PROGRAMACION II.
Modulo 6: Colecciones y Sistema de Stock.

Trabajo práctico 6: Colecciones y sistema de stock.

Alumno: LEPKA AGUSTIN
Comisión: 13

OBJETIVO GENERAL

Desarrollar estructuras de datos dinámicas en Java mediante el uso de colecciones (ArrayList) y enumeraciones (enum), implementando un sistema de stock con funcionalidades progresivas que refuerzan conceptos clave de la programación orientada a objetos.

MARCO TEORICO

Concepto	Aplicación en el proyecto
ArrayList	Estructura principal para almacenar productos en el inventario.
Enumeraciones (enum)	Representan las categorías de productos con valores predefinidos.
Relaciones 1 a N	Relación entre Inventario (1) y múltiples Productos (N).
Métodos en enum	Inclusión de descripciones dentro del enum para mejorar legibilidad.
Ciclo for-each	Recorre colecciones de productos para listado, búsqueda o filtrado.
Búsqueda y filtrado	Por ID y por categoría, aplicando condiciones.
Ordenamientos y reportes	Permiten organizar la información y mostrar estadísticas útiles.
Encapsulamiento	Restringir el acceso directo a los atributos de una clase

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.

2. Clases a implementar Clase Producto

Atributos:

- **id (String)** → Identificador único del producto.
- **nombre (String)** → Nombre del producto.
- **precio (double)** → Precio del producto.
- **cantidad (int)** → Cantidad en stock.
- **categoria (CategoriaProducto)** → Categoría del producto.

Métodos:

- **mostrarInfo()** → Muestra en consola la información del producto.

Enum CategoriaProducto

Valores:
- ALIMENTOS
- ELECTRONICA
- ROPA
- HOGAR

Método adicional:

java **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;
}
}

Clase Inventario
Atributo:
● **ArrayList<Producto> productos** Métodos requeridos:
● **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. Tareas a realizar
1. Crear al menos cinco productos con diferentes categorías y agregarlos al inventario.
2. Listar todos los productos mostrando su información y categoría.
3. Buscar un producto por ID y mostrar su información.
4. Filtrar y mostrar productos que pertenezcan a una categoría específica.
5. Eliminar un producto por su ID y listar los productos restantes.
6. Actualizar el stock de un producto existente.
7. Mostrar el total de stock disponible.
8. Obtener y mostrar el producto con mayor stock.
9. Filtrar productos con precios entre $1000 y $3000.
10. Mostrar las categorías disponibles con sus descripciones.

Clase Main:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

public class TP6 {

    public static void main(String[] args) {

        //Ejercicio 1: sistema de stock

        System.out.println("Resultados Ejercicio 1:");

        //se crea el inventario vacio
        Inventario inventario = new Inventario();

        //se crean los productos
        Producto p1 = new Producto("P001", "Carne", 1200, 50,
CategoriaProducto.ALIMENTOS);
        Producto p2 = new Producto("P002", "Computadora", 2500, 20,
CategoriaProducto.ELECTRONICA);
        Producto p3 = new Producto("P003", "Pantalon", 1500, 100,
CategoriaProducto.ROPA);
```

```
        Producto p4 = new Producto("P004", "Microondas", 2000, 30,
CategoriaProducto.HOGAR);
        Producto p5 = new Producto("P005", "Celular", 3000, 15,
CategoriaProducto.ELECTRONICA);

        //se agregan productos al inventario
        inventario.agregarProducto(p1);
        inventario.agregarProducto(p2);
        inventario.agregarProducto(p3);
        inventario.agregarProducto(p4);
        inventario.agregarProducto(p5);

        System.out.println("Lista completa de productos:");
        inventario.listarProductos();

        //busqueda por ID
        System.out.println("Buscar producto por ID 'P003':");
        Producto buscado = inventario.buscarProductoPorId("P003");
        if (buscado != null) buscado.mostrarInfo();

