

**Trabajo Práctico N° 6**

**Colecciones**

Comision 16

Marinoni Macarena <[marinonimacarena@gmail.com](mailto:marinonimacarena@gmail.com)>

**Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional**

**Programación II**

**Profesor:** Cortez, Alberto

**Tutor:** Bianchi, Neyén

16/11/2025

## ÍNDICE

<b>Objetivo general</b>	<b>3</b>
<b>Marco teórico</b>	<b>3</b>
<b>Caso práctico</b>	<b>4</b>
<b>Caso Práctico 1: Sistema de Stock.</b>	<b>4</b>
<b>Resolución:</b>	<b>5</b>
<b>Caso Práctico 2: Biblioteca y Libros.</b>	<b>9</b>
<b>Resolución:</b>	<b>11</b>
<b>Caso Práctico 3: Universidad, Profesor y Curso (bidireccional 1 a N)</b>	<b>14</b>
<b>Resolución</b>	<b>16</b>
<b>Link a Repositorio Github</b>	<b>19</b>

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

## Caso práctico

### Caso Práctico 1: Sistema de Stock.

#### 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**

##### a. Atributos:

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

##### b. Métodos:

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

##### c. Enum CategoriaProducto

Valores:

- i. ALIMENTOS
- ii. ELECTRONICA
- iii. ROPA
- iv. HOGAR

##### d. Método adicional:

- i. *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

##### a. Atributo:

- i. ArrayList<Producto> productos }

##### b. Métodos requeridos:

- i. agregarProducto(Producto p)
- ii. listarProductos()
- iii. buscarProductoPorId(String id)

- iv. eliminarProducto(String id)
- v. actualizarStock(String id, int nuevaCantidad)
- vi. filtrarPorCategoria(CategoríaProducto categoria)
- vii. obtenerTotalStock()
- viii. obtenerProductoConMayorStock()
- ix. filtrarProductosPorPrecio(double min, double max)
- x. 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.

#### Resolución:

En este punto desarrollamos un Sistema de Stock utilizando colecciones dinámicas en Java (principalmente ArrayList).

Se implementan las clases Producto, Inventario y la enumeración CategoríaProducto, aplicando conceptos fundamentales de Programación Orientada a Objetos como encapsulamiento, relaciones 1 a N, búsqueda, filtrado y manipulación de datos. Finalmente, se realizan 10 tareas que permiten probar todas las funcionalidades del sistema, desde agregar productos hasta generar reportes.

#### Clase Producto

```
/*  
package Tp6_colecciones;  
  
+ [/*...4 lines */]  
public class Producto {  
    private String id;           // Identificador único  
    private String nombre;  
    private double precio;  
    private int cantidad;  
    private CategoríaProducto categoria;  
  
    // Constructor  
    public Producto(String id, String nombre, double precio, int cantidad, CategoríaProducto categoria) {  
        this.id = id;  
        this.nombre = nombre;  
        this.precio = precio;  
        this.cantidad = cantidad;  
        this.categoría = categoría;  
    }  
  
    // Getters y setters (solo los que necesitemos)  
    public String getId() {  
        return id;  
    }  
  
    public String getNombre() {  
        return nombre;  
    }  
  
    public double getPrecio() {  
        return precio;  
    }  
}
```

## Enum Categoria Producto

## Clase Inventario

```

package Tp6_colecciones;

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

/*
 * ...4 lines ...
 */
public class inventario {
    private List<Producto> productos;

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

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

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

```

En lugar de ejecutar todas las tareas una detrás de otra, el programa:

- Creo un inventario con 5 productos inicial

```

// ===== 1) Crear al menos cinco productos con diferentes categorías y agregarlos al inventario =====
System.out.println("\n===== 1) Creación de productos y agregado al inventario =====");

Producto p1 = new Producto("P001", "Arroz 1kg", 1200.0, 50, CategoriaProducto.ALIMENTOS);
Producto p2 = new Producto("P002", "Notebook", 350000.0, 10, CategoriaProducto.ELECTRONICA);
Producto p3 = new Producto("P003", "Remera", 8000.0, 30, CategoriaProducto.ROPA);
Producto p4 = new Producto("P004", "Sartén", 15000.0, 20, CategoriaProducto.HOGAR);
Producto p5 = new Producto("P005", "Galletitas", 900.0, 100, CategoriaProducto.ALIMENTOS);

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

System.out.println("Productos iniciales cargados correctamente.\n");

```

```

===== 1) Creación de productos y agregado al inventario =====
Productos iniciales cargados correctamente.

