UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional de Avellaneda
Tecnicatura Universitaria en Programación
Modalidad a Distancia – Ciclo Lectivo 2025

TRABAJO PRÁCTICO Nº6 COLECCIONES

Estudiante: Agustina Milagros Cruz

Materia: Programación II

Profesor: Ariel Enferrel

Comisión: 12

Año: 2025

UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional de Avellaneda
Tecnicatura Universitaria en Programación
Modalidad a Distancia – Ciclo Lectivo 2025

Objetivo General

Desarrollar un sistema de stock utilizando colecciones dinámicas (ArrayList) y enumeraciones (enum) en Java, aplicando los principios de la Programación Orientada a Objetos (POO).

LINK REPOSITORIO GITHUB: https://github.com/agustinamilagroscruz/UTN-TUPaD-P2/tree/main/src

Caso Práctico 1: Sistema de Stock Consigna

general:

Se debe desarrollar un sistema de stock que permita gestionar productos en una tienda, controlando su disponibilidad, precios y categorías.
La información se modelará utilizando **clases, colecciones dinámicas y enumeraciones.**

1. Clase Producto

Consigna:
Implementar una clase Producto con los siguientes atributos y métodos:

- id (String)
- nombre (String)
- precio (double)
- cantidad (int)
- categoria (CategoriaProducto)
- mostrarInfo() → Muestra la información del producto por consola.

1.

```
public class Producto {     private String id;
private String nombre;     private double
precio;     private int cantidad;     private
CategoriaProducto categoria;


   public Producto(String id, String nombre, double precio, int cantidad,
CategoriaProducto categoria) {
      this.id = id;
```

```
        this.nombre = nombre;
this.precio = precio;        this.cantidad =
cantidad;        this.categoria = categoria;
    }

    public String getId() { return id; }     public void setCantidad(int
cantidad) { this.cantidad = cantidad; }     public int getCantidad() { return
cantidad; }     public double getPrecio() { return precio; }     public
CategoriaProducto getCategoria() { return categoria; }

    public void mostrarInfo() {
        System.out.println("ID: " + id + " | Nombre: " + nombre + " | Precio: $" + precio +
                " | Cantidad: " + cantidad + " | Categoría: " + categoria.getDescripcion());
    }

    @Override     public String toString() {        return nombre + " (" + categoria + ") - $" +
precio + " [" + cantidad + " unidades]";
    }
}
```

2. Enum CategoriaProducto

Consigna:
Crear una enumeración CategoriaProducto que represente las categorías de los productos con descripciones.

2.

```
public enum CategoriaProducto {     ALIMENTOS("Productos
comestibles"),
```

```
    ELECTRONICA("Dispositivos electrónicos"),
    ROPA("Prendas de vestir"),
    HOGAR("Artículos para el hogar");

    private final String descripcion;

    CategoriaProducto(String descripcion) {        this.descripcion
= descripcion;
    }

    public String getDescripcion() {
return descripcion;
    }

    @Override     public String toString() {
return name() + " - " + descripcion;
    }
}
```

3. Clase Inventario

Consigna:
Implementar la clase Inventario con un ArrayList<Producto> y los siguientes métodos:

- agregarProducto(Producto p)
- listarProductos()
- buscarProductoPorId(String id)
- eliminarProducto(String id)
- actualizarStock(String id, int nuevaCantidad)
- filtrarPorCategoria(CategoriaProducto categoria)
- obtenerTotalStock()
- obtenerProductoConMayorStock()
- filtrarProductosPorPrecio(double min, double max)
- mostrarCategoriasDisponibles()

