

PROGRAMACIÓN II

Trabajo Práctico 6: Colecciones y Sistema de Stock

Alumno:

Franco Sarrú

Link público de GitHub:

<https://github.com/fsarru/Programacion2.git>

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:

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

Métodos:

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

Enum CategoriaProducto Valores:

- ALIMENTOS
- ELECTRONICA
- ROPA
- HOGAR

Método adicional: java

public enum

CategoriaProducto {

ALIMENTOS("Productos comestibles"),

ELECTRONICA("Dispositivos electrónicos"),

ROPA("Prendas de vestir"),

HOGAR("Artículos para el hogar"); private

final String descripcion;

CategoriaProducto(String descripcion) {

this.descripcion = descripcion;

}

public String getDescripcion() { return

descripcion;

}

}

Clase Inventario

Atributo:

- **ArrayList<Producto> productos**

Métodos requeridos:

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

3. Tareas a realizar

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

CONCLUSIONES ESPERADAS

- Comprender el uso de **this** para acceder a atributos de instancia.
- Aplicar constructores sobrecargados para flexibilizar la creación de objetos.
- Implementar métodos con el mismo nombre y distintos parámetros.
- Representar objetos con **toString()** para mejorar la depuración.

- Diferenciar y aplicar atributos y métodos estáticos en Java.
- Reforzar el diseño modular y reutilizable mediante el paradigma orientado a objetos.

```
1 package colecciones;
2
3
4 public enum CategoriaProducto {
5     ALIMENTOS("Productos comestibles"),
6     ELECTRONICA("Dispositivos electrónicos y gadgets"),
7     ROPA("Prendas de vestir y accesorios"),
8     HOGAR("Artículos para la casa y decoración");
9
10    private final String descripcion;
11
12    CategoriaProducto(String descripcion) {
13        this.descripcion = descripcion;
14    }
15
16    public String getDescripcion() {
17        return descripcion;
18    }
19 }
```

