	EXAMEN PRÁCTICO	PROGRAMACIÓN – 1.º DAW 24-mar-2023
---	------------------------	--

Nombre y apellidos:

Una empresa nos ha contratado para gestionar la entrada de material. Esta empresa quiere actualizar el stock de cada artículo al dar la entrada de su albarán correspondiente. Es decir, que cuando se dé de alta un albarán en el sistema, se ha de incrementar la cantidad que haya de este artículo, sumándole la cantidad indicada en el albarán.

Para hacer esto nos han impuesto desde el departamento de desarrollo de la empresa, los siguientes requisitos:

- Realizar una programación orientada a objetos (POO).
- Crear todas las clases dentro de un módulo llamado `almacen`.
- Dentro de este módulo, debes implementar 3 clases, **Articulo**, **Gestion_articulo** y **Albaran**.

1) La clase **Articulo**: [2 Puntos]

Esta clase tiene como objetivo crear instancias del tipo Artículo. Esta clase se gestionará desde la clase **Gestion_articulo**.

- a. Debe tener en su estado interno un `nombre` y una `cantidad`, ambas privadas.
- b. Hay que dejar implementados todos los `setter` y `getter`.
- c. Las instancias de esta clase deben de tener una representación normal y otra legible para el usuario, que debe mostrarse esta última de la siguiente forma:

“<<Nombre del artículo:>> <<Cantidad del artículo>> [uds]”

2) La clase **Gestion_articulo**: [4 Puntos]

Esta clase tiene como objetivo, gestionar las posibles operaciones que se les podrán realizar a las instancias de la clase **Articulo**.

- a. Esta clase no tiene estado interno, todos los métodos definidos serán estáticos dentro de esta clase.
- b. Los artículos que se den de alta se deben almacenar dentro de esta clase. Por tanto, es dentro de esta clase donde hay que actualizar la cantidad de cada artículo.
- c. Debe tener como métodos estáticos, los siguientes (como mínimo):

i. Método `alta_articulo`.

- Argumentos de entrada, una cadena de texto para identificar al `artículo`.
- Al ejecutar este método, se debe obtener un error si el artículo a añadir ya existía previamente.
- Ejemplo:

```
In [2]: from almacen import *
In [3]: Gestion_articulos.alta_articulo("Tornillos")
In [4]: Gestion_articulos.alta_articulo("Juntas")
In [5]: Gestion_articulos.alta_articulo("Juntas")
```



➔ Tras la ejecución de la última línea da un error.

```
ValueError: Artículo ya dado de alta.
```

ii. Método `agregar_cantidad_articulo`.

- Argumentos de entrada, una cadena de texto para identificar el `artículo` y un dato numérico para la `cantidad`.
- En el caso de intentar agregar una cantidad a un artículo no dado de alta previamente debe saltar un error.

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

 	EXAMEN PRÁCTICO	PROGRAMACIÓN – 1.º DAW
		24-mar-2023

- iii. Método `retirar_cantidad_articulo`. [Este método no es obligatorio].
- Igual que el método anterior pero substrayendo de la `cantidad`.
 - En el caso de intentar substraer una cantidad de un artículo no existente debe mostrar un error.

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

- Se debe tener en cuenta el stock del material antes de substraer.
- iv. Método `imprimir_articulos`.
- Sin argumentos de entrada.
 - Imprimir todas los `artículos` dados de alta junto con la `cantidad` de cada uno.
 - Al final del listado de los artículos se debe poner el total de artículos dados de alta.

```
In [12]: Gestion_articulos.imprimir_articulos()
-----
Tornillos: 100 [uds]
Juntas: 120 [uds]

Total de artículos dados de alta: 2 [uds]
-----
```

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

Esta clase tiene como objetivo, crear entradas de albaranes siempre y cuando sea posible, es decir que si se da entrada de un albarán y el artículo que se le pasa no está dado de alta, debe indicarlo y no hacer la entrada del albarán. Para hacer una entrada de albarán debe utilizarse la clase `Gestion_articulos`.

- Esta clase tiene como estado interno, un `num_entrada`, un `id_albaran`, un `articulo` y la `cantidad`.
- El atributo `num_entrada` debe generarse automáticamente desde la clase y de forma correlativa, con la intención de que cuando demos la primera entrada de un albarán, se le asigne el número 1 y a la segunda el 2, y así sucesivamente.
- Los atributos `id_albarán` y `articulo`, ambas son cadenas de texto.
- Crear todos los `setter` y `getter`.
- El atributo `cantidad` debe ser numérico.
- Al instanciar la clase **Albaran**, se debe crear un nuevo objeto que se debe almacenar dentro de la misma clase y se podrá consultar posteriormente, por ejemplo desde el siguiente método.
- Debe contener los siguientes métodos:

- Método `listar_albaranes`.

- Sin argumentos de entrada.
- Muestra en pantalla todas las entradas de los albaranes.

```
In [21]: Albaran.listar_albaranes()
-----
1 -> Albarán Nº: id1; Artículo: Tornillos, 134 [uds]
2 -> Albarán Nº: id1; Artículo: Tornillos, 1000 [uds]
3 -> Albarán Nº: id1; Artículo: Juntas, 100 [uds]

Se han realizado '3' de entradas
-----
```

- Métodos mágicos de la representación normal y una representación legible para el usuario.

Ejemplo propuesto de representación para el usuario.

```
In [4]: print(Albaran("123456DA", "Juntas", 105))
1 -> Id de Albarán: 123456DA; Artículo: Juntas, 105 [uds]
```