

PROGRAMACIÓN II

Trabajo Práctico 6: Colecciones y Sistema de Stock

Caso Práctico 1

1. Descripción 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 en Java.

```
// Enum Categoria Productos
public enum Categoria {
    ELECTRONICA,
    ROPA,
    ALIMENTOS,
    HOGAR
}
```

ClaseProductos.java

```
public class Producto {
    private String nombre;
    private int stock;
    private double precio;
    private Categoria categoria;

    public Producto(String nombre, int stock, double precio, Categoria categoria)
    {
        this.nombre = nombre;
        this.stock = stock;
        this.precio = precio;
    }
}
```

```
        this.categoria = categoria;
    }

    public int getStock() {
        return stock;
    }

    public void setStock(int stock) {
        this.stock = stock;
    }

    public String getNombre() {
        return nombre;
    }

    public Categoria getCategoría() {
        return categoria;
    }

    @Override
    public String toString() {
        return "Producto{" +
            "nombre=\"" + nombre + '\"' +
            ", stock=" + stock +
            ", precio=" + precio +
            ", categoria=" + categoria +
            '}';
    }
}
```

Clase Sistema Stock

```
import java.util.ArrayList;

public class SistemaStock {
    private ArrayList<Producto> productos;

    public SistemaStock() {
        this.productos = new ArrayList<>();
    }

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

    // Listar todos los productos
    public void listarProductos() {
        for (Producto p : productos) {
            System.out.println(p);
        }
    }

    // Buscar producto por nombre
    public Producto buscarProducto(String nombre) {
        for (Producto p : productos) {
            if (p.getNombre().equalsIgnoreCase(nombre)) {
                return p;
            }
        }
    }
}
```

```

    }

    return null; // no encontrado

}

// Mostrar productos por categoría
public void listarPorCategoria(Categoría categoria) {
    for (Producto p : productos) {
        if (p.getCategoría() == categoria) {
            System.out.println(p);
        }
    }
}

```

2. Clases a implementar **Clase Producto**

```

public enum CategoríaProducto {
    ALIMENTOS("Productos comestibles"),
    ELECTRONICA("Dispositivos electrónicos"),
    ROPA("Prendas de vestir"),
    HOGAR("Artículos para el hogar");

    private final String descripción;

    CategoríaProducto(String descripción) {
        this.descripción = descripción;
    }

    public String getDescripción() {

```

```
        return descripcion;  
    }  
}
```

Clase producto

```
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;  
    }  
  
    // Getters y Setters  
    public String getId() { return id; }  
    public String getNombre() { return nombre; }  
    public double getPrecio() { return precio; }  
    public void setPrecio(double precio) { this.precio = precio; }  
    public int getCantidad() { return cantidad; }  
    public void setCantidad(int cantidad) { this.cantidad = cantidad; }
```

```
public CategoriaProducto getCategoría() { return categoria; }

// Método para mostrar información
public void mostrarInfo() {
    System.out.println("ID: " + id +
        ", Nombre: " + nombre +
        ", Precio: $" + precio +
        ", Cantidad: " + cantidad +
        ", Categoría: " + categoria +
        " (" + categoria.getDescripcion() + ")");
}

}
```