```

- Muestro un menú interactivo en consola.

```
=====
SISTEMA DE STOCK - MENÚ
=====

1. Listar todos los productos
2. Buscar producto por ID
3. Filtrar productos por categoría
4. Eliminar producto por ID
5. Actualizar stock de un producto
6. Mostrar total de stock disponible
7. Mostrar producto con mayor stock
8. Filtrar productos por rango de precio
9. Mostrar categorías disponibles
10. Agregar nuevo producto
0. Salir

=====
Elegí una opción:
```

- El usuario elige qué acción realizar:

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

```
Elegí una opción: 1

===== 1) Listar todos los productos del inventario =====
Listado de productos en inventario:
ID: P001 | Nombre: Arroz lkg | Precio: $1200.0 | Cantidad: 50 | Categoría: ALIMENTOS (Productos comestibles)
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categoría: ELECTRONICA (Dispositivos electrónicos)
ID: P003 | Nombre: Remera | Precio: $8000.0 | Cantidad: 30 | Categoría: ROPA (Prendas de vestir)
ID: P004 | Nombre: Sartén | Precio: $15000.0 | Cantidad: 20 | Categoría: HOGAR (Artículos para el hogar)
ID: P005 | Nombre: Galletitas | Precio: $900.0 | Cantidad: 100 | Categoría: ALIMENTOS (Productos comestibles)
```

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

```
Elegí una opción: 2

===== 2) Búsqueda de producto por ID =====
Ingresó el ID del producto: P002
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categoría: ELECTRONICA (Dispositivos electrónicos)
```

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

```
===== 3) Filtrar productos por categoría =====
Elegí una categoría:
1. ALIMENTOS - Productos comestibles
2. ELECTRONICA - Dispositivos electrónicos
3. ROPA - Prendas de vestir
4. HOGAR - Artículos para el hogar
Opción: 2
Productos en la categoría ELECTRONICA:
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categoría: ELECTRONICA (Dispositivos electrónicos)
```

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

```
Ingresó el ID del producto a eliminar: P002
Producto con ID P002 eliminado correctamente.
Listado de productos restantes en el inventario:
Listado de productos en inventario:
ID: P001 | Nombre: Arroz lkg | Precio: $1200.0 | Cantidad: 50 | Categoría: ALIMENTOS (Productos comestibles)
ID: P003 | Nombre: Remera | Precio: $8000.0 | Cantidad: 30 | Categoría: ROPA (Prendas de vestir)
ID: P004 | Nombre: Sartén | Precio: $15000.0 | Cantidad: 20 | Categoría: HOGAR (Artículos para el hogar)
ID: P005 | Nombre: Galletitas | Precio: $900.0 | Cantidad: 100 | Categoría: ALIMENTOS (Productos comestibles)
```

#### Actualizar el stock de un producto existente.

```
Ingresó el ID del producto a actualizar: P003
Ingresó la nueva cantidad de stock: 50
Stock actualizado correctamente.
```

#### Mostrar el total de stock disponible.

```
Elegí una opción: 6

===== 6) Total de stock disponible en el inventario =====
Total de unidades en inventario: 210
```

**Obtener y mostrar el producto con mayor stock.**

```
Producto con mayor stock:
ID: P005 | Nombre: Galletitas | Precio: $900.0 | Cantidad: 100 | Categoría: ALIMENTOS (Productos comestibles)
```

**Filtrar productos con precios entre \$1000 y \$3000.**

```
Ingresó el precio mínimo: 1000
Ingresó el precio máximo: 3000
Productos entre $1000.0 y $3000.0:
ID: P001 | Nombre: Arroz 1kg | Precio: $1200.0 | Cantidad: 50 | Categoría: ALIMENTOS (Productos comestibles)
```

**Mostrar las categorías disponibles con sus descripciones.**

```
Categorías disponibles:
- ALIMENTOS: Productos comestibles
- ELECTRONICA: Dispositivos electrónicos
- ROPA: Prendas de vestir
- HOGAR: Artículos para el hogar
```

- **Agregar producto**, se agrega una opción más, no requerida en la consigna pero entiendo que mejora la experiencia del usuario.

```
===== Extra) Agregar un nuevo producto al inventario =====
Ingresó el ID del producto: P006
Ingresó el nombre del producto: Pantalón
Ingresó el precio del producto: 55000.0
Valor inválido. Ingresó un número (usar punto como separador decimal).
Ingresó el precio del producto: 55.000
Ingresó la cantidad en stock: 25
Elegió una categoría:
1. ALIMENTOS - Productos comestibles
2. ELECTRONICA - Dispositivos electrónicos
3. ROPA - Prendas de vestir
4. HOGAR - Artículos para el hogar
Opción: 3
Producto agregado correctamente:
ID: P006 | Nombre: Pantalón | Precio: $55000.0 | Cantidad: 25 | Categoría: ROPA (Prendas de vestir)
```

**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****a. Atributos:**

- id (String) → Identificador único del autor.
- nombre (String) → Nombre del autor.
- nacionalidad (String) → Nacionalidad del autor.