        //filtrado por categoria
        System.out.println("Filtrar productos por categoria
ELECTRONICA:");
        inventario.filtrarPorCategoria(CategoriaProducto.ELECTRONICA);

        //eliminar un producto y luego listar
        System.out.println("Eliminar producto 'P001' y listar
nuevamente:");
        inventario.eliminarProducto("P001");
        inventario.listarProductos();

        //actualizar cantidad de stock de un producto
        System.out.println("Actualizar stock de 'P002' a 35
unidades:");
        inventario.actualizarStock("P002", 35);
        inventario.listarProductos();

        //mostrar total de stock de todos los productos
        System.out.println("Total de stock disponible: " +
inventario.obtenerTotalStock());

        //mostrar el producto con mayor cantidad de stock
        System.out.println("Producto con mayor stock:");
        Producto mayorStock =
inventario.obtenerProductoConMayorStock();
        if (mayorStock != null) mayorStock.mostrarInfo();

        //filtrar por rango de precio
        System.out.println("Filtrar productos con precio entre $1000 y
$3000:");
        inventario.filtrarProductosPorPrecio(1000, 3000);

        //mostrar categorias del inventario
        System.out.println("Categorías disponibles:");
        inventario.mostrarCategoriasDisponibles();

        //Ejercicio 2: Biblioteca y libros

        System.out.println("Resultados ejercicio 2:");

        //se crea la biblioteca
```

```
        Biblioteca biblioteca = new Biblioteca("Biblioteca Central");

        //se crean los autores
        Autor a1 = new Autor("A001", "Gabriel Garcia Marquez",
"Colombiana");
        Autor a2 = new Autor("A002", "J.K. Rowling", "Británica");
        Autor a3 = new Autor("A003", "George Orwell", "Británica");

        //se agregan libros
        biblioteca.agregarLibro("L001", "Cien Años de Soledad", 1967,
a1);
        biblioteca.agregarLibro("L002", "Harry Potter y la Piedra
Filosofal", 1997, a2);
        biblioteca.agregarLibro("L003", "Harry Potter y la Cámara
Secreta", 1998, a2);
        biblioteca.agregarLibro("L004", "1984", 1949, a3);
        biblioteca.agregarLibro("L005", "Animal Farm", 1945, a3);

        //se lista los libros de la biblioteca
        System.out.println("Lista completa de libros:");
        biblioteca.listarLibros();

        //se busca libro por ISBN
        System.out.println("Buscar libro por ISBN 'L002':");
        Libro libroBuscado = biblioteca.buscarLibroPorIsbn("L002");
        if (libroBuscado != null) libroBuscado.mostrarInfo();

        //filtrado por año
        System.out.println("Filtrar libros publicados en 1998:");
        biblioteca.filtrarLibrosPorAnio(1998);

        //eliminar un libro y luego listar
        System.out.println("Eliminar libro 'L005' y listar
nuevamente:");
        biblioteca.eliminarLibro("L005");
        biblioteca.listarLibros();

        //mostrar cantidad de libros
        System.out.println("Cantidad total de libros: " +
biblioteca.obtenerCantidadLibros());

        //mostrar autores disponibles
        System.out.println("Autores disponibles:");
        biblioteca.mostrarAutoresDisponibles();

        //Ejercicio 3: Universidad, profesor y curso

        System.out.println("Resultados ejercicio 3:");

        //se crea la universidad
        Universidad utn = new Universidad("UTN");

        //se crean profesores
        Profesor prof1 = new Profesor("P001", "Juan Perez",
"Programacion");
        Profesor prof2 = new Profesor("P002", "Fulano Lopez",
"Matematica");
        Profesor prof3 = new Profesor("P003", "Agustin Martinez",
"Fisica");

        //se agregan profesores
```

```
        utn.agregarProfesor(prof1);
        utn.agregarProfesor(prof2);
        utn.agregarProfesor(prof3);