Clase inventario

```
import java.util.ArrayList;

public class inventario {
    private ArrayList<Producto> productos;

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

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

    // Listar todos los productos
    public void listarProductos() {
```

```
for (Producto p : productos) {
    p.mostrarInfo();
}

}

// Buscar producto por ID

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

// Eliminar producto por ID

public void eliminarProducto(String id) {
    productos.removeIf(p -> p.getId().equalsIgnoreCase(id));
}

// Actualizar stock

public void actualizarStock(String id, int nuevaCantidad) {
    Producto p = buscarProductoPorId(id);
    if (p != null) {
        p.setCantidad(nuevaCantidad);
    }
}

// Filtrar por categoría
```

```
public void filtrarPorCategoria(CategoríaProducto categoria) {  
    for (Producto p : productos) {  
        if (p.getCategoría() == categoria) {  
            p.mostrarInfo();  
        }  
    }  
}  
  
// Obtener total de stock  
public int obtenerTotalStock() {  
    int total = 0;  
    for (Producto p : productos) {  
        total += p.getCantidad();  
    }  
    return total;  
}  
  
// Obtener producto con mayor stock  
public Producto obtenerProductoConMayorStock() {  
    if (productos.isEmpty()) return null;  
  
    Producto max = productos.get(0);  
    for (Producto p : productos) {  
        if (p.getCantidad() > max.getCantidad()) {  
            max = p;  
        }  
    }  
    return max;  
}
```

```
// Filtrar productos por precio  
  
public void filtrarProductosPorPrecio(double min, double max) {  
    for (Producto p : productos) {  
        if (p.getPrecio() >= min && p.getPrecio() <= max) {  
            p.mostrarInfo();  
        }  
    }  
}
```

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

Clase Main

```
public class Main {  
    public static void main(String[] args) {  
        Inventario inventario = new Inventario();  
  
        // Crear productos  
  
        Producto p1 = new Producto("P001", "Leche", 1.2, 50,  
        CategoriaProducto.ALIMENTOS);  
  
        Producto p2 = new Producto("P002", "TV", 450.0, 10,  
        CategoriaProducto.ELECTRONICA);  
  
        Producto p3 = new Producto("P003", "Pantalón", 35.0, 30,  
        CategoriaProducto.ROPA);
```

```
// Agregar productos
inventario.agregarProducto(p1);
inventario.agregarProducto(p2);
inventario.agregarProducto(p3);

// Listar productos
System.out.println("==> Lista de Productos ==>");
inventario.listarProductos();

// Filtrar por categoría
System.out.println("\n==> Productos ELECTRONICA ==>");
inventario.filtrarPorCategoria(CategoriaProducto.ELECTRONICA);

// Obtener producto con mayor stock
System.out.println("\n==> Producto con mayor stock ==>");
Producto maxStock = inventario.obtenerProductoConMayorStock();
if (maxStock != null) maxStock.mostrarInfo();

// Filtrar por precio
System.out.println("\n==> Productos entre $30 y $500 ==>");
inventario.filtrarProductosPorPrecio(30, 500);

// Mostrar categorías disponibles
System.out.println("\n==> Categorías disponibles ==>");
inventario.mostrarCategoriasDisponibles();
}

}
```

3. Clase Autor

```
public class Autor {  
    private String nombre;  
    private String nacionalidad;  
  
    public Autor(String nombre, String nacionalidad) {  
        this.nombre = nombre;  
        this.nacionalidad = nacionalidad;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public String getNacionalidad() {  
        return nacionalidad;  
    }  
  
    @Override  
    public String toString() {  
        return "Autor{" +  
               "nombre=\"" + nombre + "\"" +  
               ", nacionalidad=\"" + nacionalidad + "\"" +  
               '}';  
    }  
}
```

Clase libro

```
public class Libro {
```

```
private String titulo;  
private String isbn;  
private int anioPublicacion;  
private Autor autor;  
  
public Libro(String titulo, String isbn, int anioPublicacion, Autor autor) {  
    this.titulo = titulo;  
    this.isbn = isbn;  
    this.anioPublicacion = anioPublicacion;  
    this.autor = autor;  
}  
  
public String getIsbn() {  
    return isbn;  
}  
  
public int getAnioPublicacion() {  
    return anioPublicacion;  
}  
  
public Autor getAutor() {  
    return autor;  
}  
  
public void mostrarInfo() {  
    System.out.println("Título: " + titulo +  
        ", ISBN: " + isbn +  
        ", Año: " + anioPublicacion +  
        ", Autor: " + autor.getNombre() +
```

```
    " (" + autor.getNacionalidad() + ")");
}

}
```