**b. Métodos:**

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

**Clase Libro****a. Atributos:**

- i. isbn (String) → Identificador único del libro.
  - ii. titulo (String) → Título del libro.
  - iii. anioPublicacion (int) → Año de publicación.
  - iv. autor (Autor) → Autor del libro.
- b. Métodos:
- i. mostrarInfo() → Muestra título, ISBN, año y autor.

### **Clase Biblioteca**

- a. Atributo:
  - i. String nombre
  - ii. List<Libro> libros → Colección de libros de la biblioteca.
- b. Métodos requeridos:
  - i. agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor)
  - ii. listarLibros()
  - iii. buscarLibroPorIsbn(String isbn)
  - iv. eliminarLibro(String isbn)
  - v. obtenerCantidadLibros()
  - vi. filtrarLibrosPorAnio(int anio)
  - vii. 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.

### **Resolución:**

En este ejercicio vamos a modelar una Biblioteca que contiene muchos Libros, y cada libro tiene un Autor asociado.

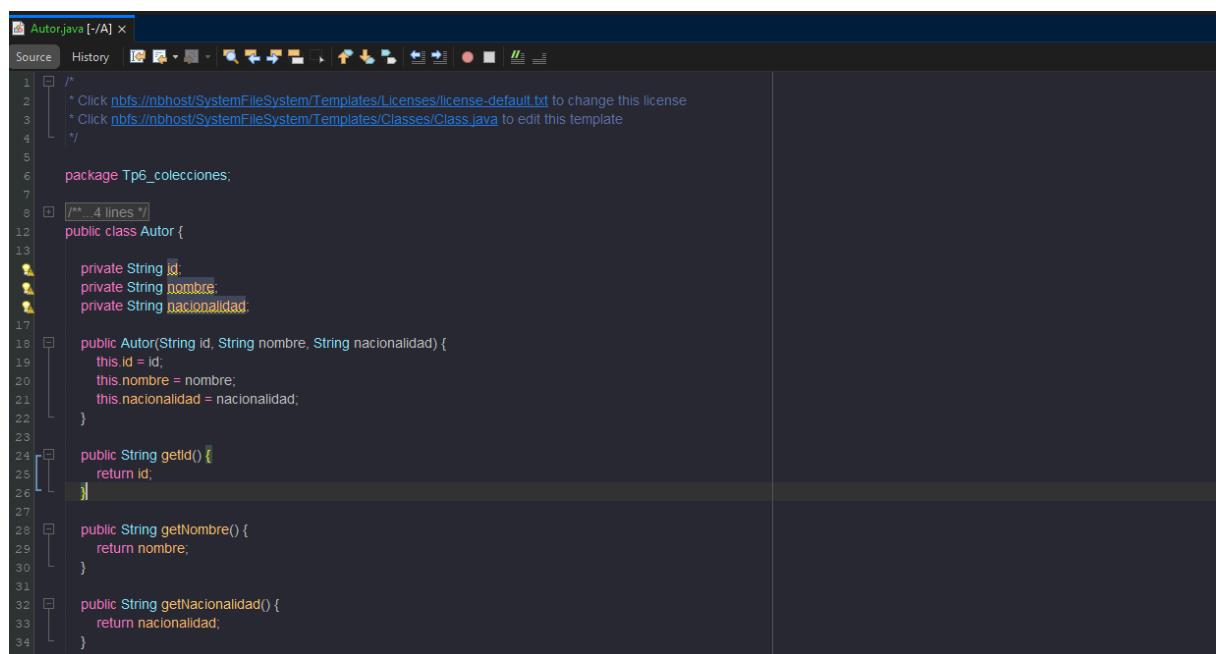
La relación es de composición 1 a N:

- Una Biblioteca contiene una colección de Libro.
- Cada Libro pertenece obligatoriamente a una Biblioteca.
- Si la Biblioteca desaparece, conceptualmente desaparecen sus libros.

