RELACIÓN DE EJERCICIOS 12 - Colecciones: ArrayList, HashSet, HashMap

Resolver los siguientes problemas escribiendo el algoritmo con lenguaje Java.

Crear en GitHub un repositorio llamado **UD6-Colecciones** para subir cada uno de los ejercicios de la relación.

1. (ArrayList) Crea una clase **Menu**, que dispondrá de un atributo arrayList de String. Tendrá un método **creaOpcion** que permitirá crear las distintas opciones del menú, otro método **muestraMenu** que mostrará todas las opciones del menú, y un método **capturaOpcion** que devolverá la opción seleccionada por el usuario. Prueba dicha clase en otra clase denominada **TestMenu**. La clase **Menu** la utilizaremos en el resto de ejercicios que deban usan un menú de opciones

2. (ArrayList). Crea un nuevo proyecto denominado **AlmacenArrayList**, donde se crearán las siguientes clases:
    - Crea la clase **Articulo**, con la siguiente información: código, descripción, precio de compra, precio de venta, stock. Los atributos serán privados y se crearán los métodos get, set necesarios. El constructor recibirá el código del artículo y la descripción.
    - Crea una clase **AlmArticulo**, dicha clase tendrá un atributo que será un arrayList de artículos. La clase dispondrá del método **almacena** que recibirá un artículo y lo insertará en el arrayList. También dispondrá de un atributo para almacenar el número total de artículos que existan en el almacén (**totalArticulos**). Sobrecarga el método **toString** de la clase **Articulo** para que realice la impresión de todos los artículos del arrayList.
    - Crea la clase **GestSimAlm** para llevar el control de los artículos de un almacén. De cada artículo se debe saber el código, la descripción, el precio de compra, el precio de venta y el stock (número de unidades). Utilizaremos la clase **Articulo** del apartado anterior y almacenaremos todos los artículos de nuestro almacén en un arrayList. La clase dispondrá de los siguientes métodos:
        - **entradaAutomatizada**: creará 4 artículos de prueba del programa.
        - **lista**: Mostrará todos los datos de los artículos del almacén.
        - **alta**: Creará un nuevo artículo en el almacén.
        - **baja**: Dará de baja un artículo en el almacén.
        - **entradaMercancia**: Aumentará el stock de un determinado artículo
        - **salidaMercancia** Disminuirá el stock de un determinado artículo. Hay que controlar que no se pueda sacar más mercancía de la que hay en el almacén.
    - Realiza la prueba de la clase anterior, para ello realiza un programa (clase **TestGestSimAlm**) que muestre el siguiente menú para que el usuario pueda seleccionar la opción elegida.
        0. Entrada automatizada (creará 4 artículos de prueba del programa)
        1. Listado (Mostrará todos los datos de los artículos del almacén)
        2. Alta (Creará un nuevo artículo en el almacén)
        3. Baja (Dará de baja un artículo en el almacén)
        4. Entrada de mercancía (Aumentará el stock de un determinado artículo)
        5. Salida de mercancía (Disminuirá el stock de un determinado artículo)
        6. Salir

RELACIÓN DE EJERCICIOS 12 - Colecciones: ArrayList, HashSet, HashMap

3. (ArrayList). En el ejercicio 8 de la relación 11 construimos una biblioteca con libros y revistas. Ahora queremos que los objetos de las clases **Libro** y **Revista** puedan ser almacenados y ordenados en un arrayList por su título. Para ello debemos implementar la interfaz **Comparable** en dichas clases. Modifica el ejercicio 8 para que se cumplan estos nuevos requerimientos. Crea una clase **TestPublicacion**, donde realizaremos la prueba de este nuevo ejercicio. Debemos almacenar al menos tres publicaciones y ordenarlas dentro de un arrayList.

4. (HashSet). Realiza un pequeño programa que pregunte al usuario 5 números diferentes (almacenándolos en un HashSet), y que **DESPUÉS** calcule la suma de los mismos (usando un bucle for-each).

5. (HashSet y Enum). Crea una aplicación en Java que controle los pedidos de un restaurante. Cada pedido tiene un tipo de comida que pertenece a una enumeración (**enum**) y un conjunto de ingredientes opcionales que se almacenan en un **HashSet**.