Clase biblioteca

```
import java.util.ArrayList;

public class Biblioteca {

    private ArrayList<Libro> libros;

    public Biblioteca() {
        libros = new ArrayList<>();
    }

    // Agregar libro
    public void agregarLibro(Libro libro) {
        libros.add(libro);
    }

    // Listar todos los libros
    public void listarLibros() {
        for (Libro l : libros) {
            l.mostrarInfo();
        }
    }

    // Buscar libro por ISBN
    public Libro buscarPorISBN(String isbn) {
        for (Libro l : libros) {
```

```
if (l.getIsbn().equalsIgnoreCase(isbn)) {
    return l;
}
return null;
}

// Filtrar libros por año de publicación
public void filtrarPorAnio(int anio) {
    for (Libro l : libros) {
        if (l.getAnioPublicacion() == anio) {
            l.mostrarInfo();
        }
    }
}

// Eliminar libro por ISBN
public void eliminarPorISBN(String isbn) {
    libros.removeIf(l -> l.getIsbn().equalsIgnoreCase(isbn));
}

// Cantidad total de libros
public int cantidadTotal() {
    return libros.size();
}

// Listar todos los autores de los libros disponibles
public void listarAutores() {
    ArrayList<Autor> autores = new ArrayList<>();
```

```

for (Libro l : libros) {
    if (!autores.contains(l.getAutor())) {
        autores.add(l.getAutor());
    }
}
for (Autor a : autores) {
    System.out.println(a);
}
}
}

```

Clase Main

```

public class Main {
    public static void main(String[] args) {
        // 1. Crear biblioteca
        Biblioteca biblioteca = new Biblioteca();

        // 2. Crear autores
        Autor autor1 = new Autor("Gabriel García Márquez", "Colombiano");
        Autor autor2 = new Autor("J.K. Rowling", "Británica");
        Autor autor3 = new Autor("George Orwell", "Británico");

        // 3. Crear libros y asociar a autores
        Libro libro1 = new Libro("Cien Años de Soledad", "ISBN001", 1967,
        autor1);
        Libro libro2 = new Libro("Harry Potter y la Piedra Filosofal", "ISBN002",
        1997, autor2);
        Libro libro3 = new Libro("Harry Potter y la Cámara Secreta", "ISBN003",
        1998, autor2);
        Libro libro4 = new Libro("1984", "ISBN004", 1949, autor3);
    }
}

```

```
Libro libro5 = new Libro("Animal Farm", "ISBN005", 1945, autor3);

// Agregar libros a la biblioteca
biblioteca.agregarLibro(libro1);
biblioteca.agregarLibro(libro2);
biblioteca.agregarLibro(libro3);
biblioteca.agregarLibro(libro4);
biblioteca.agregarLibro(libro5);

// 4. Listar todos los libros
System.out.println("==> Todos los libros ==>");
biblioteca.listarLibros();

// 5. Buscar libro por ISBN
System.out.println("\n==> Buscar libro ISBN002 ==>");
Libro buscado = biblioteca.buscarPorISBN("ISBN002");
if (buscado != null) buscado.mostrarInfo();

// 6. Filtrar por año
System.out.println("\n==> Libros publicados en 1998 ==>");
biblioteca.filtrarPorAnio(1998);

// 7. Eliminar libro por ISBN
System.out.println("\n==> Eliminar libro ISBN005 ==>");
biblioteca.eliminarPorISBN("ISBN005");
System.out.println("Libros restantes:");
biblioteca.listarLibros();

// 8. Cantidad total de libros
```

```

        System.out.println("\nCantidad total de libros: " +
biboteca.cantidadTotal());

// 9. Listar todos los autores

System.out.println("\n==== Autores disponibles en la biblioteca ===");
biboteca.listarAutores();

}

}

```

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

1.