        //se crean los cursos
        Curso c1 = new Curso("C001", "Programación I");
        Curso c2 = new Curso("C002", "Matemática Discreta");
        Curso c3 = new Curso("C003", "Física I");
        Curso c4 = new Curso("C004", "Algoritmos");
        Curso c5 = new Curso("C005", "Cálculo");

        //se agregan los cursos
        utn.agregarCurso(c1);
        utn.agregarCurso(c2);
        utn.agregarCurso(c3);
        utn.agregarCurso(c4);
        utn.agregarCurso(c5);

        //se asignan profesores a los cursos
        System.out.println("Asignar profesores a cursos:");
        utn.asignarProfesorACurso("C001", "P001");
        utn.asignarProfesorACurso("C004", "P001");
        utn.asignarProfesorACurso("C002", "P002");
        utn.asignarProfesorACurso("C005", "P002");
        utn.asignarProfesorACurso("C003", "P003");

        //se listan todos los cursos
        System.out.println("Listado de cursos:");
        utn.listarCursos();

        //se listan los profesores
        System.out.println("Listado de profesores:");
        utn.listarProfesores();

        //se cambia el profesor de un curso
        System.out.println("Cambiar profesor de C004 a Ana Gomez:");
        utn.asignarProfesorACurso("C004", "P002");

        //tras el cambio se lista de nuevo los cursos y profesores
        System.out.println("Listado de cursos tras cambio:");
        utn.listarCursos();
        System.out.println("Listado de profesores tras cambio:");
        utn.listarProfesores();

        //se elimina un curso y se muestra lo actualizado
        System.out.println("Eliminar curso C005:");
        utn.eliminarCurso("C005");
        utn.listarCursos();
        utn.listarProfesores();

        //se elimina un profesor y se muestra el listado nuevo
        System.out.println("Eliminar profesor P003:");
        utn.eliminarProfesor("P003");
        utn.listarCursos();
        utn.listarProfesores();
    }

}
```

Clase CategoriaProducto:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/

package tp6;

public enum CategoriaProducto {

    //constantes del enum con su descripcion
    ALIMENTOS("Productos comestibles"),
    ELECTRONICA("Dispositivos electrónicos"),
    ROPA("Prendas de vestir"),
    HOGAR("Artículos para el hogar");

    //atributo privado que guarda la descripcion
    private final String descripcion;

    //constructor del enum
    CategoriaProducto(String descripcion) {
        this.descripcion = descripcion;
    }

    //getter
    public String getDescripcion() {
        return descripcion;
    }
}
```

Clase Producto:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

public class Producto {

    //atributos
    private String id;
    private String nombre;
    private double precio;
    private int cantidad;
    private CategoriaProducto categoria;

    //constructor
    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;
```

```
    }

    //getter
    public String getId() { return id; }
    public int getCantidad() { return cantidad; }
    public double getPrecio() { return precio; }
    public CategoriaProducto getCategoria() { return categoria; }

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


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

    //representar el objeto como texto
    @Override
    public String toString() {
        return nombre + " [" + id + "]";
    }
}
```

Clase Inventario:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

import java.util.ArrayList;

public class Inventario {
    //se define el array
    private ArrayList<Producto> productos;

    //constructor que inicializa el arraylist
    public Inventario() {
        productos = new ArrayList<>();
    }

    //manipulacion de lista
    public void agregarProducto(Producto p) {
        productos.add(p);
    }

    public void listarProductos() {
        for (Producto p : productos) {
            p.mostrarInfo();
        }
    }

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

```
        }
        return null;
    }

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

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

    public void filtrarPorCategoria(CategoriaProducto 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 max = productos.get(0);
        for (Producto p : productos) {
            if (p.getCantidad() > max.getCantidad()) max = p;
        }
        return max;
    }

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

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

Nuevo Ejercicio Propuesto 2: Biblioteca y Libros

1. Descripción general

Se debe desarrollar un sistema para gestionar una **biblioteca**, en la cual se registren los libros disponibles y sus autores. La relación central es de **composición 1 a N**: una Biblioteca contiene múltiples Libros, y cada Libro pertenece obligatoriamente a una Biblioteca. Si la Biblioteca se elimina, también se eliminan sus Libros.