    - Define un enum llamado **TipoComida** con valores como PIZZA, HAMBURGUESA, ENSALADA y PASTA.
    - Crea una clase Pedido que tenga:
        - Un atributo numérico para el número de pedido
        - Un atributo de tipo **TipoComida** para almacenar el tipo de comida.
        - Un HashSet<String> para guardar ingredientes adicionales.
        - Un constructor y métodos para agregar ingredientes y mostrar el pedido.
    - En el main, crea varios pedidos, añade ingredientes y muestra la información.

```java
public class Restaurante {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Crear un pedido de tipo PIZZA
        Pedido pedido1 = new Pedido(TipoComida.PIZZA);
        pedido1.agregarIngrediente(ingrediente:"Pepperoni");
        pedido1.agregarIngrediente(ingrediente:"Champiñones");
        pedido1.mostrarPedido();

        // Crear un pedido de tipo HAMBURGUESA
        Pedido pedido2 = new Pedido(TipoComida.HAMBURGUESA);
        pedido2.agregarIngrediente(ingrediente:"Queso");
        pedido2.mostrarPedido();

        // Crear un pedido de tipo ENSALADA
        Pedido pedido3 = new Pedido(TipoComida.ENSALADA);
        pedido3.mostrarPedido();

        // Crear un pedido de tipo PASTA
        Pedido pedido4 = new Pedido(TipoComida.PASTA);
        pedido4.agregarIngrediente(ingrediente:"Albahaca");
        pedido4.mostrarPedido();

        // Mostrar el número total de pedidos realizados
        System.out.println("Número total de pedidos realizados: " + Pedido.getNumeroPedidos());
```

```
Pedido: PIZZA
ID del pedido: 1
Ingredientes extras: [Champiñones, Pepperoni]
Pedido: HAMBURGUESA
ID del pedido: 2
Ingredientes extras: [Queso]
Pedido: ENSALADA
ID del pedido: 3
Ingredientes extras: Ninguno
Pedido: PASTA
ID del pedido: 4
Ingredientes extras: [Albahaca]
Número total de pedidos realizados: 4
```

6. (HashMap) Un supermercado de productos ecológicos nos ha pedido hacer un programa para vender su mercancía. En esta primera versión del programa se tendrán en cuenta los productos que se indican en la tabla junto con su precio. Los productos se venden en bote, brick, etc. Cuando se realiza la compra, hay que indicar el producto y el número de unidades que se compran, por ejemplo "guisantes" si se quiere comprar un bote de guisantes y la cantidad, por ejemplo "3" si se quieren comprar 3 botes. La compra se termina con la palabra "fin". Para abreviar el problema, suponemos que el usuario no va a intentar comprar un producto que no existe. Utiliza un diccionario para almacenar los nombres y precios de los productos y una o varias listas para almacenar la compra que realiza el usuario.

   A continuación se muestra una tabla con los productos disponibles y sus respectivos precios:

avena	garbanzos	tomate	jengibre	quinoa	guisantes
2,21	2,39	1,59	3,13	4,50	1,60

   Ejemplo de ejecución:

```
Producto: tomate
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: avena
Cantidad: 1
Producto: tomate
Cantidad: 2
Producto: fin
```

```
Producto Precio Cantidad Subtotal
---------------------------------
tomate    1,59     1       1,59
quinoa    4,50     2       9,00
avena     2,21     1       2,21
tomate    1,59     2       3,18
---------------------------------
TOTAL: 15,98
```

7. (HashMap y Enum) Realiza una nueva versión del ejercicio anterior con las siguientes mejoras: Si algún producto se repite en diferentes líneas, se deben agrupar en una sola. Por ejemplo, si se pide primero 1 bote de tomate y luego 3 botes de tomate, en el extracto se debe mostrar que se han pedido 4 botes de tomate. Después de teclear "fin", el programa pide un código de descuento. Por ejemplo, si el usuario introduce el código "ECODTO", se aplica un 10% de descuento en la compra.

RELACIÓN DE EJERCICIOS 12 - Colecciones: ArrayList, HashSet, HashMap

Puede haber varios códigos descuento disponibles, que deberás guardar en un enum.

```
Producto: tomate
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: avena
Cantidad: 1
Producto: quinoa
Cantidad: 2
Producto: tomate
Cantidad: 2
Producto: fin
Introduzca código de descuento (INTRO si no tiene ninguno): ECODTO

Producto Precio Cantidad Subtotal
---------------------------------
tomate    1,59     3       4,77
quinoa    4,50     4      18,00
avena     2,21     1       2,21
---------------------------------
Descuento: 2,50
---------------------------------
TOTAL: 22,48
```