Programación II - TP6: Colecciones

Alumno: *Romero, Abel Tomás (Comisión 5)*
Link del repo de GitHub:
https://github.com/Tomu98/UTN-TUPaD-P2-TPs/tree/main/06%20Colecciones

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 Teórico:

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

```java
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;
    }
```

- **Método:** `mostrarInfo()` → Muestra en consola la info. del producto.

```java
public void mostrarInfo() {
    System.out.println("-> Producto(ID=" + id +
            ", nombre=" + nombre +
            ", precio=" + precio +
            ", cantidad=" + cantidad +
            ", categoria=" + categoria +
            ")");
}
```

- **Enum `CategoriaProducto`:** 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;
        }
} ```

```java
public enum CategoriaProducto {

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

    // Atributo
    private final String descripcion;

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

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

- **Clase Inventario**
  - **Atributo:** `ArrayList<Producto> productos`

```java
public class Inventario {

    // Atributo
    private ArrayList<Producto> productos;

    // Constructor
    public Inventario() {
        this.productos = new ArrayList();
    }
```

  - **Métodos requeridos:**
    - `agregarProducto(Producto p)`
    - `listarProductos()`
    - `buscarProductosPorId(String id)`
    - `eliminarProducto(String id)`
    - `actualizarStock(String id, int nuevaCantidad)`
    - `filtrarPorCategoria(CategoriaProducto categoria)`
    - `obtenerTotalStock()`
    - `obtenerProductoConMayorStock()`
    - `filtrarProductosPorPrecio(double min, double max)`
    - `mostrarCategoriasDisponibles()`

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

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

```java
public Producto buscarProductoPorId(String id) {
    Producto productoEncontrado = null;
    int i = 0;

    while (i < productos.size() && !this.productos.get(i).getId().equalsIgnoreCase(id)) {
        i++;
    }

    if (i < this.productos.size()) {
        productoEncontrado = this.productos.get(i);
    }

    return productoEncontrado;
}
```

```java
public void eliminarProducto(String id) {
    Producto p = buscarProductoPorId(id);

    if (p == null) {
        System.out.println("ID no encontrado.");
    } else {
        this.productos.remove(p);
        System.out.println("Producto con ID " + id + " eliminado.");
    }
}
```

```java
public void actualizarStock(String id, int nuevaCantidad) {
    Producto p = buscarProductoPorId(id);

    if (p == null) {
        System.out.println("ID no encontrado.");
    } else if (nuevaCantidad < 0) {
        System.out.println("ERROR: Cantidad inválida. Debe ingresar una cantidad positiva.");
    } else {
        int cantidadAnterior = p.getCantidad();
        p.setCantidad(nuevaCantidad);
        System.out.println("Stock actualizado correctamente.");
        System.out.println("- Stock anterior: " + cantidadAnterior);
        System.out.println("- Stock nuevo: " + p.getCantidad());
    }
```

```java
public ArrayList<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
    ArrayList<Producto> productosFiltrados = new ArrayList();

    for (Producto p : productos) {
        if (p.getCategoria() == categoria) {
            productosFiltrados.add(p);
        }
    }

    return productosFiltrados;
}
```

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

    return totalStock;
}
```

```java
public Producto productoConMayorStock() {
    int maxStock = -1;

    Producto productoConMayorStock = null;
    for (Producto p : productos) {
        if (p.getCantidad() >= maxStock) {
            maxStock = p.getCantidad();
            productoConMayorStock = p;
        }
    }

    return productoConMayorStock;
}
```

```java
public ArrayList<Producto> filtrarProductosPorPrecio(double min, double max) {
    ArrayList<Producto> productosFiltrados = new ArrayList();

    for (Producto p : productos) {
        if (p.getPrecio() >= min && p.getPrecio() <= max) {
            productosFiltrados.add(p);
        }
    }

    return productosFiltrados;
}
```

```java
public void mostrarCategoriasDisponibles() {
    System.out.println("Categorias disponbles:");
    for (CategoriaProducto c : CategoriaProducto.values()) {
        System.out.println("-> '" + c + "': " + c.getDescripcion());
    }
```

3. Tareas a realizar:

1. Crear al menos cinco productos con diferentes categorías y agregarlos al inventario.

```
Producto producto1 = new Producto("01", "Samsung Galaxy S25 Ultra", 2500000, 1, CategoriaProducto.ELECTRONICA);
Producto producto2 = new Producto("02", "Mesa circular", 120000, 3, CategoriaProducto.HOGAR);
Producto producto3 = new Producto("03", "Playstation 5 Pro", 1500000, 2, CategoriaProducto.ELECTRONICA);
Producto producto4 = new Producto("04", "1Kg de Asado", 15000, 6, CategoriaProducto.ALIMENTOS);
Producto producto5 = new Producto("05", "Camisa Lino blanca", 20000, 2, CategoriaProducto.ROPA);
inventario1.agregarProducto(producto1);
inventario1.agregarProducto(producto2);
inventario1.agregarProducto(producto3);
inventario1.agregarProducto(producto4);
inventario1.agregarProducto(producto5);
```

