UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Práctico N° 6

Colecciones

Comision 16

Marinoni Macarena <*marinonimacarena@gmail.com*>

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional

Programación II

Profesor: Cortez, Alberto

Tutor: Bianchi, Neyén

16/11/2025

ÍNDICE

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

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.
Agregación 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 la 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

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(CategoriaProducto 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 CategoriaProducto, 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

```java
*/

package Tp6_colecciones;

/**...4 lines */
public class Producto {
    private String id;              // Identificador único
    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;
    }

    // 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

```java
CategoriaProducto.java [-/A] ×  Producto.java [-/A] ×  Inventario.java [-/A] ×  SistemaStock.java [-/A] ×
Source  History
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

package Tp6_colecciones;

/**...4 lines */
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

```java
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

```java
// ===== 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) Creaci□n de productos y agregado al inventario =====
Productos iniciales cargados correctamente.
```

- Muestro un menú interactivo en consola.

```
========================================
        SISTEMA DE STOCK - MEN�
========================================
1. Listar todos los productos
2. Buscar producto por ID
3. Filtrar productos por categor�a
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 categor�as disponibles
10. Agregar nuevo producto
0. Salir
========================================
Eleg� una opci�n:
```

- El usuario elige qué acción realizar:

Listar todos los productos mostrando su información y categoría.

```
Eleg� una opci�n: 1

===== 1) Listar todos los productos del inventario =====
Listado de productos en inventario:
ID: P001 | Nombre: Arroz 1kg | Precio: $1200.0 | Cantidad: 50 | Categor�a: ALIMENTOS (Productos comestibles)
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categor�a: ELECTRONICA (Dispositivos electr�nicos)
ID: P003 | Nombre: Remera | Precio: $8000.0 | Cantidad: 30 | Categor�a: ROPA (Prendas de vestir)
ID: P004 | Nombre: Sart�n | Precio: $15000.0 | Cantidad: 20 | Categor�a: HOGAR (Art�culos para el hogar)
ID: P005 | Nombre: Galletitas | Precio: $900.0 | Cantidad: 100 | Categor�a: ALIMENTOS (Productos comestibles)
```

Buscar un producto por ID y mostrar su información.

```
Eleg� una opci�n: 2

===== 2) B�squeda de producto por ID =====
Ingres� el ID del producto: P002
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categor�a: ELECTRONICA (Dispositivos electr�nicos)
```

Filtrar y mostrar productos que pertenezcan a una categoría específica.

```
===== 3) Filtrar productos por categor�a =====
Eleg� una categor�a:
1. ALIMENTOS - Productos comestibles
2. ELECTRONICA - Dispositivos electr�nicos
3. ROPA - Prendas de vestir
4. HOGAR - Art�culos para el hogar
Opci�n: 2
Productos en la categor�a ELECTRONICA:
ID: P002 | Nombre: Notebook | Precio: $350000.0 | Cantidad: 10 | Categor�a: ELECTRONICA (Dispositivos electr�nicos)
```

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

```
Ingres� 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 1kg | Precio: $1200.0 | Cantidad: 50 | Categor�a: ALIMENTOS (Productos comestibles)
ID: P003 | Nombre: Remera | Precio: $8000.0 | Cantidad: 30 | Categor�a: ROPA (Prendas de vestir)
ID: P004 | Nombre: Sart�n | Precio: $15000.0 | Cantidad: 20 | Categor�a: HOGAR (Art�culos para el hogar)
ID: P005 | Nombre: Galletitas | Precio: $900.0 | Cantidad: 100 | Categor�a: ALIMENTOS (Productos comestibles)
```

Actualizar el stock de un producto existente.

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

Mostrar el total de stock disponible.

```
Eleg� una opci�n: 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 | Categor�a: ALIMENTOS (Productos comestibles)
```

Filtrar productos con precios entre $1000 y $3000.

```
Ingres� el precio m�nimo: 1000
Ingres� el precio m�ximo: 3000
Productos entre $1000.0 y $3000.0:
ID: P001 | Nombre: Arroz 1kg | Precio: $1200.0 | Cantidad: 50 | Categor�a: ALIMENTOS (Productos comestibles)
```

Mostrar las categorías disponibles con sus descripciones.