2. Clases a implementar

Clase Autor
Atributos:
• **id (String)** → Identificador único del autor.
• **nombre (String)** → Nombre del autor.
• **nacionalidad (String)** → Nacionalidad del autor.

Métodos:
• **mostrarInfo()** → Muestra la información del autor en consola.

Clase Libro
Atributos:
• **isbn (String)** → Identificador único del libro.
• **titulo (String)** → Título del libro.
• **anioPublicacion (int)** → Año de publicación.
• **autor (Autor)** → Autor del libro.

Métodos:
• **mostrarInfo()** → Muestra título, ISBN, año y autor.

Clase Biblioteca
Atributo:
• String nombre
• **List<Libro> libros** → Colección de libros de la biblioteca.

Métodos requeridos:

• **agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor)**
• listarLibros()
• buscarLibroPorIsbn(String isbn)
• eliminarLibro(String isbn)
• obtenerCantidadLibros()
• filtrarLibrosPorAnio(int anio)
• mostrarAutoresDisponibles()

3. Tareas a realizar

1. Creamos una biblioteca.
2. Crear al menos tres autores
3. Agregar 5 libros asociados a alguno de los Autores a la biblioteca.
4. Listar todos los libros con su información y la del autor.
5. Buscar un libro por su ISBN y mostrar su información.
6. Filtrar y mostrar los libros publicados en un año específico.
7. Eliminar un libro por su ISBN y listar los libros restantes.
8. Mostrar la cantidad total de libros en la biblioteca.
9. Listar todos los autores de los libros disponibles en la biblioteca.

Clase Biblioteca:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

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

public class Biblioteca {

    private String nombre;
    private List<Libro> libros;

    //constructor que inicia el arraylist
    public Biblioteca(String nombre) {
        this.nombre = nombre;
        libros = new ArrayList<>();
    }

    //agregar libro a la biblioteca
    public void agregarLibro(String isbn, String titulo, int
anioPublicacion, Autor autor) {
        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
    }

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

    //buscar libro por ISBN
    public Libro buscarLibroPorIsbn(String isbn) {
        for (Libro l : libros) {
            if (l.getIsbn().equals(isbn)) return l;
        }
        return null;
    }

    //eliminar libro por ISBN
    public void eliminarLibro(String isbn) {
        libros.removeIf(l -> l.getIsbn().equals(isbn));
    }

    //cantidad de libros de la biblioteca
    public int obtenerCantidadLibros() {
        return libros.size();
    }

    //filtra y muestra libros de determinado año
    public void filtrarLibrosPorAnio(int anio) {
        for (Libro l : libros) {
            if (l.getAnioPublicacion() == anio) l.mostrarInfo();
        }
    }

    //muestra los autores disponibles en biblioteca
```

```
    public void mostrarAutoresDisponibles() {
        List<String> autores = new ArrayList<>();
        for (Libro l : libros) {
            if (!autores.contains(l.getAutor().getNombre()))
autores.add(l.getAutor().getNombre());
        }
        for (String nombre : autores) System.out.println(nombre);
    }
}
```

Clase Autor:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

public class Autor {
    //atributos
    private String id;
    private String nombre;
    private String nacionalidad;

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

    public void mostrarInfo() {
        System.out.println("ID: " + id + ", Nombre: " + nombre + ",
Nacionalidad: " + nacionalidad);
    }

    //getter
    public String getNombre() {
        return nombre;
    }
}
```

Clase Libro:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

public class Libro {
    //atributos
    private String isbn;
    private String titulo;
    private int anioPublicacion;
    private Autor autor;
```