Vamos a practicar:

- Uso de clases y composición (Biblioteca → List<Libro>)
- Manejo de colecciones dinámicas (ArrayList)
- Búsqueda, filtrado y eliminación de objetos
- Listado de autores a partir de los libros registrados

## Clase Autor



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Tp6_colecciones;

public class Autor {

    private String id;
    private String nombre;
    private String nacionalidad;

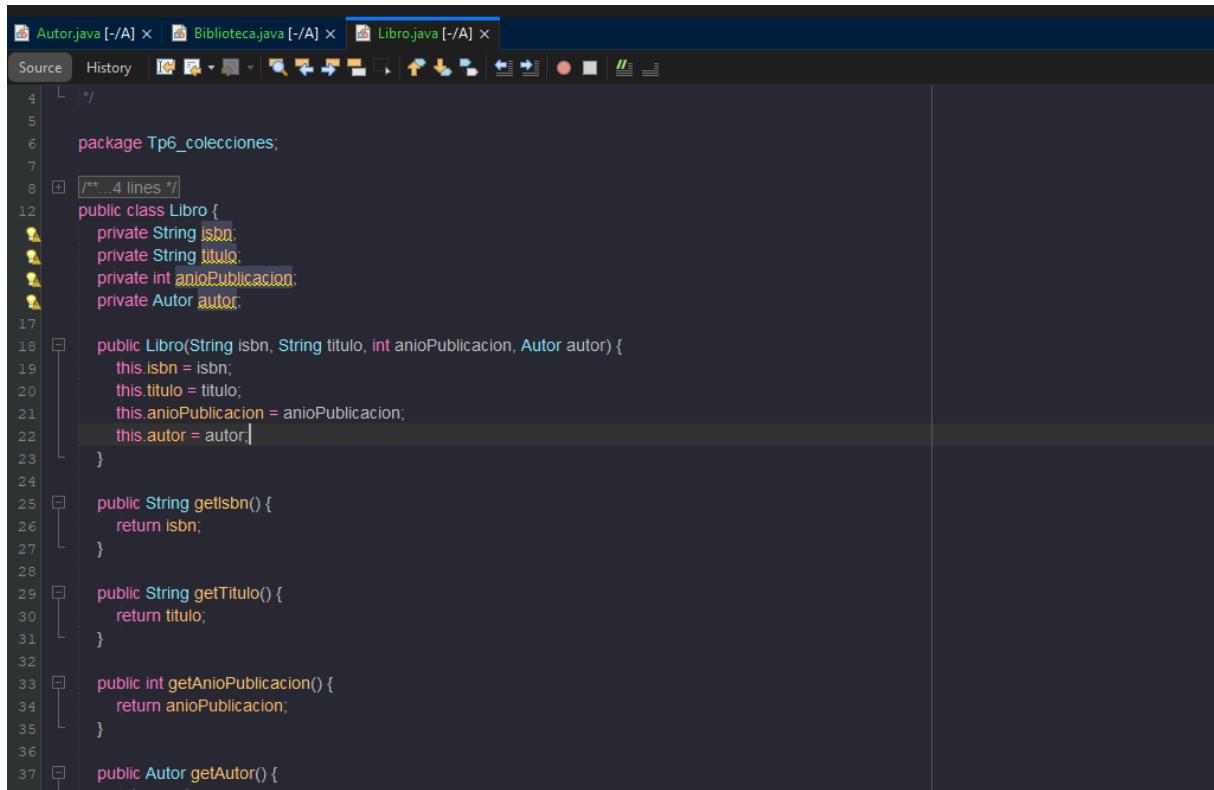
    public Autor(String id, String nombre, String nacionalidad) {
        this.id = id;
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }

    public String getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public String getNacionalidad() {
        return nacionalidad;
    }
}
```

## Clase Libro

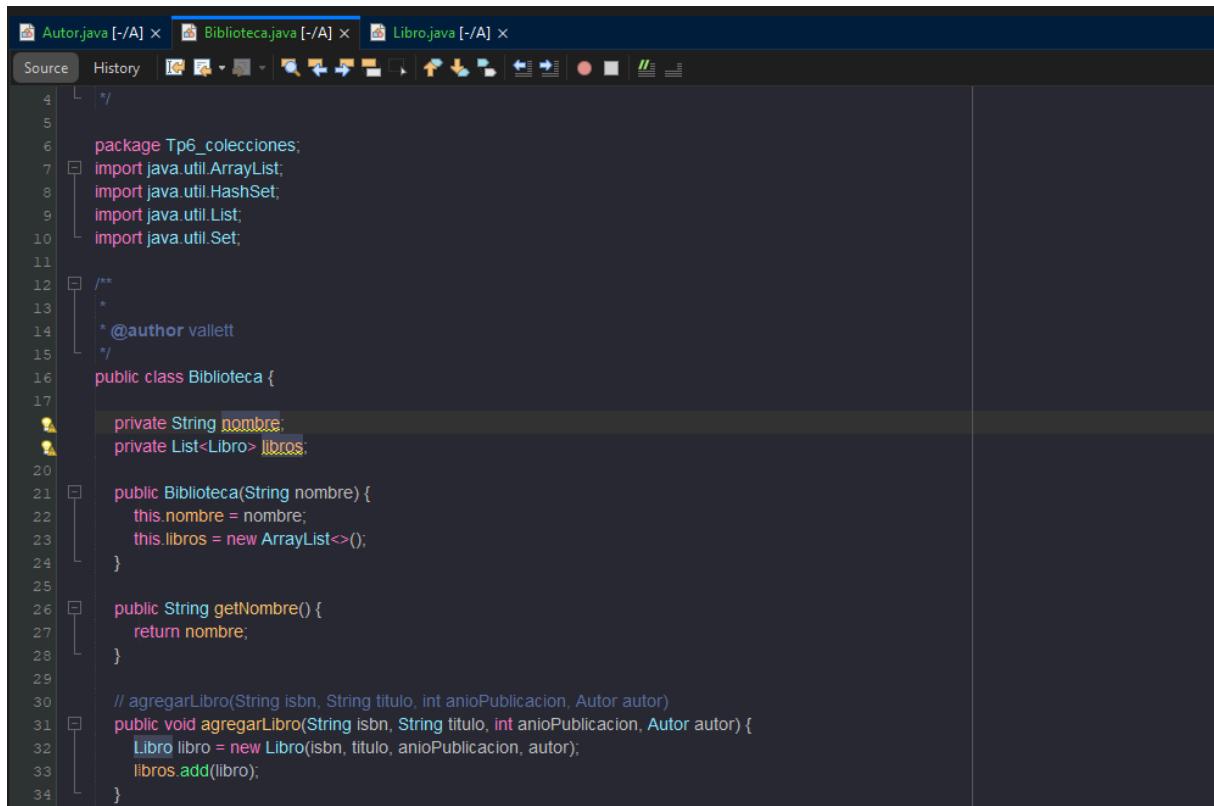


```

4   */
5
6 package Tp6_colecciones;
7
8 /**
9  * .4 lines */
10 public class Libro {
11     private String isbn;
12     private String titulo;
13     private int anioPublicacion;
14     private Autor autor;
15
16     public Libro(String isbn, String titulo, int anioPublicacion, Autor autor) {
17         this.isbn = isbn;
18         this.titulo = titulo;
19         this.anioPublicacion = anioPublicacion;
20         this.autor = autor;
21     }
22
23
24     public String getIsbn() {
25         return isbn;
26     }
27
28     public String getTitulo() {
29         return titulo;
30     }
31
32     public int getAnioPublicacion() {
33         return anioPublicacion;
34     }
35
36     public Autor getAutor() {
37
38     }

```

## Clase Biblioteca



```

4   */
5
6 package Tp6_colecciones;
7
8 import java.util.ArrayList;
9 import java.util.HashSet;
10 import java.util.List;
11 import java.util.Set;
12
13 /**
14  * @author vallett
15  */
16 public class Biblioteca {
17
18     private String nombre;
19     private List<Libro> libros;
20
21     public Biblioteca(String nombre) {
22         this.nombre = nombre;
23         this.libros = new ArrayList<>();
24     }
25
26     public String getNombre() {
27         return nombre;
28     }
29
30     // agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor)
31     public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
32         Libro libro = new Libro(isbn, titulo, anioPublicacion, autor);
33         libros.add(libro);
34     }

```

A continuación, se presentan las tareas implementadas, cada una acompañada de su salida correspondiente, permitiendo visualizar el funcionamiento completo del sistema.

### Creamos una biblioteca.

Crear al menos tres autores

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

The screenshot shows an IDE interface with the code for `SistemaBiblioteca.java` in the editor and its execution output in the terminal below.

```

import java.util.List;
import java.util.Scanner;

public class SistemaBiblioteca {

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        // 1) Creamos una biblioteca.
        System.out.println("===== 1) Creación de la biblioteca =====");
        Biblioteca biblioteca = new Biblioteca("Biblioteca Central");
        System.out.println("Biblioteca creada: " + biblioteca.getNombre());

        // 2) Crear al menos tres autores.
        System.out.println("===== 2) Creación de autores =====");
        Autor autor1 = new Autor("A001", "Gabriel García Márquez", "Colombia");
    }
}

```

**Output:**

```

run:
=====
===== 1) Creaci n de la biblioteca =====
Biblioteca creada: Biblioteca Central

===== 2) Creaci n de autores =====
Autores creados:
Autor { ID='A001', Nombre='Gabriel Garc a M rquez', Nacionalidad='Colombia' }
Autor { ID='A002', Nombre='Julio Cort zar', Nacionalidad='Argentina' }
Autor { ID='A003', Nombre='Jane Austen', Nacionalidad='Reino Unido' }

===== 3) Agregar libros a la biblioteca =====
Libros iniciales cargados correctamente.