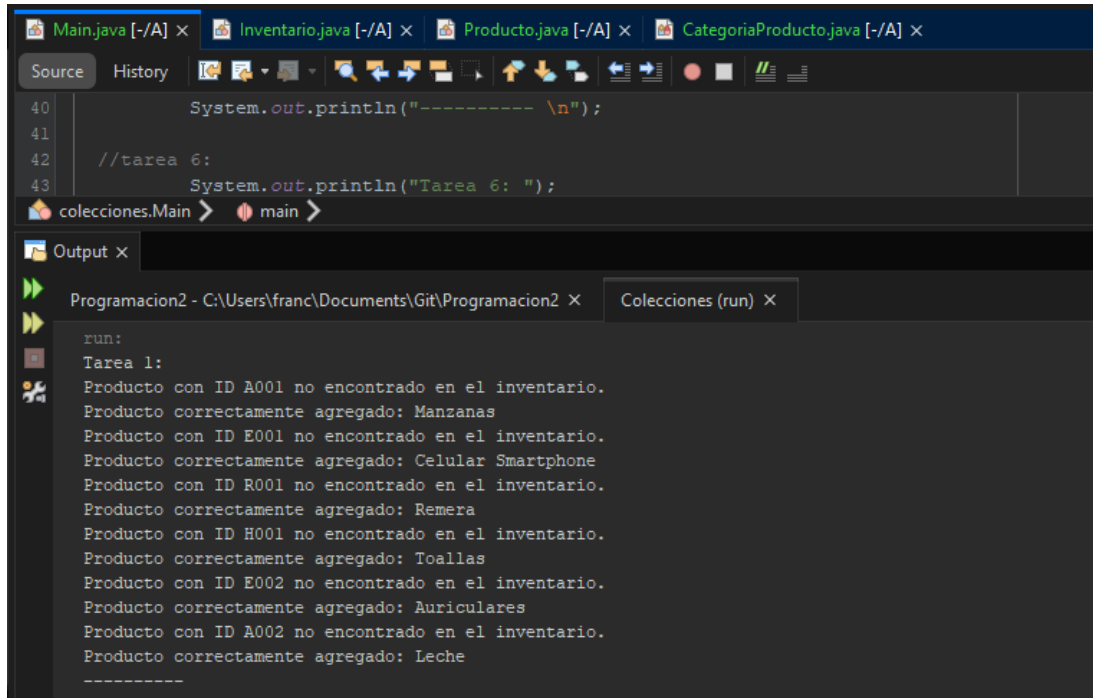
```
1 package colecciones;
2
3
4 public class Producto {
5     private String id;
6     private String nombre;
7     private double precio;
8     private int cantidad;
9     private CategoriaProducto categoria;
10
11    // Constructor
12    public Producto(String id, String nombre, double precio, int cantidad, CategoriaProducto categoria) {
13        this.id = id;
14        this.nombre = nombre;
15        this.precio = precio;
16        this.cantidad = cantidad;
17        this.categoria = categoria;
18    }
19
20    // Método para mostrar la información del producto
21    public void mostrarInfo() {
22        System.out.println("ID: " + id);
23        System.out.println("Nombre: " + nombre);
24        System.out.println("Precio: " + precio);
25        System.out.println("Stock: " + cantidad);
26        System.out.println("Categoria: " + categoria.name() + " (" + categoria.getDescripcion() + ")");
27    }
28 }
```

```
28
29 public String getNombre() {
30     return nombre;
31 }
32
33
34 public String getId() {
35     return id;
36 }
37
38 public int getCantidad() {
39     return cantidad;
40 }
41
42 public void setCantidad(int cantidad) {
43     this.cantidad = cantidad;
44 }
45
46 public CategoriaProducto getCategoria() {
47     return categoria;
48 }
49
50 public double getPrecio() {
51     return precio;
52 }
53
54 }
```

```
Main.java [-/A] x Inventario.java [-/A] x Producto.java [-/A] x CategoriaProducto.java [-/A] x
Source History
1 package colecciones;
2
3 import java.util.ArrayList;
4
5 public class Inventario {
6
7     private ArrayList<Producto> productos;
8
9     public Inventario() {
10         this.productos = new ArrayList<>();
11     }
12
13     public void agregarProducto(Producto p) {
14         if (buscarProductoPorId(p.getId()) == null) {
15             productos.add(p);
16             System.out.println("Producto correctamente agregado: " + p.getNombre());
17         } else {
18             System.out.println("Ya existe un producto con el ID " + p.getId());
19         }
20     }
21
22     public void listarProductos() {
23         if (productos.isEmpty()) {
24             System.out.println("El listado está vacío.");
25             return;
26         }
27         System.out.println("El listado cuenta con " + productos.size() + " productos en total");
28         for (Producto producto : productos) {
29             producto.mostrarInfo();
30         }
31     }
32
33
34     public Producto buscarProductoPorId(String id) {
35         int ubicacion = 0;
36         for (Producto producto : productos) {
37             if (producto.getId().equals(id)) {
38                 System.out.println("Producto encontrado ");
39                 return producto;
40             }
41             ubicacion++;
42         }
43         System.out.println("Producto con ID " + id + " no encontrado en el inventario.");
44         return null;
45     }
46 }
```



```
47
48     public void eliminarProducto(String id) {
49         Producto p = buscarProductoPorId(id);
50         if (p != null) {
51             productos.remove(p);
52             System.out.println("Fue eliminado el producto '" + p.getNombre() + "' con ID " + id);
53         } else {
54             System.out.println("No se encontró un producto con el ID " + id + ". No se pudo eliminar.");
55         }
56     }
57
58     public void actualizarStock(String id, int nuevaCantidad) {
59         Producto p = buscarProductoPorId(id);
60         if (p != null) {
61             p.setCantidad(nuevaCantidad);
62             System.out.println("Se actualizo el stock para " + p.getNombre() + ". Nuevo Stock: " + nuevaCantidad);
63         } else {
64             System.out.println("No se encontró un producto con el ID " + id + ".");
65         }
66     }
67
68     public ArrayList<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
69         ArrayList<Producto> filtrados = new ArrayList<>();
70         System.out.println("Productos por categoria: " + categoria.name());
71         for (Producto producto : productos) {
72             if (producto.getCategoria() == categoria) {
73                 filtrados.add(producto);
74                 System.out.println(producto.getNombre());
75             }
76         }
77         System.out.println("Total de productos de " + categoria.name() + " encontrados: " + filtrados.size());
78
79         return filtrados;
80     }
81
82
83
84     public int obtenerTotalStock() {
85         int total = 0;
86         for (Producto producto : productos) {
87             total += producto.getCantidad();
88         }
89         return total;
90
91
92     public Producto obtenerProductoConMayorStock() {
93         if (productos.isEmpty()) {
94             return null;
95         }
96         Producto mayorStock = productos.get(0);
97         for (Producto p : productos) {
98             if (p.getCantidad() > mayorStock.getCantidad()) {
99                 mayorStock = p;
100             }
101         }
102         return mayorStock;
103     }
104
105
106     public ArrayList<Producto> filtrarProductosPorPrecio(double min, double max) {
107         ArrayList<Producto> filtrados = new ArrayList<>();
108         for (Producto producto : productos) {
109             if (producto.getPrecio() >= min && producto.getPrecio() <= max) {
110                 filtrados.add(producto);
111             }
112         }
113         return filtrados;
114     }
115
116     public void mostrarCategoriasDisponibles() {
117         for (CategoriaProducto categoria : CategoriaProducto.values()) {
118             System.out.println(" - " + categoria.name() + ": " + categoria.getDescripcion());
119         }
120     }
121 }
```

