# Explicación Del Código Caso2

# ¿Qué hace este sistema?

Este sistema permite registrar, buscar, actualizar y mostrar productos dentro de un inventario.  
El usuario puede interactuar a través de un menú en consola y ejecutar distintas opciones para gestionar los productos.  
Está construido aplicando los conceptos básicos de la Programación Orientada a Objetos (POO) en Java.

# Clases del sistema y estructura del código:

1. Producto.java

- Es una clase que representa un producto real del inventario.

- Tiene 4 atributos: nombre, código, precio y cantidad.

- Usa un constructor para inicializar estos valores.

- Ejemplo de abstracción: un objeto 'Producto' representa algo real con características específicas.

2. GestorInventario.java

- Es la clase que contiene toda la lógica del sistema (agregar, buscar, actualizar stock, etc).

- Tiene métodos separados por funcionalidad, lo cual aplica el principio de encapsulamiento.

- Se utilizan variables globales para facilitar la reutilización entre métodos.

- También se usa una lista de productos (ArrayList<Producto>) que actúa como el 'almacenamiento temporal' del inventario.

- Aquí se aplican validaciones para entradas vacías, códigos repetidos y conversiones de texto a número.

3. Main.java

- Es la clase que ejecuta el programa.

- Contiene el menú que el usuario ve y permite elegir las opciones con un 'switch'.

- Cada opción del menú llama a un método específico de la clase GestorInventario.

# ¿Dónde se usa la Programación Orientada a Objetos (POO)?

\* Abstracción:

- La clase Producto representa un producto real con solo los datos que necesitamos (nombre, precio, etc).

- Se ocultan los detalles técnicos al usuario final; solo interactúan con menús simples.

\* Encapsulamiento:

- Toda la lógica (agregar, buscar, actualizar) está dentro de la clase GestorInventario.

- El código está organizado en métodos para mantener todo separado y claro.

# Explicación de partes clave del código:

\* Scanner: Se usa para leer lo que el usuario escribe en consola.

\* ArrayList<Producto>: Es una lista que guarda todos los productos registrados.

\* switch-case: Permite ejecutar distintas acciones según la opción que el usuario elija.

\* trim(): Elimina espacios en blanco al principio y final del texto ingresado.

\* equals(""): Verifica si el usuario dejó un campo vacío.

\* Integer.parseInt(): Convierte texto a número entero para poder hacer operaciones.

# Validaciones que hace el sistema:

- No deja agregar productos con campos vacíos.

- No deja registrar dos productos con el mismo código.

- No deja usar letras donde deben ir números (precio y cantidad).

- No permite disminuir más stock del que hay disponible.