```
Categor�as disponibles:
- ALIMENTOS: Productos comestibles
- ELECTRONICA: Dispositivos electr�nicos
- ROPA: Prendas de vestir
- HOGAR: Art�culos 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 =====
Ingres� el ID del producto: P006
Ingres� el nombre del producto: Pantalón
Ingres� el precio del producto: 55000.0
Valor inv�lido. Ingres� un n�mero (usar punto como separador decimal).
Ingres� el precio del producto: 55.000
Ingres� la cantidad en stock: 25
Eleg� una categor�a:
1. ALIMENTOS - Productos comestibles
2. ELECTRONICA - Dispositivos electr�nicos
3. ROPA - Prendas de vestir
4. HOGAR - Art�culos para el hogar
Opci�n: 3
Producto agregado correctamente:
ID: P006 | Nombre: Pantal�n | Precio: $55000.0 | Cantidad: 25 | Categor�a: 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:
        i. id (String) → Identificador único del autor.
        ii. nombre (String) → Nombre del autor.
        iii. nacionalidad (String) → Nacionalidad del autor.
    b. Métodos:
        i. 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.

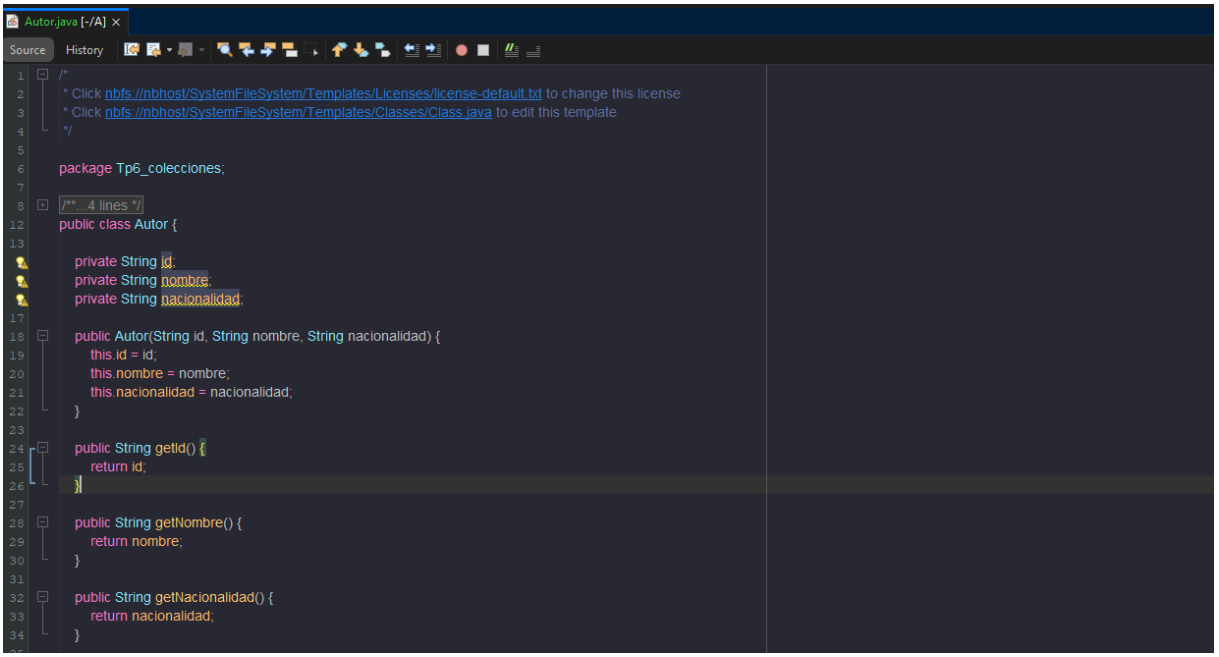
La relación es de composición 1 a N:

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

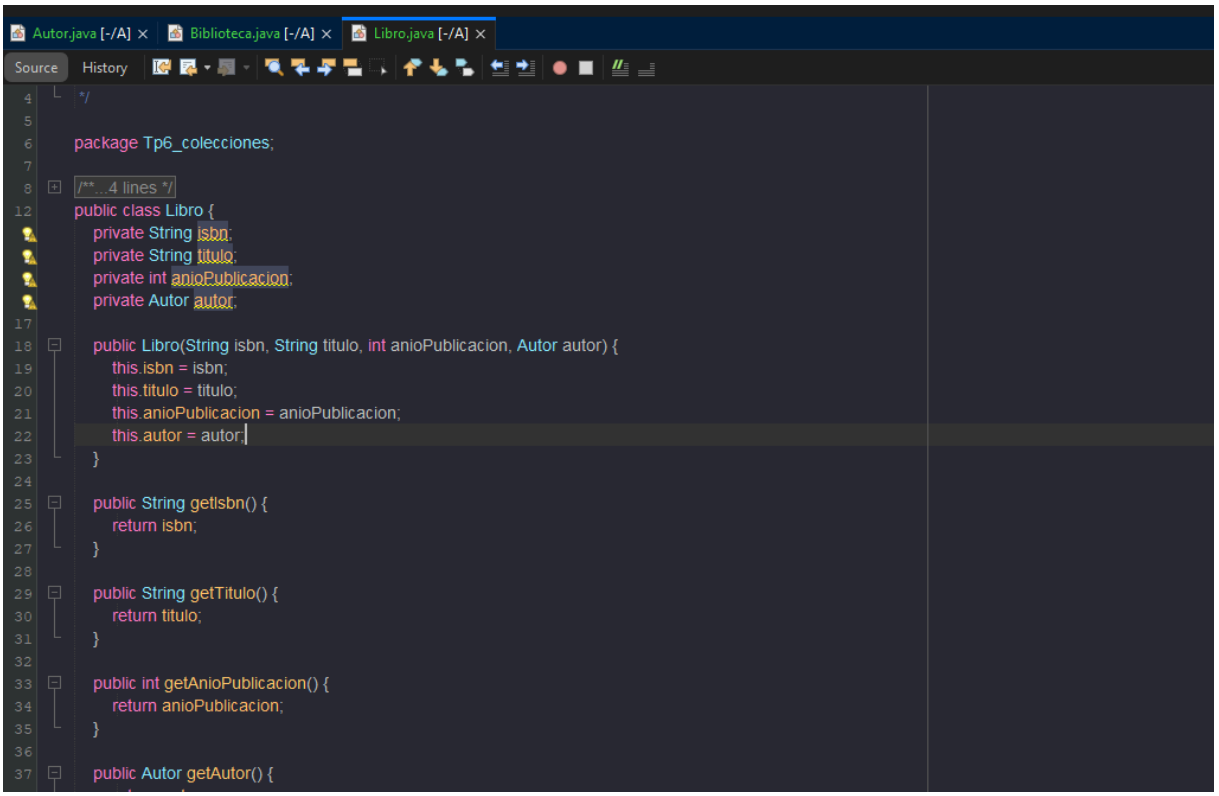
Vamos a practicar:

- 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

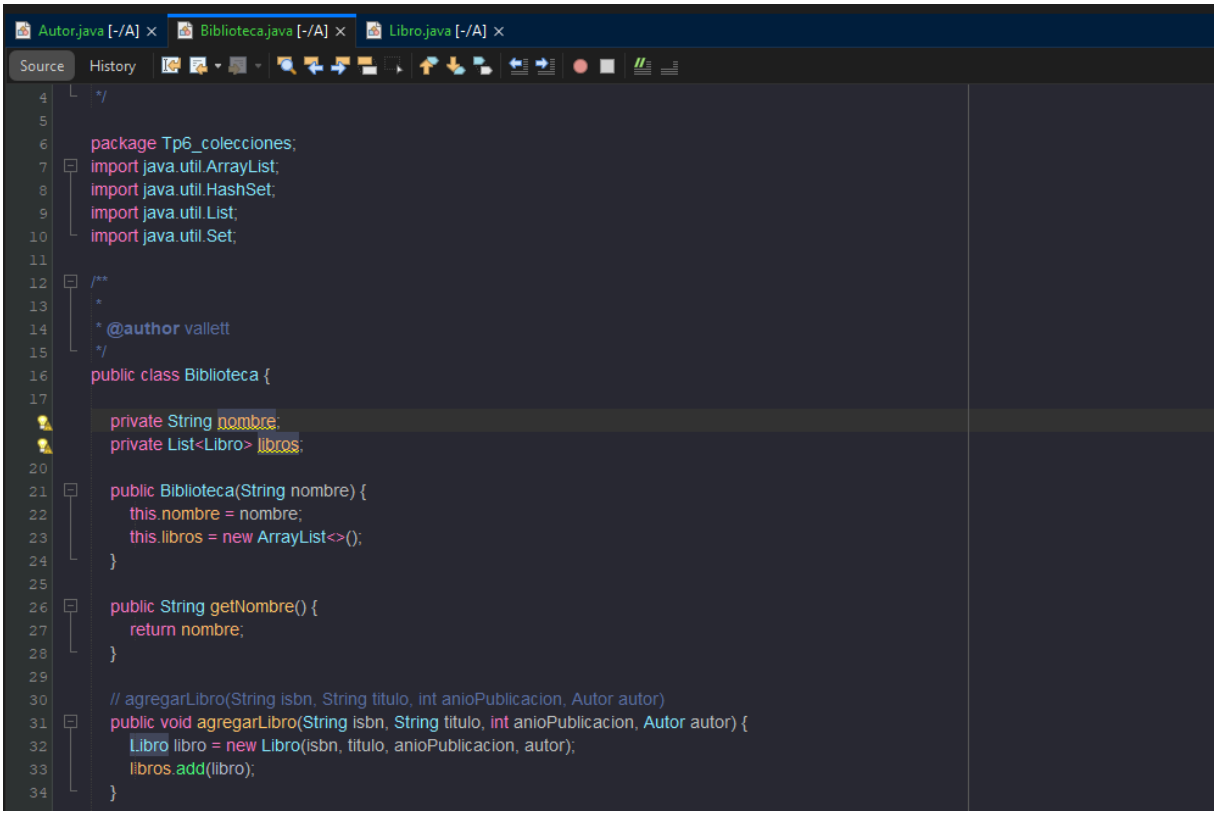


Clase Libro



```java
*/

package Tp6_colecciones;

/**...4 lines */
public class Libro {
    private String isbn;
    private String titulo;
    private int anioPublicacion;
    private Autor autor;

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

    public String getIsbn() {
        return isbn;
    }

    public String getTitulo() {
        return titulo;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    public Autor getAutor() {
```

Clase Biblioteca



```java
*/

package Tp6_colecciones;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

/**
 *
 * @author vallett
 */
public class Biblioteca {

    private String nombre;
    private List<Libro> libros;

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

    public String getNombre() {
        return nombre;
    }

    // agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor)
    public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
        Libro libro = new Libro(isbn, titulo, anioPublicacion, autor);
        libros.add(libro);
    }
```

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.

```
SistemaBiblioteca.java [-/A]
Source  History
7  import java.util.List;
8  import java.util.Scanner;
9
10 /**...4 lines */
14 public class SistemaBiblioteca {
15
16     private static final Scanner scanner = new Scanner(System.in);
17
18     public static void main(String[] args) {
19
20         // 1) Creamos una biblioteca.
21         System.out.println("\n===== 1) Creación de la biblioteca =====");
22         Biblioteca biblioteca = new Biblioteca("Biblioteca Central");
23         System.out.println("Biblioteca creada: " + biblioteca.getNombre());
24
25         // 2) Crear al menos tres autores.
26         System.out.println("\n===== 2) Creación de autores =====");
27         Autor autor1 = new Autor("A001", "Gabriel García Márquez", "Colombia");
```

```
Output
n\Desktop\UTN-TUPaD-P2_Valletto ×  TP6_Colecciones (run) ×  Debugger Console ×  TP6_Colecciones (run) #2 ×  TP6_Colecciones (run) #3 ×  TP6_Colecciones (run)
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 prejuicio', 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 año específico.

```
===== 6) Filtrar libros por a□o de publicaci□n =====
Ingres□ el a□o a filtrar: 1985
Libros publicados en el a□o 1985:
Libro { ISBN='ISBN002', T□tulo='El amor en los tiempos del c□lera', A□o=1985, Autor=Gabriel Garc□a M□rquez (Colombia) }
```

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

```
===== 7) Eliminar libro por ISBN y listar los restantes =====
Ingres□ 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', 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='ISBN004', T□tulo='Orgullo y prejuicio', A□o=1813, Autor=Jane Austen (Reino Unido) }
Libro { ISBN='ISBN005', T□tulo='Emma', A□o=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 Garc□a M□rquez', 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 Garc□a M□rquez', 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