2. Listar todos los productos mostrando su información y categoría.

```
System.out.println("\n=============================================================");
System.out.println("2. Lista de todos los productos:");
System.out.println("=============================================================");
inventario1.listarProductos();
```

3. Buscar un producto por ID y mostrar su información.

```
System.out.println("\n=============================================================");
System.out.println("3. Buscando producto con ID 03:");
System.out.println("=============================================================");
Producto productoPorId = inventario1.buscarProductoPorId("03");
productoPorId.mostrarInfo();
```

4. Filtrar y mostrar productos que pertenezcan a una categoría específica.

```
System.out.println("\n=============================================================");
System.out.println("4. Filtrar y mostrar productos de la categoría 'ELECTRONICA':");
System.out.println("=============================================================");
ArrayList<Producto> productosDeElectronica = inventario1.filtrarPorCategoria(CategoriaProducto.ELECTRONICA);
for (Producto p : productosDeElectronica) {
    p.mostrarInfo();
}
```

5. Eliminar un producto por su ID y listar los productos restantes.

```
System.out.println("\n=============================================================");
System.out.println("5. Eliminamos el producto de ID 04:");
System.out.println("=============================================================");
inventario1.eliminarProducto("04");
System.out.println("\nInventario actualizado:");
inventario1.listarProductos();
```

6. Actualizar el stock de un producto existente.

```
System.out.println("\n=============================================================");
System.out.println("6. Actualizamos el stock de un producto:");
System.out.println("=============================================================");
inventario1.actualizarStock("01", 2);
```

7. Mostrar el total de stock disponible.

```
System.out.println("\n=============================================================");
System.out.println("7. Mostramos el total de stock disponible:");
System.out.println("=============================================================");
System.out.println("Stock total disponible: " + inventario1.obtenerTotalStock());
```

8. Obtener y mostrar el producto con mayor stock.

```
System.out.println("\n=============================================================");
System.out.println("8. Mostramos el producto con mayor cantidad en stock:");
System.out.println("=============================================================");
inventariol.productoConMayorStock().mostrarInfo();
```

9. Filtrar productos con precios entre $1000 y $3000.

```
System.out.println("\n=============================================================");
System.out.println("9. Filtramos los productos con precio entre $100000 y $2000000:");
System.out.println("=============================================================");
ArrayList<Producto> productosFiltradosPorPrecio = inventariol.filtrarProductosPorPrecio(100000, 2000000);
for (Producto p : productosFiltradosPorPrecio) {
    p.mostrarInfo();
}
```

10. Mostrar las categorías disponibles con sus descripciones.

```
System.out.println("\n=============================================================");
System.out.println("10. Mostramos las categorías disponibles con sus descripciones:");
System.out.println("=============================================================");
inventariol.mostrarCategoriasDisponibles();
```

Caso Práctico 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:**
       - `String id` → Identificador único del autor.
       - `String nombre` → Nombre del autor.
       - `String nacionalidad` → Nacionalidad del 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;
}
```

     - **Método:** `mostrarInfo()` → Muestra la información del autor..`

```
public String mostrarInfo() {
    return "-> Autor(id=" + id +
           ", nombre=" + nombre +
           ", nacionalidad=" +
           nacionalidad + ")";
}
```

- **Clase Libro:**
  - **Atributos:**
    - `String isbn` → Identificador único del libro.
    - `String titulo` → Titulo del libro.
    - `int anioPublicacion` → Año de publicación.
    - `Autor autor` → Autor del 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;
}
```

  - **Método:** `mostrarInfo()` → Muestra título, ISBN, año y autor.`

```
public void mostrarInfo() {
    System.out.println("-> Libro(ISBN=" + isbn +
            ", titulo=" + titulo +
            ", anioPublicacion=" + anioPublicacion +
            ", autor=" + autor.mostrarInfo() + ")");
}
```

- **Clase Biblioteca:**
  - **Atributos:**
    - `String nombre`
    - `List<Libro> libros` → Colección de libros de la biblioteca.

```
// Atributos
private String nombre;
private List<Libro> libros;

// Constructor
public Biblioteca(String nombre) {
    this.nombre = nombre;
    this.libros = new ArrayList<>();
}
```