The screenshot shows an IDE with four open files: `Main.java`, `Inventario.java`, `Producto.java`, and `CategoriaProducto.java`. The `Source` tab is active, showing lines 40 to 43 of `Main.java`:

```
40     System.out.println("----- \n");
41
42     //tarea 6:
43     System.out.println("Tarea 6: ");
```

The `Output` tab is also active, showing the execution results of the program:

```
run:
Tarea 1:
Producto con ID A001 no encontrado en el inventario.
Producto correctamente agregado: Manzanas
Producto con ID E001 no encontrado en el inventario.
Producto correctamente agregado: Celular Smartphone
Producto con ID R001 no encontrado en el inventario.
Producto correctamente agregado: Remera
Producto con ID H001 no encontrado en el inventario.
Producto correctamente agregado: Toallas
Producto con ID E002 no encontrado en el inventario.
Producto correctamente agregado: Auriculares
Producto con ID A002 no encontrado en el inventario.
Producto correctamente agregado: Leche
-----
```

```
Output x
Programacion2 - C:\Users\franc\Documents\Git\Programacion2 x  Colecciones (run) x

Tarea 2:
El listado cuenta con 6 productos en total
ID: A001
Nombre: Manzanas
Precio: 150.5
Stock: 150
Categoria: ALIMENTOS (Productos comestibles)
ID: E001
Nombre: Celular Smartphone
Precio: 8500.99
Stock: 45
Categoria: ELECTRONICA (Dispositivos electrónicos y gadgets)
ID: R001
Nombre: Remera
Precio: 2200.0
Stock: 70
Categoria: ROPA (Prendas de vestir y accesorios)
ID: H001
Nombre: Toallas
Precio: 1850.75
Stock: 90
Categoria: HOGAR (Artículos para la casa y decoración)
ID: E002
Nombre: Auriculares
Precio: 1250.0
Stock: 120
Categoria: ELECTRONICA (Dispositivos electrónicos y gadgets)
ID: A002