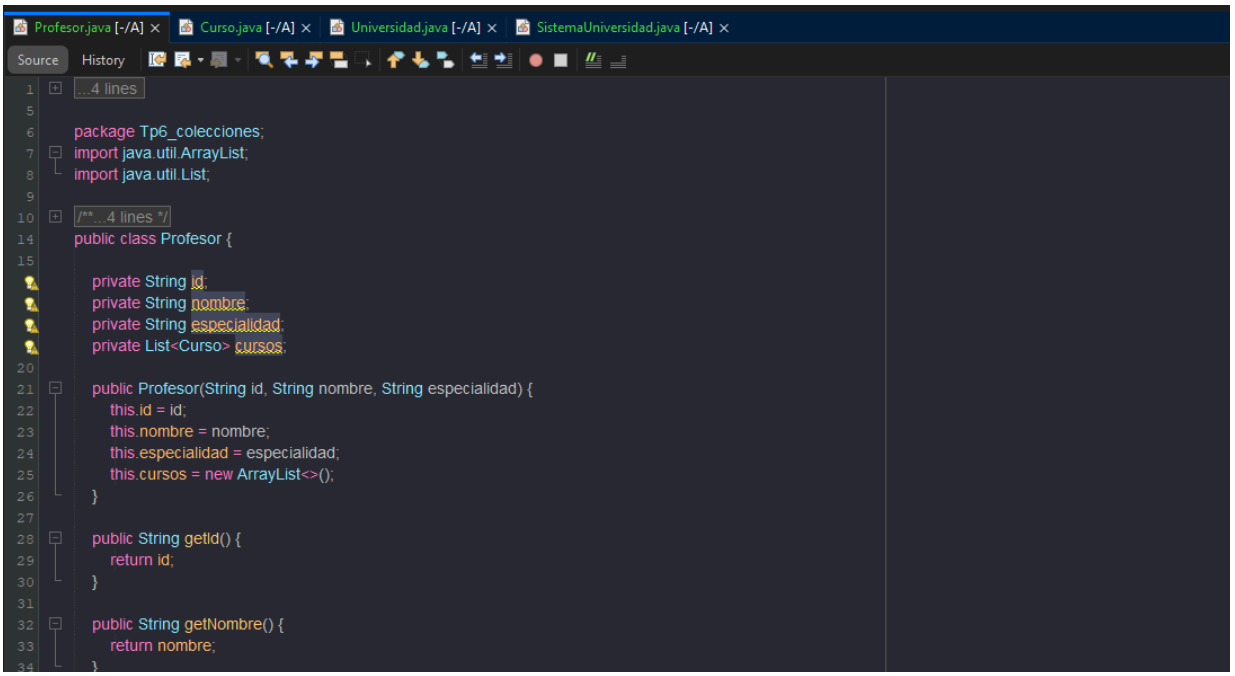
En este ejercicio vamos a modelar un sistema académico donde: Un Profesor dicta muchos Cursos, cada Curso tiene exactamente un Profesor responsable.

