



# **UNIDAD I2: COLECCIONES**

**CLASE 5: EJERCICIOS**

## Ejercicio 1: **pedidos**

Una empresa nos pide gestionar el listado de pedidos que se realizan. Para ello crearemos las siguientes clases:

**LíneaPedido** que tendrá:

- Los atributos **producto** que será un string, **cantidad**, **precio** (double ambos) así como un entero **identificador**.

Como métodos tendrá:

- **Constructor** con parámetros, **getters** y **setters**
- **obtenerPVPLínea()** que devolverá el total de la linea multiplicando cantidad por precio.
- Tendrá el método **toString()** sobrescrito con el formato:

“Línea **identificador**  
**/t producto cantidad precio = PVPLínea**”.

## Ejercicio 1:

Por otro lado, la clase **Pedido** tendrá:

- Los atributos **dirección de envío**, **descripción** del pedido e **identificador** del pedido.

Tendrá **constructor** con parámetros, **getters**, **setters** y método **toString()**

Después crea un main en el que almacenaremos un mapa cuya clave será un pedido y el valor será un conjunto de LineaPedido. Crea varios pedidos y almacenalos en el mapa. Después crea un menú que permita listar los pedidos disponibles y, al seleccionar uno, mostrará las líneas de pedido asociadas a ese pedido ordenadas por identificador.

## Ejercicio 2: **frecuencia**

Crea un programa que almacene una lista de números enteros ingresados por el usuario (ejemplo: **10 5 3 10 2 3 5 5 7 8 10 3**) y utilizando diferentes tipos de colecciones, realices:

- **Detectar números duplicados:** Identificar los números que aparecen más de una vez en la lista y mostrarlos tal que **[10, 5, 3]**
- **Contar la frecuencia de cada número:** Mostrar cuántas veces aparece cada número en la lista.

10 → 3 veces / 5 → 3 veces / 3 → 3 veces / 2 → 1 vez / 7 → 1 vez / 8 → 1 vez

- **Ordenar los números por frecuencia:** Mostrar la lista de números duplicados ordenados de mayor a menor frecuencia y de menor a mayor. **[3, 5, 10, 2, 7, 8]**
- **Eliminar duplicados y mostrar la lista sin repetidos:** Crear una nueva lista con los números únicos, manteniendo su primer orden de aparición. **[10, 5, 3, 2, 7, 8]**

## Ejercicio 3: **historial\_de\_navegacion**

Implementa un programa que simule el historial de navegación de un navegador web mediante un conjunto de Strings elegido por ti, ten en cuenta que debe contar con las siguientes funcionalidades que serán accesibles por un menú:

- Visitar una nueva página: El usuario escribirá por pantalla la URL a la que quiere acceder. Debes comprobar que comience por www. y finalice en .com o .es. Después, agrega la URL al historial y actualiza la posición actual del usuario en el array.
- Retroceder (Botón "Atrás"): Permite volver a la página anterior si es posible.
- Avanzar (Botón "Adelante"): Si el usuario retrocedió en el historial, permite avanzar nuevamente.

### Ejercicio 3:

- Mostrar el historial completo: Lista todas las páginas visitadas en orden cronológico.
- Mostrar la página actual: Indica el nombre de la página en la que se encuentra.
- Eliminar una URL específica del historial: Permite eliminar una página del historial sin afectar la posición actual.
- Buscar si una URL ha sido visitada: Verifica si una URL está en el historial.
- Limpiar el historial: Borra todas las páginas registradas.

## Ejercicio 4: **censo\_de\_poblacion**

Crea un programa que gestione un censo de población. Se deberá crear la clase **Ciudadano** que tendrá:

- Los atributos **nombre, edad y ciudad**.

Así como los métodos:

- **Constructor** con parámetros, **getters y setters** y método **toString** que muestre la información con el formato: “**nombre** de edad **edad** vive en **ciudad**”

En main, deberás crear un mapa que permita identificar a los objetos de tipo ciudadano por su DNI. Además, mediante un menú deberás permitir:

- Registrar ciudadanos ingresando su DNI y la información a almacenar dentro de ciudadano. Se deben evitar registros duplicados (un ciudadano con el mismo DNI no puede registrarse dos veces).
-

#### Ejercicio 4:

- Buscar ciudadanos en el censo mediante su DNI.
- Eliminar ciudadanos del censo.
- Actualizar información de un ciudadano.
- Contar el número total de ciudadanos registrados.
- Además, incluir una opción que permita mostrar un informe con:
  - Cantidad de ciudadanos por ciudad.
  - Promedio de edad de los ciudadanos registrados.
  - Persona más joven y más mayor del censo.