TRABAJO PRACTICO 6 – COLECCIONES

ALUMNO: EMILIANO JARA

🧰 Caso Práctico 1: Inventario con ArrayList y enum (Java)

1) enum CategoriaProducto (con descripciones)

El enunciado sugiere valores y una versión con descripción. Te dejo la versión completa

```
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;
  }
}
```

2) Clase Producto

Atributos y método mostrarInfo() según el PDF. Agregamos toString() para mejor depuración (parte de las conclusiones esperadas).

```
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 String getNombre() { return nombre; }
    public double getPrecio() { return precio; }
    public int getCantidad() { return cantidad; }
    public CategoriaProducto getCategoria() { return categoria; }

    public void setCantidad(int cantidad) { this.cantidad = cantidad; }

    public void mostrarInfo() {
        System.out.printf(
            "Producto: %s (ID: %s) | Precio: $%.2f | Stock: %d | Categoría: %s - %s%n",
            nombre, id, precio, cantidad, categoria.name(), categoria.getDescripcion()
        );
    }
```

```
  @Override
  public String toString() {
    return String.format("Producto{id='%s', nombre='%s', precio=%.2f,
cantidad=%d, categoria=%s}",
        id, nombre, precio, cantidad, categoria);
  }
}
```

3) Clase Inventario

Métodos requeridos por el enunciado: agregar, listar, buscar, eliminar, actualizar stock, filtrar por categoría, total, mayor stock, filtro por precio y mostrar categorías disponibles.

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

public class Inventario {
  private ArrayList<Producto> productos = new ArrayList<>();

  // 1) Agregar producto
  public void agregarProducto(Producto p) {
    productos.add(p);
  }

  // 2) Listar productos
  public void listarProductos() {
```

```
    if (productos.isEmpty()) {
        System.out.println("No hay productos en el inventario.");
        return;
    }
    for (Producto p : productos) {
        p.mostrarInfo();
    }
}

// 3) Buscar producto por ID
public Producto buscarProductoPorId(String id) {
    for (Producto p : productos) {
        if (p.getId().equalsIgnoreCase(id)) {
            return p;
        }
    }
    return null;
}

// 4) Eliminar producto por ID
public boolean eliminarProducto(String id) {
    Producto p = buscarProductoPorId(id);
    if (p != null) {
        return productos.remove(p);
    }
    return false;
}
```

```
// 5) Actualizar stock por ID
public boolean actualizarStock(String id, int nuevaCantidad) {
   Producto p = buscarProductoPorId(id);
   if (p != null) {
      p.setCantidad(nuevaCantidad);
      return true;
   }
   return false;
}

// 6) Filtrar por categoría
public List<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
   return productos.stream()
         .filter(p -> p.getCategoria() == categoria)
         .collect(Collectors.toList());
}

// 7) Obtener total de stock
public int obtenerTotalStock() {
   int total = 0;
   for (Producto p : productos) {
      total += p.getCantidad();
   }
   return total;
}

// 8) Producto con mayor stock
public Producto obtenerProductoConMayorStock() {
```

```
        return productos.stream()
                .max(Comparator.comparingInt(Producto::getCantidad))
                .orElse(null);
    }

    // 9) Filtrar productos por precio entre min y max
    public List<Producto> filtrarProductosPorPrecio(double min, double max) {
        return productos.stream()
                .filter(p -> p.getPrecio() >= min && p.getPrecio() <= max)
                .collect(Collectors.toList());
    }

    // 10) Mostrar categorías disponibles con sus descripciones
    public void mostrarCategoriasDisponibles() {
        for (CategoriaProducto c : CategoriaProducto.values()) {
            System.out.printf("%s: %s%n", c.name(), c.getDescripcion());
        }
    }
}
```

4) Main con TODAS las tareas (1–10)

Este Main corre punto por punto lo que pide el trabajo

```
public class Main {
    public static void main(String[] args) {
        Inventario inventario = new Inventario();

        // 1. Crear al menos cinco productos y agregarlos
```

```
        Producto p1 = new Producto("A1", "Arroz", 1200.0, 50,
CategoriaProducto.ALIMENTOS);
        Producto p2 = new Producto("E1", "Auriculares", 3500.0, 20,
CategoriaProducto.ELECTRONICA);
        Producto p3 = new Producto("R1", "Remera", 2500.0, 35,
CategoriaProducto.ROPA);
        Producto p4 = new Producto("H1", "Almohada", 1800.0, 15,
CategoriaProducto.HOGAR);
        Producto p5 = new Producto("E2", "Mouse", 2900.0, 40,
CategoriaProducto.ELECTRONICA);

        inventario.agregarProducto(p1);
        inventario.agregarProducto(p2);
        inventario.agregarProducto(p3);
        inventario.agregarProducto(p4);
        inventario.agregarProducto(p5);