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

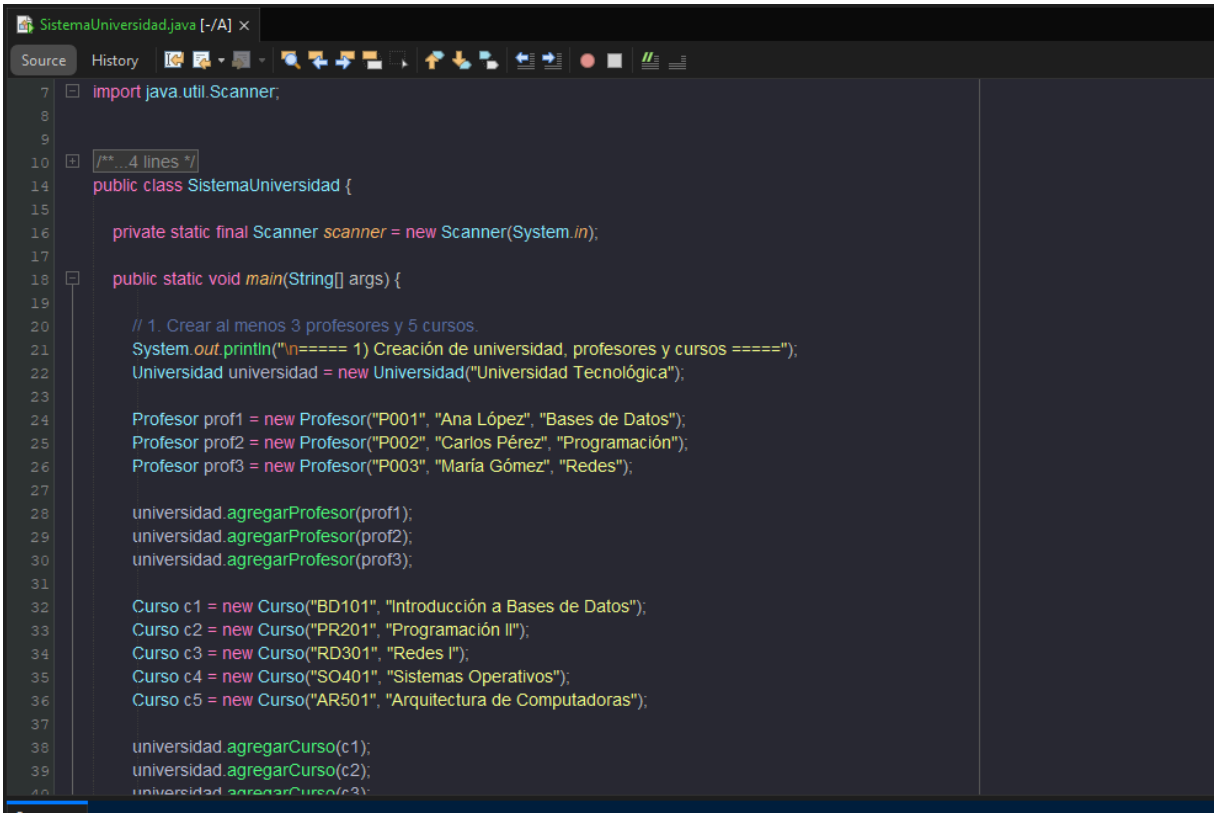
```java
...4 lines

package Tp6_colecciones;

/**...4 lines */
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

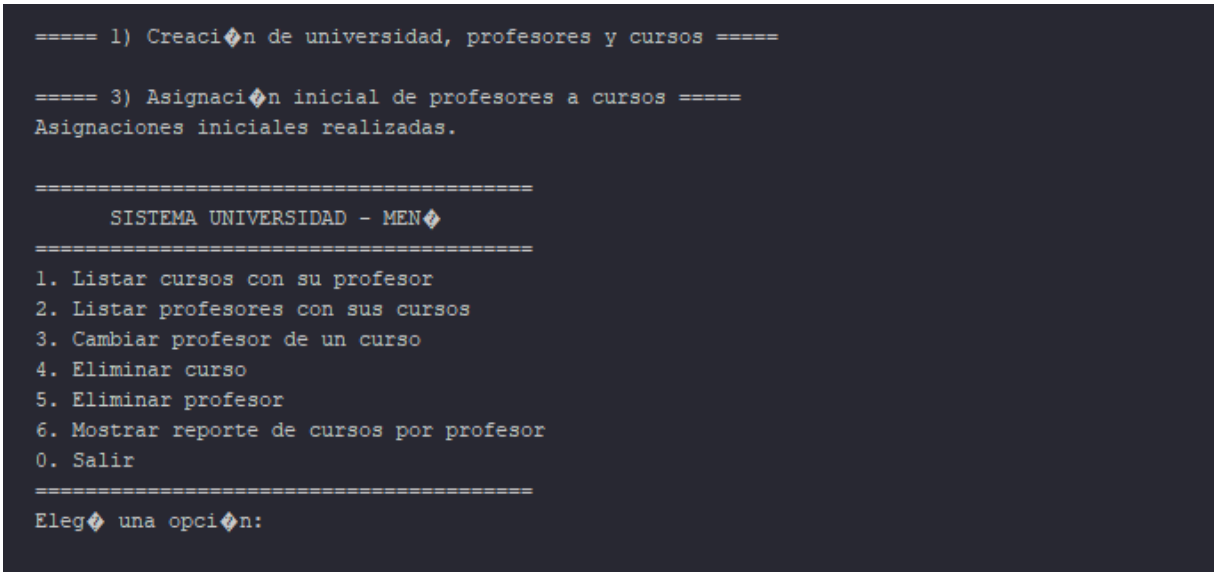
```java
...4 lines

package Tp6_colecciones;
import java.util.ArrayList;
import java.util.List;

/**...4 lines */
public class Universidad {

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

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

    public String getNombre() {
        return nombre;
    }

    // agregarProfesor(Profesor p)
    public void agregarProfesor(Profesor p) {
        if (p != null && !profesores.contains(p)) {
            profesores.add(p);
        }
    }

    // agregarCurso(Curso c)
    public void agregarCurso(Curso c) {
        if (c != null && !cursos.contains(c)) {
```

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(...).