  - **Métodos:**
    - `agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor)`
    - `listarLibros()`
    - `buscarLibroPorIsbn(String isbn)`
    - `eliminarLibro(String isbn)`
    - `obtenerCantidadLibros()`
    - `filtrarLibrosPorAnio(int anio)`
    - `mostrarAutoresDisponibles()`

```java
public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
    if (!isbn.equals("") && !titulo.equals("") && anioPublicacion > 0 && autor != null) {
        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
    }
}
```

```java
public void listarLibros() {
    System.out.println("Libros de la biblioteca '" + nombre + "':");
    for (Libro libro : libros) {
        libro.mostrarInfo();
    }
}
```

```java
public Libro buscarLibroPorIsbn(String isbn) {
    for (Libro libro : libros) {
        if (libro.getIsbn().equals(isbn)) {
            return libro;
        }
    }

    return null;
}
```

```java
public void eliminarLibro(String isbn) {
    if (isbn != null) {
        Libro libroEliminar = this.buscarLibroPorIsbn(isbn);
        if (libroEliminar != null) {
            libros.remove(libroEliminar);
            System.out.println("Libro con ISBN=" + isbn + " eliminado.");
        }
    } else {
        System.out.println("ISBN no encontrado.");
    }
}
```

```java
public int obtenerCantidadLibros() {
    return libros.size();
}
```

```java
public List<Libro> filtrarLibrosPorAnio(int anio) {
    List<Libro> libroPorAnio = new ArrayList();
    for(Libro libro : libros) {
        if (libro.getAnioPublicacion() == anio) {
            libroPorAnio.add(libro);
        }
    }

    return Collections.unmodifiableList(libroPorAnio);
}
```

```java
public void mostrarAutoresDisponibles() {
    for(Libro libro : libros) {
        System.out.println(libro.getAutor().mostrarInfo());
    }
}
```

```java
public String getNombre() {
    return nombre;
}
```

3. Tareas a realizar:

1. Creamos una biblioteca.

```java
System.out.println("\n============================================================");
System.out.println("1. Creamos una biblioteca:");
System.out.println("============================================================");
Biblioteca bibliotecal = new Biblioteca("Biblioteca Turing");
System.out.println("Nombre de la biblioteca creaada: " + bibliotecal.getNombre());
```

2. Crear al menos tres autores

```
System.out.println("\n============================================================");
System.out.println("2. Creamos tres autores:");
System.out.println("============================================================");
Autor knuth = new Autor("01", "Donald Knuth", "Estados Unidos");
Autor kernighan = new Autor("02", "Brian Kernighan", "Canadá");
Autor martin = new Autor("03", "Robert C. Martin", "Estados Unidos");
System.out.println("Los autores creados son:");
System.out.println(knuth.mostrarInfo());
System.out.println(kernighan.mostrarInfo());
System.out.println(martin.mostrarInfo());
```

3. Agregar 5 libros asociados a alguno de los Autores a la biblioteca.

```
bibliotecal.agregarLibro("K123", "The Art of Computer Programming", 1968, knuth);
bibliotecal.agregarLibro("K456", "The C Programming Language", 1978, kernighan);
bibliotecal.agregarLibro("M789", "Clean Code", 2008, martin);
bibliotecal.agregarLibro("M234", "Clean Architecture", 2017, martin);
bibliotecal.agregarLibro("K789", "The Practice of Programming", 1999, kernighan);
```

4. Listar todos los libros con su información y la del autor.

```
System.out.println("\n============================================================");
System.out.println("4. Listamos todos los libros con su información:");
System.out.println("============================================================");
bibliotecal.listarLibros();
```

5. Buscar un libro por su ISBN y mostrar su información.

```
System.out.println("\n============================================================");
System.out.println("5. Buscamos un libro por su ISBN y mostramos su información:");
System.out.println("============================================================");
System.out.println("Buscamos el libro con ISBN: M789");
bibliotecal.buscarLibroPorIsbn("M789").mostrarInfo();
```

6. Filtrar y mostrar los libros publicados en un año específico.

```
System.out.println("\n============================================================");
System.out.println("6. Filtramos y mostramos los libros publicados en un año específico:");
System.out.println("============================================================");
int filtro = 2008;
List<Libro> librosPorAnio = bibliotecal.filtrarLibrosPorAnio(filtro);
System.out.println("Libros encontrados del año " + filtro + ":");
for (Libro libro : librosPorAnio) {
    libro.mostrarInfo();
}
```