        // 2. Listar todos los productos
        System.out.println("== LISTADO DE PRODUCTOS ==");
        inventario.listarProductos();

        // 3. Buscar por ID
        System.out.println("\n== BUSCAR PRODUCTO POR ID ==");
        Producto buscado = inventario.buscarProductoPorId("E1");
        System.out.println(buscado != null ? "Encontrado: " + buscado : "No se
encontró el producto");

        // 4. Filtrar por categoría
        System.out.println("\n== FILTRO POR CATEGORÍA: ELECTRONICA ==");
```

```
    for (Producto p :
inventario.filtrarPorCategoria(CategoriaProducto.ELECTRONICA)) {
        p.mostrarInfo();
    }

    // 5. Eliminar por ID
    System.out.println("\n== ELIMINAR PRODUCTO CON ID 'H1' ==");
    boolean eliminado = inventario.eliminarProducto("H1");
    System.out.println("Eliminado: " + eliminado);
    System.out.println("Productos restantes:");
    inventario.listarProductos();

    // 6. Actualizar stock
    System.out.println("\n== ACTUALIZAR STOCK DE 'A1' A 80 ==");
    boolean actualizado = inventario.actualizarStock("A1", 80);
    System.out.println("Actualizado: " + actualizado);
    inventario.listarProductos();

    // 7. Total de stock
    System.out.println("\n== TOTAL DE STOCK DISPONIBLE ==");
    System.out.println("Total: " + inventario.obtenerTotalStock());

    // 8. Producto con mayor stock
    System.out.println("\n== PRODUCTO CON MAYOR STOCK ==");
    Producto mayor = inventario.obtenerProductoConMayorStock();
    System.out.println(mayor != null ? mayor : "Inventario vacío");

    // 9. Filtrar por precio entre $1000 y $3000
```

```
        System.out.println("\n== FILTRO POR PRECIO ($1000 - $3000) ==");
        for (Producto p : inventario.filtrarProductosPorPrecio(1000, 3000)) {
            p.mostrarInfo();
        }

        // 10. Mostrar categorías disponibles y sus descripciones
        System.out.println("\n== CATEGORÍAS DISPONIBLES ==");
        inventario.mostrarCategoriasDisponibles();
    }
}
```

Ejercicio Propuesto 2: Biblioteca, Autor y Libro (Composición 1 a N)

Estructura mínima para que arranques y luego completes las tareas del PDF.

```
// Autor.java
public class Autor {
    private String id;
    private String nombre;
    private String nacionalidad;

    public Autor(String id, String nombre, String nacionalidad) {
        this.id = id;
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }

    public String getId() { return id; }
    public String getNombre() { return nombre; }
```

```
  public void mostrarInfo() {
    System.out.printf("Autor: %s (ID: %s) | Nacionalidad: %s%n", nombre, id, nacionalidad);
  }
}
```

🏫 Ejercicio: Universidad, Profesor y Curso (Bidireccional 1 a N)

Te dejo el esqueleto que garantiza la **bidireccionalidad** y mantiene el **invariante de asociación** (sincroniza ambos lados siempre).

```
// Profesor.java
import java.util.ArrayList;
import java.util.List;

public class Profesor {
  private String id;
  private String nombre;
  private String especialidad;
  private List<Curso> cursos = new ArrayList<>();

  public Profesor(String id, String nombre, String especialidad) {
    this.id = id;
    this.nombre = nombre;
    this.especialidad = especialidad;
  }

  public String getId() { return id; }
  public String getNombre() { return nombre; }

  // Agrega curso si no está y sincroniza el lado del curso
```

```
public void agregarCurso(Curso c) {
    if (!cursos.contains(c)) {
        cursos.add(c);
    }
    if (c.getProfesor() != this) {
        c.setProfesor(this); // sincroniza del otro lado
    }
}

// Quita curso y sincroniza
public void eliminarCurso(Curso c) {
    if (cursos.remove(c)) {
        if (c.getProfesor() == this) {
            c.setProfesor(null); // rompe relación del otro lado
        }
    }
}

public void listarCursos() {
    System.out.println("Cursos de " + nombre + ":");
    for (Curso c : cursos) {
        System.out.printf("- %s (%s)%n", c.getNombre(), c.getCodigo());
    }
}

public void mostrarInfo() {
    System.out.printf("Profesor: %s (ID: %s) | Esp.: %s | Cursos: %d%n",
            nombre, id, especialidad, cursos.size());}}
```