=====
SISTEMA DE BIBLIOTECA - MEN 
=====
1. Listar todos los libros
2. Buscar libro por ISBN
3. Filtrar libros por a o de publicaci n
4. Eliminar libro por ISBN y listar restantes
5. Mostrar cantidad total de libros
6. Listar autores de los libros disponibles
7. Agregar nuevo libro
0. Salir

Eleg  una opci n:

```

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

```

=====
4) Listar todos los libros con su informaci n y la del autor =====
Listado de libros de la biblioteca "Biblioteca Central":
Libro { ISBN='ISBN001', T tulo='Cien a os de soledad', A o=1967, Autor=Gabriel Garc a M rquez (Colombia) }
Libro { ISBN='ISBN002', T tulo='El amor en los tiempos del c lera', A o=1985, Autor=Gabriel Garc a M rquez (Colombia) }
Libro { ISBN='ISBN003', T tulo='Rayuela', A o=1963, Autor=Julio Cort zar (Argentina) }
Libro { ISBN='ISBN004', T tulo='Orgullo y p rejuicio', A o=1813, Autor=Jane Austen (Reino Unido) }
Libro { ISBN='ISBN005', T tulo='Emma', A o=1815, Autor=Jane Austen (Reino Unido) }

```

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

```

=====
5) Buscar libro por ISBN =====
Ingres  el ISBN del libro: A004
No se encontr  un libro con ISBN A004

```

### Filtrar y mostrar los libros publicados en un a o espec fico.

```
===== 6) Filtrar libros por año de publicación =====
Ingresó el año a filtrar: 1985
Libros publicados en el año 1985:
Libro { ISBN='ISBN002', Título='El amor en los tiempos del cólera', Año=1985, Autor=Gabriel García Márquez (Colombia) }
```

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

```
===== 7) Eliminar libro por ISBN y listar los restantes =====
Ingresó el ISBN del libro a eliminar: ISBN003
Libro con ISBN ISBN003 eliminado correctamente.
Libros restantes en la biblioteca:
Listado de libros de la biblioteca "Biblioteca Central":
Libro { ISBN='ISBN001', Título='Cien años de soledad', Año=1967, Autor=Gabriel García Márquez (Colombia) }
Libro { ISBN='ISBN002', Título='El amor en los tiempos del cólera', Año=1985, Autor=Gabriel García Márquez (Colombia) }
Libro { ISBN='ISBN004', Título='Orgullo y prejuicio', Año=1813, Autor=Jane Austen (Reino Unido) }
Libro { ISBN='ISBN005', Título='Emma', Año=1815, Autor=Jane Austen (Reino Unido) }
```

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

```
===== 9) Listar todos los autores de los libros disponibles =====
Autores de los libros disponibles en la biblioteca:
Autor { ID='A003', Nombre='Jane Austen', Nacionalidad='Reino Unido' }
Autor { ID='A001', Nombre='Gabriel García Márquez', Nacionalidad='Colombia' }
```

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

```
===== 9) Listar todos los autores de los libros disponibles =====
Autores de los libros disponibles en la biblioteca:
Autor { ID='A003', Nombre='Jane Austen', Nacionalidad='Reino Unido' }
Autor { ID='A001', Nombre='Gabriel García Márquez', Nacionalidad='Colombia' }
```

## 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

- a. Atributos:
  - i. id (String) → Identificador único.
  - ii. nombre (String) → Nombre completo.
  - iii. especialidad (String) → Área principal.
  - iv. List<Curso> cursos → Cursos que dicta.
- b. Métodos sugeridos:
  - i. agregarCurso(Curso c) → Agrega el curso a su lista si no está y sincroniza el lado del curso.

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

### **Clase Curso**

- a. Atributos:
  - i. codigo (String) → Código único.
  - ii. nombre (String) → Nombre del curso.
  - iii. profesor (Profesor) → Profesor responsable.
- b. Métodos sugeridos:
  - i. setProfesor(Profesor p) → Asigna/cambia el profesor sincronizando ambos lados: o Si tenía profesor previo, quitarse de su lista.
  - ii. mostrarInfo() → Muestra código, nombre y nombre del profesor (si tiene).

### **Clase Universidad**

- a. Atributos:
  - i. String nombre
  - ii. List<Profesor> profesores
  - iii. List<Curso> cursos
- b. Métodos requeridos:
  - i. agregarProfesor(Profesor p)
  - ii. agregarCurso(Curso c)
  - iii. asignarProfesorACurso(String codigoCurso, String idProfesor) → Usa setProfesor del curso.
  - iv. listarProfesores() / listarCursos()
  - v. buscarProfesorPorId(String id)
  - vi. buscarCursoPorCodigo(String codigo)
  - vii. eliminarCurso(String codigo) → Debe romper la relación con su profesor si la hubiera.
  - viii. 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.

## Resolución

En este ejercicio vamos a modelar un sistema académico donde: Un Profesor dicta muchos Cursos, cada Curso tiene exactamente un Profesor responsable.

La relación es bidireccional 1 a N:

- Desde Curso puedo acceder a su Profesor
  - Desde Profesor puedo acceder a la lista de Curso que dicta

Además, una clase Universidad administra: Altas y bajas de profesores y cursos, Asignación de profesor a curso, Búsquedas, listados y un reporte de cantidad de cursos por profesor

## Clase Profesor

## Clase Curso

The screenshot shows an IDE interface with several tabs at the top: Profesor.java [-/A] x, Curso.java [-/A] x, Universidad.java [-/A] x, and SistemaUniversidad.java [-/A] x. The main window displays the source code for the Curso.java file. The code defines a class Curso with private attributes codigo and nombre, and a constructor that initializes them. It also includes methods to get each attribute and a method setProfesor that returns null if the passed Profesor object is already assigned. The code is color-coded with syntax highlighting: comments are gray, strings are yellow, and keywords like package, public, and private are purple. Brackets and braces are shown in black. The code editor has a toolbar with various icons for file operations and navigation.