7. Eliminar un libro por su ISBN y listar los libros restantes.

```
System.out.println("\n============================================================");
System.out.println("7. Eliminamos un libro por su ISBN:");
System.out.println("============================================================");
bibliotecal.eliminarLibro("K789");
```

8. Mostrar la cantidad total de libros en la biblioteca.

```
System.out.println("\n============================================================");
System.out.println("8. Listamos los libros de la biblioteca luego de eliminar:");
System.out.println("============================================================");
bibliotecal.listarLibros();
```

9. Listar todos los autores de los libros disponibles en la biblioteca.

```
System.out.println("\n============================================================");
System.out.println("9. Listamos los autores de los libros de la biblioteca:");
System.out.println("============================================================");
bibliotecal.mostrarAutoresDisponibles();
```

Caso Práctico 3: 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.

   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).

2. **Clases a implementar:**
   - **Clase Profesor:**
     - **Atributos:**
       - `String id` → Identificador único.
       - `String nombre` → Nombre completo.
       - `String especialidad` → Área principal.
       - `List<Curso> cursos` → Cursos que dicta.

```
// 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;
    this.cursos = new ArrayList<>();
}
```

     - **Métodos:**
       - `agregarCurso(Curso c)` → Agrega el curso a su lista si no está y sincroniza.
       - `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.

```
public void agregarCurso(Curso curso) {
    if (curso != null && !cursos.contains(curso)) {
        cursos.add(curso);
        if (curso.getProfesor() != this) {
            curso.setProfesor(this);
        }
    }
}
```

```java
public void eliminarCurso(Curso curso) {
    if (curso != null && cursos.contains(curso)) {
        cursos.remove(curso);

        if (curso.getProfesor() == this) {
            curso.setProfesor(null);
        }
    }
}
```

```java
public void listarCursos() {
    System.out.println("Los cursos del profesor " + nombre + " son: ");
    for (Curso curso : cursos) {
        curso.mostrarInfo();
    }
}
```

```java
public void mostrarInfo() {
    System.out.println("-> Profesor(ID=" + id +
            ", nombre=" + nombre +
            ", especialidad=" + especialidad +
            ", cursos=" + cursos.size() + ")");
}
```

- **Clase Curso:**
  - **Atributos:**
    - `String codigo` → Código único.
    - `String nombre` → Nombre del curso.
    - `Profesor profesor` → Profesor responsable.

2.

```java
// Atributos
private String codigo;
private String nombre;
private Profesor profesor;

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

  - **Métodos:**
    - `setProfesor(Profesor p)` → Asigna/cambia el profesor sincronizando ambos lados: Si tenía profesor previo, quitarse de su lista.
    - `mostrarInfo()` → Muestra código, nombre y nombre del profesor (si tiene).

```java
public void setProfesor(Profesor profesor) {
    // Mismo profesor ya asignado -> no se hace nada
    if (profesor == this.profesor) {
        return;
    }

    // Si el profesor agregado es distinto de null
    if (this.profesor != null) {
        this.profesor.eliminarCurso(this);
    }

    // Asignamos al nuevo profesor
    this.profesor = profesor;

    // Si el nuevo profesor es distinto de null y no se agregó a este curso, lo agregamos
    if (profesor != null && !profesor.getCursos().contains(this)) {
        profesor.agregarCurso(this);
    }
}
```

```java
public void mostrarInfo() {
    String nombreProfesor = (profesor != null) ? profesor.getNombre() : "Sin profesor";
    System.out.println("-> Curso(codigo=" + codigo +
            ", nombre=" + nombre +
            ", profesor=" + nombreProfesor + ")");
}
```

- **Clase Universidad:**
  - **Atributos:**
    - `String nombre`
    - `List<Profesor> profesores`
    - `List<Curso> cursos`

```java
// Atributos
private String nombre;
private List<Curso> cursos;
private List<Profesor> profesores;

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

  - **Métodos:**
    - `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.