Clase Profesor

```
import java.util.ArrayList;
```

```

public class Profesor {

    private String nombre;
    private String legajo;
    private ArrayList<Curso> cursos;

```

```

public Profesor(String nombre, String legajo) {
    this.nombre = nombre;
    this.legajo = legajo;
    this.cursos = new ArrayList<>();
}

}
```

```

public String getNombre() { return nombre; }
public String getLegajo() { return legajo; }
public ArrayList<Curso> getCursos() { return cursos; }
```

```

// Agregar curso a la lista del profesor (solo desde Curso)
protected void agregarCurso(Curso curso) {
    if (!cursos.contains(curso)) {
        cursos.add(curso);
    }
}

// Eliminar curso de la lista del profesor (solo desde Curso)
protected void eliminarCurso(Curso curso) {
    cursos.remove(curso);
}

// Mostrar información del profesor y sus cursos
public void mostrarInfo() {
    System.out.println("Profesor: " + nombre + " (Legajo: " + legajo + ")");
    if (cursos.isEmpty()) {
        System.out.println("No dicta cursos actualmente.");
    } else {
        System.out.println("Cursos dictados:");
        for (Curso c : cursos) {
            System.out.println("- " + c.getNombre());
        }
    }
}

```

Clase Curso

```

public class Curso {
    private String nombre;

```

```
private String codigo;
private Profesor profesor;

public Curso(String nombre, String codigo, Profesor profesor) {
    this.nombre = nombre;
    this.codigo = codigo;
    this.profesor = null; // se asigna usando setProfesor
    setProfesor(profesor);
}

public String getNombre() { return nombre; }
public String getCodigo() { return codigo; }
public Profesor getProfesor() { return profesor; }

// Cambiar/Asignar profesor
public void setProfesor(Profesor nuevoProfesor) {
    if (this.profesor != null) {
        this.profesor.eliminarCurso(this); // quitar de la lista del profesor anterior
    }
    this.profesor = nuevoProfesor;
    if (nuevoProfesor != null) {
        nuevoProfesor.agregarCurso(this); // agregar a la lista del nuevo
profesor
    }
}

public void mostrarInfo() {
    System.out.println("Curso: " + nombre + " (Código: " + codigo + ")");
    if (profesor != null) {
```

```
        System.out.println("Profesor responsable: " + profesor.getNombre());
    } else {
        System.out.println("Sin profesor asignado.");
    }
}
```

Clase Universidad

```
import java.util.ArrayList;
```

```
public class Universidad {
    private ArrayList<Profesor> profesores;
    private ArrayList<Curso> cursos;

    public Universidad() {
        profesores = new ArrayList<>();
        cursos = new ArrayList<>();
    }
}
```

```
// Agregar profesor
```

```
public void agregarProfesor(Profesor profesor) {
    if (!profesores.contains(profesor)) {
        profesores.add(profesor);
    }
}
```

```
// Eliminar profesor (y remover de sus cursos)
```

```
public void eliminarProfesor(Profesor profesor) {
    if (profesores.contains(profesor)) {
        // Quitar profesor de todos sus cursos
    }
}
```

```
for (Curso c : new ArrayList<>(profesor.getCursos())) {
    c.setProfesor(null);
}
profesores.remove(profesor);
}

// Agregar curso
public void agregarCurso(Curso curso) {
    if (!cursos.contains(curso)) {
        cursos.add(curso);
    }
}

// Eliminar curso
public void eliminarCurso(Curso curso) {
    if (cursos.contains(curso)) {
        curso.setProfesor(null); // quitar referencia al profesor
        cursos.remove(curso);
    }
}

// Listar profesores
public void listarProfesores() {
    for (Profesor p : profesores) {
        p.mostrarInfo();
        System.out.println();
    }
}
```

```

// Listar cursos

public void listarCursos() {
    for (Curso c : cursos) {
        c.mostrarInfo();
        System.out.println();
    }
}

// Buscar profesor por legajo

public Profesor buscarProfesor(String legajo) {
    for (Profesor p : profesores) {
        if (p.getLegajo().equalsIgnoreCase(legajo)) return p;
    }
    return null;
}

// Buscar curso por código

public Curso buscarCurso(String codigo) {
    for (Curso c : cursos) {
        if (c.getCodigo().equalsIgnoreCase(codigo)) return c;
    }
    return null;
}