```
package Tp6_colecciones;

public class Curso {

    private String codigo;
    private String nombre;
    private Profesor profesor; // puede ser null

    public Curso(String codigo, String nombre) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.profesor = null;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public Profesor getProfesor() {
        return profesor;
    }

    // setProfesor(Profesor p) → sincroniza ambos lados
    public void setProfesor(Profesor p) {
        if (this.profesor == p) {
            return; // nada que hacer
        }
    }
}
```

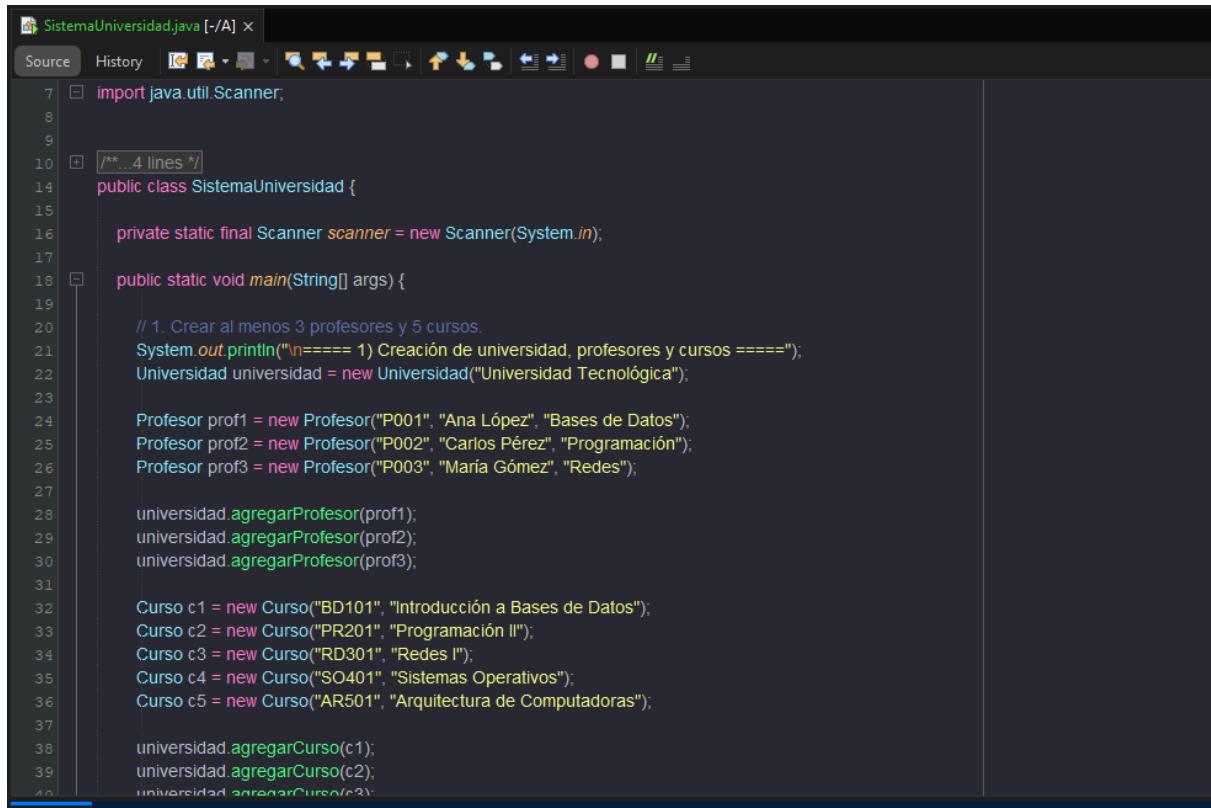
## Clase Universidad

A continuación, se presentan las tareas implementadas, cada una acompañada de su salida correspondiente, permitiendo visualizar el funcionamiento completo del sistema.

**Crear al menos 3 profesores y 5 cursos.**

**Agregar profesores y cursos a la universidad.**

**Asignar profesores a cursos usando asignarProfesorACurso(...).**



