Materia:	Programación II		
Nivel:	2º Cuatrimestre		
Tipo de Examen:	Primer Parcial		
Apellido(1):		Fecha:	
Nombre/s(1):		Docente a cargo(2):	
División(1):		Nota(2):	
DNI(1):		Firma(2):	

(1) Campos a completar solo por el estudiante en caso de imprimir este enunciado en papel.
(2) Campos a completar solo por el docente en caso de imprimir este enunciado en papel.

Objetivo General

De acuerdo con las descripciones de las siguientes clases, se pide: Modelar en **UML** (adjuntando la imagen en la entrega) y posteriormente crear el código correspondiente en **Java** (en todos los casos, reutilizar código).

Desarrollar un sistema que permita gestionar un inventario de productos con persistencia de datos a través de archivos. El sistema deberá implementar un CRUD completo para gestionar los productos del inventario, incorporando serialización y manejo de excepciones. Este proyecto integrará los conceptos de Programación Orientada a Objetos en Java, incluyendo herencia, polimorfismo y la organización en paquetes.

Contexto

Una empresa necesita un sistema que gestione un inventario digital de productos, con la capacidad de guardar y recuperar la información en archivos para garantizar la persistencia de los datos.

Funcionalidades CRUD

- Crear: Agregar productos al inventario.
- Leer: Consultar productos individuales o mostrar la lista completa.
- Actualizar: Modificar los datos de un producto existente.
- Eliminar: Eliminar productos del inventario.

Estructura del Proyecto

1. Herencia y Polimorfismo

- Jerarquía de Productos

Crear una jerarquía de clases para representar diferentes tipos de productos:

- Clase abstracta Producto:
    - Atributos comunes:
        - String codigo
        - String nombre
        - double precio
        - int stock
    - Métodos abstractos:
        - void mostrarDetalles()
        - boolean verificarDisponibilidad(int cantidad)
- Clases hijas:
    - Electrodomestico:
        - Atributos adicionales:
            - int garantiaMeses
            - String categoriaEnergetica
        - Método específico: boolean esBajoConsumo()
    - Alimento:
        - Atributos adicionales:
            - LocalDate fechaVencimiento
            - boolean refrigeracionRequerida
        - Método específico: boolean estaVencido()

2. Serialización y Manejo de Archivos
    - Implementar métodos para guardar el inventario completo en un archivo usando serialización (Serializable).
    - Implementar métodos para cargar el inventario desde un archivo.
3. Excepciones y Paquetes
    - Excepciones personalizadas:
        - ProductoNoEncontradoException

- StockInsuficienteException
- Paquetes sugeridos:
  - com.inventario.productos
  - com.inventario.gestores
  - com.inventario.excepciones

4. Interfaz Funcional y Filtrado
   - Interfaz funcional FiltroProducto para realizar búsquedas personalizadas, como productos con stock bajo o por categoría.
5. Extras
   - Generar un reporte del inventario en formato de archivo de texto.
   - (Bonus) Crear una interfaz gráfica sencilla para interactuar con el sistema.

MAIN

```
import com.inventario.productos.*;
import com.inventario.gestores.*;
import com.inventario.excepciones.*;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        GestorInventario gestor = new GestorInventario();
        Scanner scanner = new Scanner(System.in);
        int opcion;
        do {
            System.out.println("\n--- Gestión de Inventario ---");
            System.out.println("1. Agregar Producto");
            System.out.println("2. Mostrar Todos los Productos");
```

```
            System.out.println("3. Buscar Producto por Código");
            System.out.println("4. Modificar Producto");
            System.out.println("5. Eliminar Producto");
            System.out.println("6. Guardar Inventario en Archivo");
            System.out.println("7. Cargar Inventario desde Archivo");
            System.out.println("8. Salir");
            System.out.print("Seleccione una opción: ");
            opcion = scanner.nextInt();
            switch (opcion) {
                case 1 -> gestor.agregarProductoDesdeConsola(scanner);
                case 2 -> gestor.mostrarInventario();
                case 3 -> gestor.buscarProductoPorCodigoDesdeConsola(scanner);
                case 4 -> gestor.modificarProductoDesdeConsola(scanner);
                case 5 -> gestor.eliminarProductoDesdeConsola(scanner);
                case 6 -> gestor.guardarInventario("inventario.dat");
                case 7 -> gestor.cargarInventario("inventario.dat");
                case 8 -> System.out.println("Saliendo del sistema...");
                default -> System.out.println("Opción inválida. Intente nuevamente.");
            }
        } while (opcion != 8);
        scanner.close();
    }
}
```

IMPORTANTE:

- Dos (2) errores en el mismo tema anulan su puntaje.
- NO se corregirán exámenes que NO compilen.
- NO se corregirán exámenes que NO contengan la imagen del modelado en UML.
- No se corregirán proyectos que no sea identificable su autor.
- Reutilizar tanto código como sea posible.
- Colocar nombre de la clase (en estáticos), this o super en todos los casos que corresponda.
- Subir los proyectos y la imagen UML en un único archivo .7z, .zip, .rar o similar. Nombrarlo con su Apellido.Nombre.

Consideraciones Finales:

- Fecha de entrega: 29 de noviembre de 2024 (18.30 hs)
- Trabajo individual
- Presentación oral y defensa del proyecto (29 de noviembre de 2024)