```
    //constructor
    public Libro(String isbn, String titulo, int anioPublicacion,
Autor autor) {
        this.isbn = isbn;
        this.titulo = titulo;
        this.anioPublicacion = anioPublicacion;
        this.autor = autor;
    }

    //getters
    public String getIsbn() { return isbn; }
    public int getAnioPublicacion() { return anioPublicacion; }
    public Autor getAutor() { return autor; }

    //muestra la informacion del libro
    public void mostrarInfo() {
        System.out.println("Título: " + titulo + ", ISBN: " + isbn +
", Año: " + anioPublicacion + ", Autor: " + autor.getNombre());
    }
}
```

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

1. Descripción general

Se debe modelar un sistema académico donde **un Profesor dicta muchos Cursos** y cada **Curso** tiene exactamente **un Profesor responsable**. La relación **Profesor–Curso** es **bidireccional**:
• Desde **Curso** se accede a su **Profesor**.
• Desde **Profesor** se accede a la **lista de Cursos** que dicta. Además, existe la clase **Universidad** que administra el alta/baja y consulta de profesores y cursos.

2. Clases a implementar

Invariante de asociación: cada vez que se asigne o cambie el profesor de un curso, **debe actualizarse en los dos lados** (agregar/quitar en la lista del profesor correspondiente).

Clase Profesor

Atributos:
• **id (String)** → Identificador único.
• **nombre (String)** → Nombre completo.
• **especialidad (String)** → Área principal.
• **List<Curso> cursos** → Cursos que dicta.

Métodos sugeridos:
• **agregarCurso(Curso c)** → Agrega el curso a su lista si no está y sincroniza el lado del curso.

• **eliminarCurso(Curso c)** → Quita el curso y sincroniza el lado del curso (dejar profesor en null si corresponde).
• **listarCursos()** → Muestra códigos y nombres.
• **mostrarInfo()** → Imprime datos del profesor y cantidad de cursos.

Clase Curso

Atributos:
• **codigo (String)** → Código único.
• **nombre (String)** → Nombre del curso.
• **profesor (Profesor)** → Profesor responsable.

Métodos sugeridos:

• **setProfesor(Profesor p)** → Asigna/cambia el profesor **sincronizando ambos lados**: o Si tenía profesor previo, quitarse de su lista.

• **mostrarInfo()** → Muestra código, nombre y nombre del profesor (si tiene).

Clase Universidad

Atributos:
• String nombre
• List<Profesor> profesores
• List<Curso> cursos

Métodos requeridos:
• agregarProfesor(Profesor p)
• agregarCurso(Curso c)
• **asignarProfesorACurso(String codigoCurso, String idProfesor)** → Usa setProfesor del curso.
• **listarProfesores() / listarCursos()**
• **buscarProfesorPorId(String id)**
• buscarCursoPorCodigo(String codigo)
• **eliminarCurso(String codigo)** → Debe **romper la relación** con su profesor si la hubiera.
• **eliminarProfesor(String id)** → Antes de remover, dejar null los cursos que dictaba.

Tareas a realizar

1. Crear **al menos 3 profesores** y **5 cursos**.
2. Agregar profesores y cursos a la universidad.
3. Asignar profesores a cursos usando asignarProfesorACurso(...).
4. Listar cursos con su profesor y profesores con sus cursos.
5. Cambiar el profesor de un curso y verificar que ambos lados quedan sincronizados.
6. Remover un curso y confirmar que ya **no** aparece en la lista del profesor.
7. Remover un profesor y dejar profesor = null,
8. Mostrar un reporte: cantidad de cursos por profesor.

Clase Profesor:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

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

public class Profesor {

    //atributos
    private String id;
    private String nombre;
    private String especialidad;
    private List<Curso> cursos;

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

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

    //agrega un curso a la lista
    public void agregarCurso(Curso c) {
        if (!cursos.contains(c)) {
            cursos.add(c);
            c.setProfesor(this);
        }
    }


    //elimina un curso de la lista
    public void eliminarCurso(Curso c) {
        if (cursos.contains(c)) {
            cursos.remove(c);
            c.setProfesor(null);
        }
    }

    //muestra los cursos dictados por el profesor
    public void listarCursos() {
        for (Curso c : cursos) {
            System.out.println(c.getCodigo() + " = " + c.getNombre());
        }
    }

    //muestra la informacion del profesor
    public void mostrarInfo() {
```

```
        System.out.println("ID: " + id + ", Nombre: " + nombre + ",
Especialidad: " + especialidad + ", Cursos dictados: " +
cursos.size());
    }


}
```