```

1 import java.util.Scanner;
2
3 /**
4  * ... 4 lines ...
5 */
6 public class SistemaUniversidad {
7
8     private static final Scanner scanner = new Scanner(System.in);
9
10    public static void main(String[] args) {
11
12        // 1. Crear al menos 3 profesores y 5 cursos.
13        System.out.println("\n===== 1) Creación de universidad, profesores y cursos =====");
14        Universidad universidad = new Universidad("Universidad Tecnológica");
15
16        Profesor prof1 = new Profesor("P001", "Ana López", "Bases de Datos");
17        Profesor prof2 = new Profesor("P002", "Carlos Pérez", "Programación");
18        Profesor prof3 = new Profesor("P003", "María Gómez", "Redes");
19
20        universidad.agregarProfesor(prof1);
21        universidad.agregarProfesor(prof2);
22        universidad.agregarProfesor(prof3);
23
24        Curso c1 = new Curso("BD101", "Introducción a Bases de Datos");
25        Curso c2 = new Curso("PR201", "Programación II");
26        Curso c3 = new Curso("RD301", "Redes I");
27        Curso c4 = new Curso("SO401", "Sistemas Operativos");
28        Curso c5 = new Curso("AR501", "Arquitectura de Computadoras");
29
30        universidad.agregarCurso(c1);
31        universidad.agregarCurso(c2);
32        universidad.agregarCurso(c3);
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
305
306
307
308
309
309
310
311
312
313
313
314
315
316
316
317
318
319
319
320
321
322
323
323
324
325
325
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
```

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

```
===== 4) Listar cursos con su profesor =====
Cursos de la universidad Universidad Tecnológica:
Curso { Código='BD101', Nombre='Introducción a Bases de Datos', Profesor='Ana López' }
Curso { Código='PR201', Nombre='Programación II', Profesor='Carlos Pérez' }
Curso { Código='RD301', Nombre='Redes I', Profesor='María Gómez' }
Curso { Código='SO401', Nombre='Sistemas Operativos', Profesor='Carlos Pérez' }
Curso { Código='AR501', Nombre='Arquitectura de Computadoras', Profesor='María Gómez' }
```

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

```
===== 6) Eliminar curso y verificar sincronización =====
Ingresó el código del curso a eliminar: PR201
Curso eliminado correctamente.
Cursos actuales de la universidad:
Cursos de la universidad Universidad Tecnológica:
Curso { Código='BD101', Nombre='Introducción a Bases de Datos', Profesor='Ana López' }
Curso { Código='RD301', Nombre='Redes I', Profesor='María Gómez' }
Curso { Código='SO401', Nombre='Sistemas Operativos', Profesor='Carlos Pérez' }
Curso { Código='AR501', Nombre='Arquitectura de Computadoras', Profesor='María Gómez' }
```

**Remover un profesor y dejar profesor = null,**

```
===== 7) Eliminar profesor y dejar cursos sin profesor =====
Ingresó el ID del profesor a eliminar: P001
Profesor eliminado correctamente.
Cursos (para verificar que quedaron sin profesor si correspondía):
Cursos de la universidad Universidad Tecnológica:
Curso { Código='BD101', Nombre='Introducción a Bases de Datos', Profesor='Sin profesor asignado' }
Curso { Código='RD301', Nombre='Redes I', Profesor='María Gómez' }
Curso { Código='SO401', Nombre='Sistemas Operativos', Profesor='Carlos Pérez' }
Curso { Código='AR501', Nombre='Arquitectura de Computadoras', Profesor='María Gómez' }
```

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

```
===== 8) Reporte: cantidad de cursos por profesor =====
===== Reporte: Cantidad de cursos por profesor =====
Ana López: 1 curso(s)
Carlos Pérez: 2 curso(s)
María Gómez: 2 curso(s)
```

**Link a Repositorio Github**

[https://github.com/MaquiMarinoni/TUPaD\\_P2-C22025.git](https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git)