Nombre: Leche
Precio: 80.0
Stock: 200
Categoria: ALIMENTOS (Productos comestibles)
-----

Tarea 3:
Producto encontrado
-----
```

```
Output x
Programacion2 - C:\Users\franc\Documents\Git\Programacion2 x  Colecciones (run) x

Tarea 4:
Productos por categoria: ELECTRONICA
Celular Smartphone
Auriculares
Total de productos de ELECTRONICA encontrados: 2
-----

Tarea 5:
Producto encontrado
Fue eliminado el producto 'Auriculares' con ID E002
El listado cuenta con 5 productos en total
ID: A001
Nombre: Manzanas
Precio: 150.5
Stock: 150
Categoria: ALIMENTOS (Productos comestibles)
ID: E001
Nombre: Celular Smartphone
Precio: 8500.99
Stock: 45
Categoria: ELECTRONICA (Dispositivos electronicos y gadgets)
ID: R001
Nombre: Remera
Precio: 2200.0
Stock: 70
Categoria: ROPA (Prendas de vestir y accesorios)
ID: H001
Nombre: Toallas
Precio: 1850.75
Stock: 90
Categoria: HOGAR (Articulos para la casa y decoracion)
ID: A002
Nombre: Leche
Precio: 80.0
Stock: 200
Categoria: ALIMENTOS (Productos comestibles)
-----

Tarea 6:
Producto encontrado
Se actualizo el stock para Manzanas. Nuevo Stock: 300
-----
```

```
Tarea 7:  
  Stock Total del Inventario: 705 unidades.  
-----  
  
Tarea 8:  
ID: A001  
Nombre: Manzanas  
Precio: 150.5  
Stock: 300  
Categoria: ALIMENTOS (Productos comestibles)  
-----  
  
Tarea 9:  
Productos en rango de precio (2 encontrados):  
-----  
  
Tarea 10:  
  - ALIMENTOS: Productos comestibles  
  - ELECTRONICA: Dispositivos electrónicos y gadgets  
  - ROPA: Prendas de vestir y accesorios  
  - HOGAR: Artículos para la casa y decoración  
-----  
  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Nuevo Ejercicio Propuesto 2: Biblioteca y Libros