3.

```
import java.util.ArrayList;


public class Inventario {


   private ArrayList<Producto> productos;


   public Inventario() {        productos
= new ArrayList<>();
   }


   public void agregarProducto(Producto p) {        productos.add(p);
   }


   public void listarProductos() {
if (productos.isEmpty()) {
         System.out.println("No hay productos en el inventario.");
      } else {
         for (Producto p : productos) {
p.mostrarInfo();
         }
      }
   }


   public Producto buscarProductoPorId(String id) {
for (Producto p : productos) {           if
(p.getId().equalsIgnoreCase(id)) {              return p;
         }
```

```
        }
        return null;
    }

    public void eliminarProducto(String id) {
        Producto encontrado = buscarProductoPorId(id);
if (encontrado != null) {
productos.remove(encontrado);
            System.out.println("Producto eliminado correctamente.");
        } else {
            System.out.println("No se encontró un producto con el ID especificado.");
        }
    }

    public void actualizarStock(String id, int nuevaCantidad) {
Producto p = buscarProductoPorId(id);        if (p != null) {
            p.setCantidad(nuevaCantidad);
            System.out.println("Stock actualizado correctamente.");
        } else {
            System.out.println("No se encontró el producto.");
        }
    }

    public void filtrarPorCategoria(CategoriaProducto categoria) {
System.out.println("Productos de la categoría: " + categoria);        for
(Producto p : productos) {            if (p.getCategoria() == categoria) {
p.mostrarInfo();
            }
        }
    }
```

```
    public int obtenerTotalStock() {
        int total = 0;        for (Producto
p : productos) {            total +=
p.getCantidad();
        }
        return total;
    }

    public Producto obtenerProductoConMayorStock() {        if
(productos.isEmpty()) return null;

        Producto mayor = productos.get(0);        for
(Producto p : productos) {            if (p.getCantidad()
> mayor.getCantidad()) {                mayor = p;
            }
        }
        return mayor;
    }

    public void filtrarProductosPorPrecio(double min, double max) {
System.out.println("Productos con precio entre $" + min + " y $" + max + ":");          for
(Producto p : productos) {                if (p.getPrecio() >= min && p.getPrecio() <= max) {
p.mostrarInfo();
            }
        }
    }
```

```
   public void mostrarCategoriasDisponibles() {
System.out.println("Categorías disponibles:");       for
(CategoriaProducto c : CategoriaProducto.values()) {
          System.out.println("- " + c);
      }
   }
}
```

4. Clase Main

Consigna:
Probar todas las funcionalidades del sistema creando productos, agregándolos al inventario y ejecutando los métodos anteriores.

```
public class Main {     public static void
main(String[] args) {

      Inventario inventario = new Inventario();

      // Crear productos
      Producto p1 = new Producto("P001", "Arroz", 1500, 30,
CategoriaProducto.ALIMENTOS);
      Producto p2 = new Producto("P002", "Televisor", 250000, 5,
CategoriaProducto.ELECTRONICA);
     Producto p3 = new Producto("P003", "Remera", 5000, 15,
CategoriaProducto.ROPA);
      Producto p4 = new Producto("P004", "Licuadora", 32000, 8,
CategoriaProducto.HOGAR);
      Producto p5 = new Producto("P005", "Pan", 1200, 50,
CategoriaProducto.ALIMENTOS);

      // Agregar productos
inventario.agregarProducto(p1);
inventario.agregarProducto(p2);
```

```
inventario.agregarProducto(p3);
inventario.agregarProducto(p4);
inventario.agregarProducto(p5);

    // Listar productos
    System.out.println("LISTA DE PRODUCTOS:");       inventario.listarProductos();

    // Buscar producto por ID
    System.out.println("\n BUSCAR PRODUCTO POR ID:");       Producto
buscado = inventario.buscarProductoPorId("P003");       if (buscado !=
null) buscado.mostrarInfo();

    // Filtrar por categoría
    System.out.println("\n FILTRO POR CATEGORÍA:");
inventario.filtrarPorCategoria(CategoriaProducto.ALIMENTOS);

    // Eliminar un producto
    System.out.println("\n ELIMINAR PRODUCTO:");
inventario.eliminarProducto("P002");

   // Actualizar stock
    System.out.println("\n ACTUALIZAR STOCK:");
inventario.actualizarStock("P001", 60);

    // Mostrar total de stock
    System.out.println("\n TOTAL DE STOCK DISPONIBLE: " +
inventario.obtenerTotalStock());

    // Producto con mayor stock
    System.out.println("\n PRODUCTO CON MAYOR STOCK:");
```

```
        System.out.println(inventario.obtenerProductoConMayorStock());


        // Filtrar por precio
        System.out.println("\n FILTRAR POR PRECIO ENTRE $1000 Y $30000:");
inventario.filtrarProductosPorPrecio(1000, 30000);


        // Categorías disponibles
        System.out.println("\n CATEGORÍAS DISPONIBLES:");
inventario.mostrarCategoriasDisponibles();
    }
}
```

Conclusiones

Este trabajo práctico permitió:

- Aplicar colecciones dinámicas (ArrayList) y enumeraciones (enum).
- Comprender el uso de encapsulamiento y búsqueda dentro de listas.
- Practicar la creación de métodos de filtrado, eliminación y actualización.
- Reforzar el diseño modular y reutilizable del paradigma orientado a objetos.