```

Clase Main

```

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

```

```
Universidad uni = new Universidad();  
  
// Crear profesores  
Profesor prof1 = new Profesor("Ana Pérez", "P001");  
Profesor prof2 = new Profesor("Luis Gómez", "P002");  
  
uni.agregarProfesor(prof1);  
uni.agregarProfesor(prof2);  
  
// Crear cursos  
Curso curso1 = new Curso("Programación II", "C001", prof1);  
Curso curso2 = new Curso("Bases de Datos", "C002", prof1);  
Curso curso3 = new Curso("Inteligencia Artificial", "C003", prof2);  
  
uni.agregarCurso(curso1);  
uni.agregarCurso(curso2);  
uni.agregarCurso(curso3);  
  
// Listar cursos y profesores  
System.out.println("==== Cursos ====");  
uni.listarCursos();  
  
System.out.println("==== Profesores ====");  
uni.listarProfesores();  
  
// Cambiar profesor de un curso  
System.out.println("==== Reasignar curso C002 a Luis Gómez ====");  
curso2.setProfesor(prof2);
```

```

        System.out.println("==> Cursos actualizados ==>");
        uni.listarCursos();

        System.out.println("==> Profesores actualizados ==>");
        uni.listarProfesores();
    }
}

```

1. Relación bidireccional Profesor–Curso:

- Curso conoce a su Profesor.
- Profesor mantiene una lista de sus Cursos.

2. Invariante de asociación:

- Cada vez que se asigna o cambia un profesor de un curso, se actualiza automáticamente la lista de cursos del profesor correspondiente-

3. Gestión desde Universidad:

- Alta, baja y consulta de profesores y cursos.

4. Uso de ArrayList:

- Para manejar colecciones dinámicas de profesores y cursos.

5. Modularidad y encapsulamiento:

- Métodos agregarCurso y eliminarCurso en Profesor son protected para controlar la relación desde Curso.

Clases a implementar

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
public class Profesor {
```

```
    private String id;
```

```
    private String nombre;
```

```
private String especialidad;
private List<Curso> cursos;

public Profesor(String id, String nombre, String especialidad) {
    this.id = id;
    this.nombre = nombre;
    this.especialidad = especialidad;
    this.cursos = new ArrayList<>();
}

public String getId() { return id; }
public String getNombre() { return nombre; }
public String getEspecialidad() { return especialidad; }
public List<Curso> getCursos() { return cursos; }

// Agregar curso y sincronizar la relación bidireccional
public void agregarCurso(Curso curso) {
    if (!cursos.contains(curso)) {
        cursos.add(curso);
        if (curso.getProfesor() != this) {
            curso.setProfesor(this); // sincroniza el lado del curso
        }
    }
}

// Eliminar curso y sincronizar la relación bidireccional
public void eliminarCurso(Curso curso) {
    if (cursos.contains(curso)) {
        cursos.remove(curso);
    }
}
```

```

        if (curso.getProfesor() == this) {
            curso.setProfesor(null); // dejar profesor en null
        }
    }

// Listar cursos con código y nombre
public void listarCursos() {
    if (cursos.isEmpty()) {
        System.out.println(nombre + " no dicta cursos actualmente.");
    } else {
        System.out.println("Cursos dictados por " + nombre + ":");

        for (Curso c : cursos) {
            System.out.println("- " + c.getCodigo() + ": " + c.getNombre());
        }
    }
}

// Mostrar información del profesor
public void mostrarInfo() {
    System.out.println("Profesor: " + nombre +
        " (ID: " + id + ", Especialidad: " + especial

```

Características implementadas

1. Bidireccionalidad con Curso:

- agregarCurso() y eliminarCurso() actualizan ambos lados de la relación.

2. Control de duplicados:

- Evita agregar el mismo curso varias veces.

3. Modularidad:

- Métodos separados para agregar, eliminar, listar y mostrar información.

4. Sincronización automática:

- Cada vez que un curso cambia de profesor, se actualiza la lista de cursos del profesor.

Clase Curso