1. Descripción general

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

2. Clases a implementar **Clase Autor**

Atributos:

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

Métodos:

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

Clase Libro

Atributos:

- **isbn (String)** → Identificador único del libro.
- **titulo (String)** → Título del libro.
- **anioPublicacion (int)** → Año de publicación.
- **autor (Autor)** → Autor del libro.

Métodos:

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

Clase Biblioteca Atributo:

- **String nombre**
- **List<Libro> libros** → Colección de libros de la biblioteca.

Métodos requeridos:

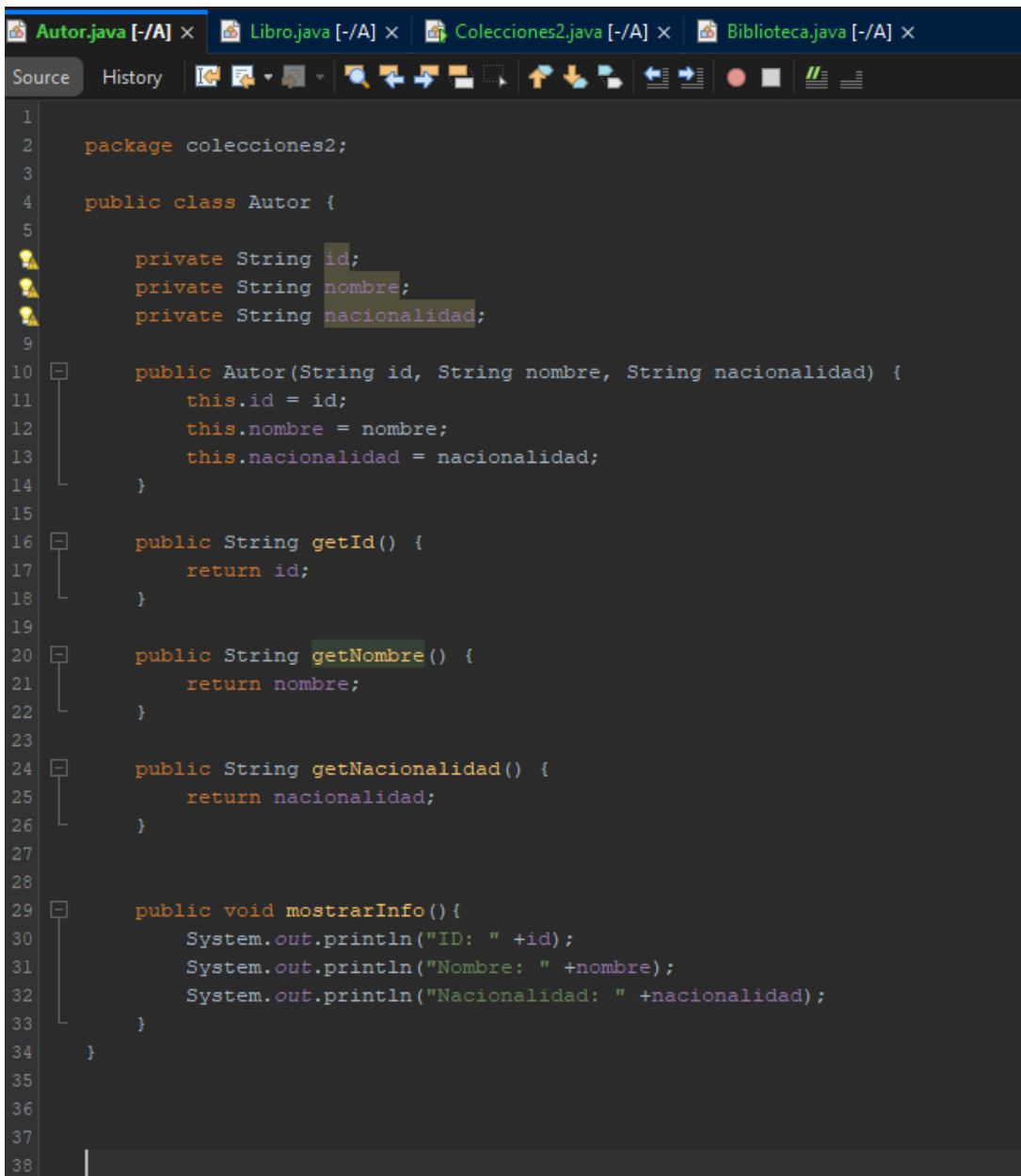
- **agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor)**
- **listarLibros()**
- **buscarLibroPorIsbn(String isbn)**
- **eliminarLibro(String isbn)**
- **obtenerCantidadLibros()**
- **filtrarLibrosPorAnio(int anio)**
- **mostrarAutoresDisponibles()**

3. Tareas a realizar

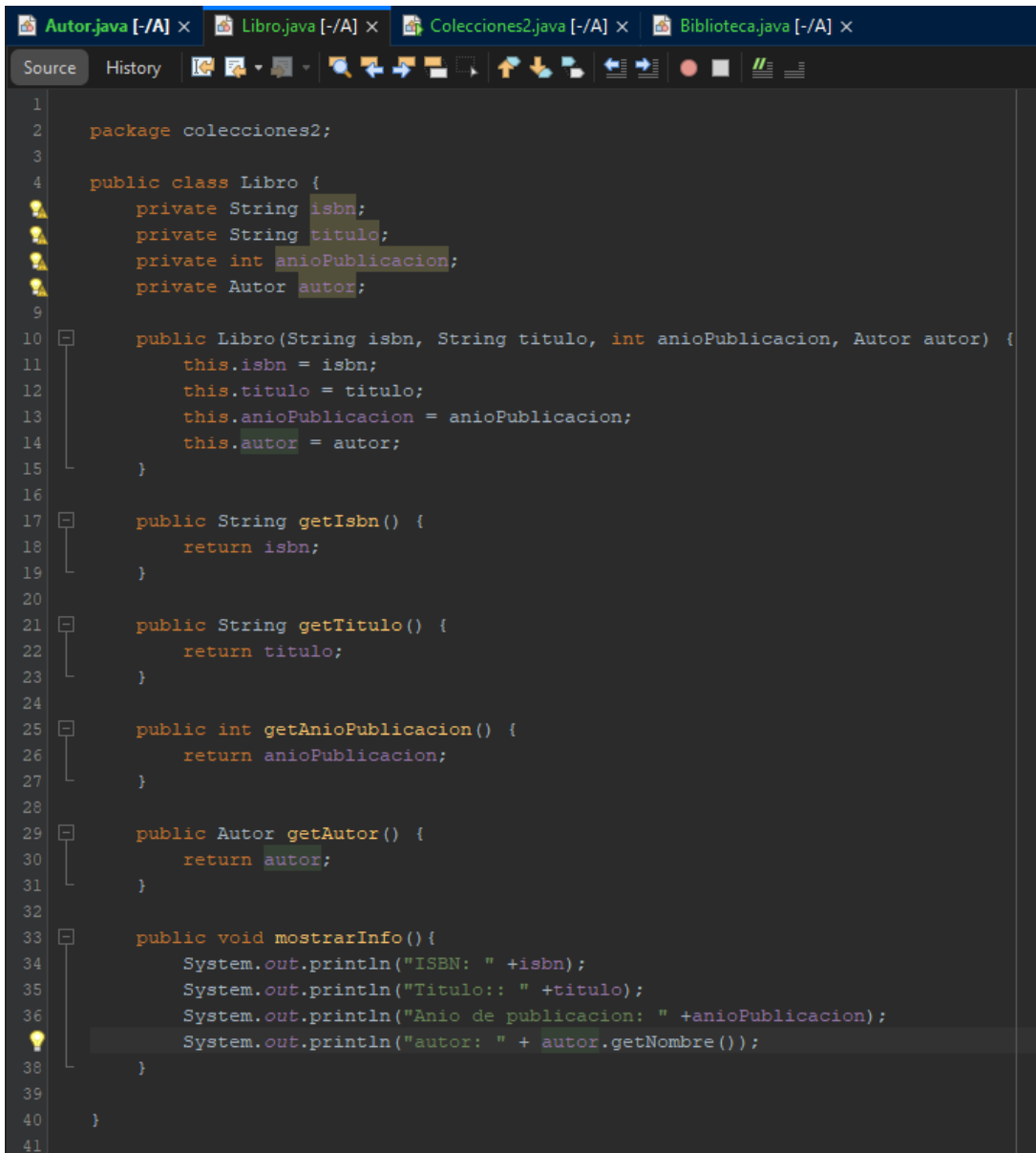
1. Creamos una biblioteca.
2. Crear al menos tres autores
3. Agregar 5 libros asociados a alguno de los Autores a la biblioteca.
4. Listar todos los libros con su información y la del autor.
5. Buscar un libro por su ISBN y mostrar su información.
6. Filtrar y mostrar los libros publicados en un año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

Conclusiones esperadas

- Comprender la **composición 1 a N** entre Biblioteca y Libro.
- Reforzar el manejo de **colecciones dinámicas** (ArrayList).
- Practicar el uso de **métodos de búsqueda, filtrado y eliminación**.
- Mejorar la modularidad aplicando el paradigma de **programación orientada a objetos**.



