Universidad EAN

Taller 4

Esteban Mateo González Ruiz

Programación Orientada a Objetos

Docente: Jhan Yuler de la Pava Torres

Bogotá D.C, Colombia

2025

# Documentación del Proyecto Inventario - Papelería El Nevado

## 1. Descripción del problema

La papelería "El Nevado" requiere un sistema para gestionar el inventario de sus productos, registrar entradas y salidas, modificar datos y mantener el control actualizado sobre el stock. Actualmente, el control se realiza manualmente, lo que provoca errores y pérdida de información.  
  
El sistema debe permitir:  
- Registrar productos con atributos como ID, nombre, descripción, cantidad y precio.  
- Consultar el inventario completo.  
- Modificar los datos de productos existentes.  
- Registrar la salida de productos (ventas o uso interno).  
- Mantener persistencia de datos para conservar la información entre sesiones.

## 2. Solución implementada

Se desarrolló una aplicación Java con interfaz gráfica Swing organizada en varias ventanas:  
  
- RegistrarProducto: Permite añadir nuevos productos al inventario.  
- MostrarProducto: Muestra el listado de productos existentes con opción a consultar, modificar y registrar salidas.  
- ModificarProducto: Permite actualizar atributos de un producto.  
- RegistrarSalida: Permite registrar la salida de inventario por producto y cantidad.  
  
El sistema utiliza un controlador central que enlaza las vistas con la lógica y la persistencia. La persistencia se maneja mediante archivos de texto que almacenan la información de productos.

## 3. Reflexión sobre SOLID aplicado al proyecto

El principio SOLID consiste en 5 reglas para diseñar software orientado a objetos de manera modular y mantenible. A continuación, la reflexión sobre cómo se aplican estos principios en el proyecto:  
  
- S - Principio de Responsabilidad Única (Single Responsibility Principle):   
 Cada clase tiene una función clara y única:   
 - Las vistas solo manejan la interfaz gráfica.   
 - La clase Controlador maneja la lógica y coordinación entre vistas y modelo.   
 - La persistencia está separada en la clase PersistenciaInventario, encargada exclusivamente del acceso a archivos.   
 Esto facilita mantenimiento y evita que una clase tenga múltiples responsabilidades.  
  
- O - Principio de Abierto/Cerrado (Open/Closed Principle):   
 Las clases están diseñadas para que se puedan extender (p.ej., nuevas ventanas o funcionalidades) sin modificar las existentes. Por ejemplo, agregar un nuevo tipo de producto o método de registro no debe afectar el código base.  
  
- L - Principio de Sustitución de Liskov (Liskov Substitution Principle):   
 En este proyecto no hay mucha herencia aún, pero si se introduce herencia para tipos de productos, se debe asegurar que las subclases puedan reemplazar a la superclase sin romper la funcionalidad.  
  
- I - Principio de Segregación de Interfaces (Interface Segregation Principle):   
 Las interfaces (si se usan) deben ser específicas para cada necesidad. Aquí, cada vista tiene su propio contrato con el controlador, evitando una interfaz general que obligue a implementar métodos no necesarios.  
  
- D - Principio de Inversión de Dependencias (Dependency Inversion Principle):   
 Las clases de alto nivel (controlador) no dependen de detalles concretos (como el almacenamiento en archivo) sino de abstracciones o interfaces (podría mejorarse implementando interfaces para persistencia), facilitando cambiar la forma de almacenamiento sin afectar la lógica.

## 4. Diagrama de clases (descripción textual)