```
public class Curso {  
  
    private String codigo;  
  
    private String nombre;  
  
    private Profesor profesor;  
  
  
    public Curso(String codigo, String nombre, Profesor profesor) {  
  
        this.codigo = codigo;  
  
        this.nombre = nombre;  
  
        this.profesor = null; // se asigna usando setProfesor  
  
        setProfesor(profesor); // asignar profesor inicial y sincronizar  
    }  
  
  
    public String getCodigo() { return codigo; }  
    public String getNombre() { return nombre; }  
    public Profesor getProfesor() { return profesor; }  
  
  
    // Asignar o cambiar profesor, sincronizando ambos lados  
    public void setProfesor(Profesor nuevoProfesor) {  
  
        // Si tenía profesor previo, quitarse de su lista  
        if (this.profesor != null && this.profesor != nuevoProfesor) {  
  
            this.profesor.eliminarCurso(this);  
        }  
  
        this.profesor = nuevoProfesor;  
        nuevoProfesor.agregarCurso(this);  
    }  
}
```

```

}

this.profesor = nuevoProfesor;

// Agregar curso a la lista del nuevo profesor
if (nuevoProfesor != null && !nuevoProfesor.getCurso().contains(this)) {
    nuevoProfesor.agregarCurso(this);
}

}

// Mostrar información del curso
public void mostrarInfo() {
    System.out.print("Curso: " + nombre + " (Código: " + codigo + ")");
    if (profesor != null) {
        System.out.println(", Profesor: " + profesor.getNombre());
    } else {
        System.out.println(", Profesor: Sin asignar");
    }
}

```

Clase Universidad

```

import java.util.ArrayList;
import java.util.List;

public class Universidad {
    private String nombre;
    private List<Profesor> profesores;
    private List<Curso> cursos;

```

```
public Universidad(String nombre) {  
    this.nombre = nombre;  
    this.profesores = new ArrayList<>();  
    this.cursos = new ArrayList<>();  
}  
  
// Agregar profesor  
public void agregarProfesor(Profesor p) {  
    if (!profesores.contains(p)) {  
        profesores.add(p);  
    }  
}  
  
// Agregar curso  
public void agregarCurso(Curso c) {  
    if (!cursos.contains(c)) {  
        cursos.add(c);  
    }  
}  
  
// Asignar profesor a un curso  
public void asignarProfesorACurso(String codigoCurso, String idProfesor) {  
    Curso curso = buscarCursoPorCodigo(codigoCurso);  
    Profesor profesor = buscarProfesorPorId(idProfesor);  
    if (curso != null && profesor != null) {  
        curso.setProfesor(profesor); // sincroniza ambos lados automáticamente  
    }  
}
```

```
// Listar profesores

public void listarProfesores() {
    System.out.println("==> Profesores de " + nombre + " ==>");
    for (Profesor p : profesores) {
        p.mostrarInfo();
        System.out.println();
    }
}

// Listar cursos

public void listarCursos() {
    System.out.println("==> Cursos de " + nombre + " ==>");
    for (Curso c : cursos) {
        c.mostrarInfo();
    }
}

// Buscar profesor por ID

public Profesor buscarProfesorPorId(String id) {
    for (Profesor p : profesores) {
        if (p.getId().equalsIgnoreCase(id)) {
            return p;
        }
    }
    return null;
}

// Buscar curso por código
```

```
public Curso buscarCursoPorCodigo(String codigo) {  
    for (Curso c : cursos) {  
        if (c.getCodigo().equalsIgnoreCase(codigo)) {  
            return c;  
        }  
    }  
    return null;  
}  
  
// Eliminar curso  
public void eliminarCurso(String codigo) {  
    Curso curso = buscarCursoPorCodigo(codigo);  
    if (curso != null) {  
        curso.setProfesor(null); // romper relación con profesor  
        cursos.remove(curso);  
    }  
}  
  
// Eliminar profesor  
public void eliminarProfesor(String id) {  
    Profesor profesor = buscarProfesorPorId(id);  
    if (profesor != null) {  
        // Dejar null los cursos que dictaba  
        for (Curso c : new ArrayList<>(profesor.getCursoes())) {  
            c.setProfesor(null);  
        }  
        profesores.remove(profesor);  
    }  
}
```

}