Clase Curso:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

public class Curso {

    private String codigo;
    private String nombre;
    private Profesor profesor;

    //constructor
    public Curso(String codigo, String nombre) {
        this.codigo = codigo;
        this.nombre = nombre;
    }

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

    //setter
    public void setProfesor(Profesor p) {
        if (this.profesor != null)
this.profesor.getCursos().remove(this);
        this.profesor = p;
        if (p != null && !p.getCursos().contains(this))
p.getCursos().add(this);
    }

    //muestra la informacion del curso
    public void mostrarInfo() {
        String nombreProfesor = (profesor != null) ?
profesor.getNombre() : "Sin profesor";
        System.out.println("Código: " + codigo + ", Nombre: " + nombre
+ ", Profesor: " + nombreProfesor);
    }
}
```

Clase Universidad:

```
/*
PROGRAMACION II - TRABAJO PRACTICO 6

ALUMNO: LEPKA AGUSTIN MARIO NICOLAS
COMISION: 13
*/
package tp6;

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

public class Universidad {

    //atributos
    private String nombre;
    private List<Profesor> profesores;
    private List<Curso> cursos;

    //constructor
    public Universidad(String nombre) {
        this.nombre = nombre;
        profesores = new ArrayList<>();
        cursos = new ArrayList<>();
    }

    //getter
    public String getNombre() {
    return nombre;}

    //agregar profesor
    public void agregarProfesor(Profesor p) { profesores.add(p); }
    //agregar curso
    public void agregarCurso(Curso c) { cursos.add(c); }

    //asignar profesor a un curso
    public void asignarProfesorACurso(String codigoCurso, String
idProfesor) {
        Curso curso = buscarCursoPorCodigo(codigoCurso);
        Profesor prof = buscarProfesorPorId(idProfesor);
        if (curso != null && prof != null) curso.setProfesor(prof);
    }

    //listados

    public void listarProfesores() {
        for (Profesor p : profesores) p.mostrarInfo();
    }

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

    //busquedas

    //profesor por ID

    public Profesor buscarProfesorPorId(String id) {
        for (Profesor p : profesores) {
```

```
            if (p.getId().equals(id)) return p;
        }
        return null;
    }

    //curso por codigo
    public Curso buscarCursoPorCodigo(String codigo) {
        for (Curso c : cursos) {
            if (c.getCodigo().equals(codigo)) return c;
        }
        return null;
    }

    //remocion

    //eliminar curso
    public void eliminarCurso(String codigo) {
        Curso c = buscarCursoPorCodigo(codigo);
        if (c != null) {
            if (c.getProfesor() != null)
c.getProfesor().getCursos().remove(c);
            cursos.remove(c);
        }
    }

    //eliminar profresor
    public void eliminarProfesor(String id) {
    Profesor p = buscarProfesorPorId(id);
    if (p != null) {
        //se crea una copia de la lista para evitar
ConcurrentModificationException
        ArrayList<Curso> cursosCopia = new ArrayList<>(p.getCursos());
        for (Curso c : cursosCopia) {
            c.setProfesor(null);
        }
        profesores.remove(p);
    }
}
}
```