# EJERCICIOS PROGRAMACION III
# Práctica SEMANA 7

CONTENIDOS: Introducción MVC Simple (sin herencia o interfaces),
reparto de responsabilidades en POO y ejercicios básicos.

## Ejercicios obligatorios:

Todos los ejercicios deben ser implementados teniendo en cuenta las responsabilidades del patrón MVC impartido en la asignatura hasta la fecha (sin herencia/interfaces). La Vista gestionará toda la E/S, el controlador será un mero intermediario y orquestador de la aplicación y el Modelo tendrá el estado de la aplicación y la lógica de negocio.

1. Un profesor necesita sumar dos números enteros. Se pide crear un programa con el siguiente menú y se debe ceñir al patrón MVC.

   ```
   1.- Leer los números
   2.- Calcular la suma
   3.- Mostrar el resultado
   q.- Salir
   ```

2. Una compañía tiene cinco almacenes, cada uno de los cuales vende dos productos. En cada almacén, se dispone de una tabla para almacenar las ventas por unidades de los dos productos. Se dispone también de los precios que tiene cada unidad de los dos productos igual para todos los almacenes. Es necesaria la creación de una aplicación con el siguiente menú:

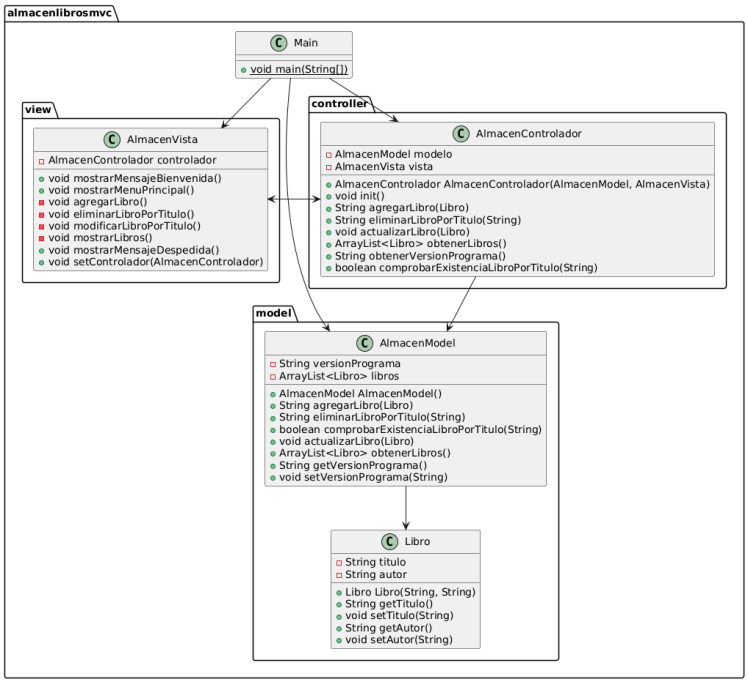
   ```
   1.- Leer tabla de ventas
   2.- Leer tabla de precios
   3.- Calcular ingresos totales
   4.- Mostrar resultados
   q.- Salir
   ```

   Es posible utilizar Arrays o Colecciones para resolver este ejercicio. Contempla los casos en los que no es posible utilizar las opciones 3 y 4 del menú.

3. Refactoriza los ejemplos proporcionados en la sesión anterior CRUD sobre colecciones, para que el proyecto siga el patrón MVC. Vigila la responsabilidad de cada clase y que no se realizan acciones que no les correspondan (e.g. E/S en el modelo o en el controlador, etc.).

# Ejercicios de extensión:

4. Refactoriza el ejercicio 4 de la sesión anterior para que siga el patrón MVC tratado en clase.
5. UML (*Unified Modeling Language*) es un lenguaje de modelado que trataréis más adelante en asignaturas de Ingeniería del Software. Entre los múltiples diagramas que existen podemos destacar los diagramas de clases, en los que se representan las clases con sus atributos, métodos y las relaciones existentes entre ellas. Un lenguaje textual para definir estos diagramas es PlantUML y su sintaxis la podéis encontrar aquí. Para convertir ese lenguaje PlantUML a una imagen podéis utilizar su editor online. **Este ejercicio consiste en generar un diagrama de paquetes con clases y sus atributos/métodos a partir de uno de los proyectos anteriores**. Se deberá crear el código en PlantUML para generar la imagen del diagrama de paquetes y clases del proyecto, ejemplo:




**TIP1**: Podéis utilizar esta utilidad que permite generar el código PlantUML a partir de los ficheros fuente .java de un proyecto. Aviso: no indica las relaciones entre clases y es necesario crear un paquete raíz común.

**TIP2**: Podéis utilizar inteligencias artificiales de vuestro gusto para obtener un primer boceto de archivo PlantUML (puml) a partir del código fuente java de vuestro proyecto. Aviso: es probable que el código puml generado tenga errores y lo tengáis que modificar para que refleje fielmente vuestro proyecto.

**TIP3**: Existe una extensión para VSCode para trabajar con ficheros puml que podéis instalar en vuestros ordenadores, pero quizás os sea más sencillo al principio utilizar el editor online de PlantUML.