```java
import java.util.Scanner;


/**...4 lines */
public class SistemaUniversidad {

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

    public static void main(String[] args) {

        // 1. Crear al menos 3 profesores y 5 cursos.
        System.out.println("\n===== 1) Creación de universidad, profesores y cursos =====");
        Universidad universidad = new Universidad("Universidad Tecnológica");

        Profesor prof1 = new Profesor("P001", "Ana López", "Bases de Datos");
        Profesor prof2 = new Profesor("P002", "Carlos Pérez", "Programación");
        Profesor prof3 = new Profesor("P003", "María Gómez", "Redes");

        universidad.agregarProfesor(prof1);
        universidad.agregarProfesor(prof2);
        universidad.agregarProfesor(prof3);

        Curso c1 = new Curso("BD101", "Introducción a Bases de Datos");
        Curso c2 = new Curso("PR201", "Programación II");
        Curso c3 = new Curso("RD301", "Redes I");
        Curso c4 = new Curso("SO401", "Sistemas Operativos");
        Curso c5 = new Curso("AR501", "Arquitectura de Computadoras");

        universidad.agregarCurso(c1);
        universidad.agregarCurso(c2);
```



```
===== 1) Creaci□n de universidad, profesores y cursos =====

===== 3) Asignaci□n inicial de profesores a cursos =====
Asignaciones iniciales realizadas.

========================================
      SISTEMA UNIVERSIDAD - MEN□
========================================
1. Listar cursos con su profesor
2. Listar profesores con sus cursos
3. Cambiar profesor de un curso
4. Eliminar curso
5. Eliminar profesor
6. Mostrar reporte de cursos por profesor
0. Salir
========================================
Eleg□ una opci□n:
```

Listar cursos con su profesor y profesores con sus cursos.

```
===== 4) Listar cursos con su profesor =====
Cursos de la universidad Universidad Tecnol�gica:
Curso { C�digo='BD101', Nombre='Introducci�n a Bases de Datos', Profesor='Ana L�pez' }
Curso { C�digo='PR201', Nombre='Programaci�n II', Profesor='Carlos P�rez' }
Curso { C�digo='RD301', Nombre='Redes I', Profesor='Mar�a G�mez' }
Curso { C�digo='SO401', Nombre='Sistemas Operativos', Profesor='Carlos P�rez' }
Curso { C�digo='AR501', Nombre='Arquitectura de Computadoras', Profesor='Mar�a G�mez' }
```

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

```
===== 6) Eliminar curso y verificar sincronizaci�n =====
Ingres� el c�digo del curso a eliminar: PR201
Curso eliminado correctamente.
Cursos actuales de la universidad:
Cursos de la universidad Universidad Tecnol�gica:
Curso { C�digo='BD101', Nombre='Introducci�n a Bases de Datos', Profesor='Ana L�pez' }
Curso { C�digo='RD301', Nombre='Redes I', Profesor='Mar�a G�mez' }
Curso { C�digo='SO401', Nombre='Sistemas Operativos', Profesor='Carlos P�rez' }
Curso { C�digo='AR501', Nombre='Arquitectura de Computadoras', Profesor='Mar�a G�mez' }
```

Remover un profesor y dejar profesor = null,

```
===== 7) Eliminar profesor y dejar cursos sin profesor =====
Ingres� el ID del profesor a eliminar: P001
Profesor eliminado correctamente.
Cursos (para verificar que quedaron sin profesor si correspond�a):
Cursos de la universidad Universidad Tecnol�gica:
Curso { C�digo='BD101', Nombre='Introducci�n a Bases de Datos', Profesor='Sin profesor asignado' }
Curso { C�digo='RD301', Nombre='Redes I', Profesor='Mar�a G�mez' }
Curso { C�digo='SO401', Nombre='Sistemas Operativos', Profesor='Carlos P�rez' }
Curso { C�digo='AR501', Nombre='Arquitectura de Computadoras', Profesor='Mar�a G�mez' }
```

Mostrar un reporte: cantidad de cursos por profesor.

```
===== 8) Reporte: cantidad de cursos por profesor =====
===== Reporte: Cantidad de cursos por profesor =====
Ana L�pez: 1 curso(s)
Carlos P�rez: 2 curso(s)
Mar�a G�mez: 2 curso(s)
```

Link a Repositorio Github

https://github.com/MaquiMarinoni/TUPaD_P2-C22025.git