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

```java
public void agregarCurso(Curso curso) {
    if (curso != null && !cursos.contains(curso)) {
        cursos.add(curso);
    }
}
```

```java
public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
    Profesor profesor = this.buscarProfesorPorId(idProfesor);
    Curso curso = this.buscarCursoPorCodigo(codigoCurso);

    if (profesor != null && curso != null) {
        curso.setProfesor(profesor);
    } else {
        System.out.println("ERROR: Docente o curso no existen.");
    }
}
```

```java
public void listarProfesor() {
    System.out.println("Cantidad de profesores en la universidad " + nombre + ": " + profesores.size());

    for (Profesor profesor : profesores) {
        profesor.mostrarInfo();
    }
}
```

```java
public void listarCursos() {
    System.out.println("Cantidad de cursos en la universidad " + nombre + ": " + cursos.size());

    for (Curso curso : cursos) {
        curso.mostrarInfo();
    }
}
```

```java
public Profesor buscarProfesorPorId(String id) {
    if (id != null) {
        for (Profesor profesor : profesores) {
            if (profesor.getId().equals(id)) {
                return profesor;
            }
        }
    }

    return null;
}
```

```java
public Curso buscarCursoPorCodigo(String codigo) {
    if (codigo != null) {
        for (Curso curso : cursos) {
            if (curso.getCodigo().equals(codigo)) {
                return curso;
            }
        }
    }

    return null;
}
```

```java
public void eliminarProfesor(Profesor profesor) {
    if (profesor != null && profesores.contains(profesor)) {
        for (Curso curso : new ArrayList<>(profesor.getCursos())) {
            curso.setProfesor(null);
        }

        profesores.remove(profesor);
        System.out.println("Profesor '" + profesor.getNombre() + "' eliminado correctamente.\n");
    } else {
        System.out.println("NO se encontró el profesor a eliminar.\n");
    }
}
```

```java
public void eliminarCurso(Curso curso) {
    if (curso != null && cursos.contains(curso)) {
        curso.setProfesor(null);
        cursos.remove(curso);
        System.out.println("Curso '" + curso.getNombre() + "' eliminado correctamente.\n");
    }
}
```

3. Tareas a realizar:

1. Crea al menos 3 profesores y 5 cursos.

```
// Creamos la Universidad
Universidad unil = new Universidad("UTN");
// Creamos los profesores
Profesor profe1 = new Profesor("01", "Juancito", "Programacion");
Profesor profe2 = new Profesor("02", "Mengana", "Ingles");
Profesor profe3 = new Profesor("03", "Fulano", "AySO");
// Creamos los cursos
Curso ingles1 = new Curso("01", "Ingles 1");
Curso ayso = new Curso("02", "AySO");
Curso programacion1 = new Curso("03", "Programacion 1");
Curso bdd = new Curso("04", "Bases de Datos 1");
Curso programacion2 = new Curso("05", "Programacion 2");
```

2. Agregar profesores y cursos a la universidad.

```
// Agregamos los profesores
unil.agregarProfesor(profe1);
unil.agregarProfesor(profe2);
unil.agregarProfesor(profe3);
// Agregamos los cursos
unil.agregarCurso(ingles1);
unil.agregarCurso(ayso);
unil.agregarCurso(programacion1);
unil.agregarCurso(bdd);
unil.agregarCurso(programacion2);
```

3. Asignar profesores a cursos usando `asignarProfesorACurso(...)`.

```
unil.asignarProfesorACurso("03", "01");     // Juancito
unil.asignarProfesorACurso("05", "01");     // Juancito
unil.asignarProfesorACurso("01", "02");     // Mengana
unil.asignarProfesorACurso("02", "03");     // Fulano
unil.asignarProfesorACurso("04", "03");     // Fulano
```

4. Listar cursos con su profesor y profesores con sus cursos.

```
System.out.println("\n============================================================");
System.out.println("4. Listamos cursos y profesores:");
System.out.println("============================================================");
unil.listarProfesor();
System.out.println();
unil.listarCursos();
```

5. Cambiar el profesor de un curso y verificar que ambos lados quedan sincronizados.

```
System.out.println("\n============================================================");
System.out.println("5. Cambiamos el profesor del curso 'Bases de Datos 1' a Mengana:");
System.out.println("============================================================");
unil.asignarProfesorACurso("04", "02");
unil.listarProfesor();
System.out.println();
unil.listarCursos();
```

6. Remover un curso y confirmar que ya no aparece en la lista del profesor.

```
System.out.println("\n=============================================================");
System.out.println("6. Removemos el curso 'Bases de Datos 1' y verificamos los cambios:");
System.out.println("=============================================================");
unil.eliminarCurso(bdd);
unil.listarProfesor();
System.out.println();
unil.listarCursos();
```

7. Remover un profesor y dejar profesor = null.

```
System.out.println("\n=============================================================");
System.out.println("7. Removemos al profesor Fulano y dejamos sus cursos sin profesor:");
System.out.println("=============================================================");
unil.eliminarProfesor(profe3);
unil.listarProfesor();
System.out.println();
unil.listarCursos();
```

8. Mostrar un reporte: cantidad de cursos por profesor.

```
System.out.println("\n=============================================================");
System.out.println("8. Mostramos un reporte: cantidad de cursos por profesor:");
System.out.println("=============================================================");
unil.listarCursos();
```