```
1 package colecciones2;
2
3
4 public class Autor {
5
6     private String id;
7     private String nombre;
8     private String nacionalidad;
9
10    public Autor(String id, String nombre, String nacionalidad) {
11        this.id = id;
12        this.nombre = nombre;
13        this.nacionalidad = nacionalidad;
14    }
15
16    public String getId() {
17        return id;
18    }
19
20    public String getNombre() {
21        return nombre;
22    }
23
24    public String getNacionalidad() {
25        return nacionalidad;
26    }
27
28
29    public void mostrarInfo() {
30        System.out.println("ID: " +id);
31        System.out.println("Nombre: " +nombre);
32        System.out.println("Nacionalidad: " +nacionalidad);
33    }
34 }
35
36
37
38
```

```
1 package colecciones2;
2
3
4 public class Libro {
5     private String isbn;
6     private String titulo;
7     private int anioPublicacion;
8     private Autor autor;
9
10    public Libro(String isbn, String titulo, int anioPublicacion, Autor autor) {
11        this.isbn = isbn;
12        this.titulo = titulo;
13        this.anioPublicacion = anioPublicacion;
14        this.autor = autor;
15    }
16
17    public String getIsbn() {
18        return isbn;
19    }
20
21    public String getTitulo() {
22        return titulo;
23    }
24
25    public int getAnioPublicacion() {
26        return anioPublicacion;
27    }
28
29    public Autor getAutor() {
30        return autor;
31    }
32
33    public void mostrarInfo() {
34        System.out.println("ISBN: " + isbn);
35        System.out.println("Titulo: " + titulo);
36        System.out.println("Anio de publicacion: " + anioPublicacion);
37        System.out.println("autor: " + autor.getNombre());
38    }
39
40 }
41
```

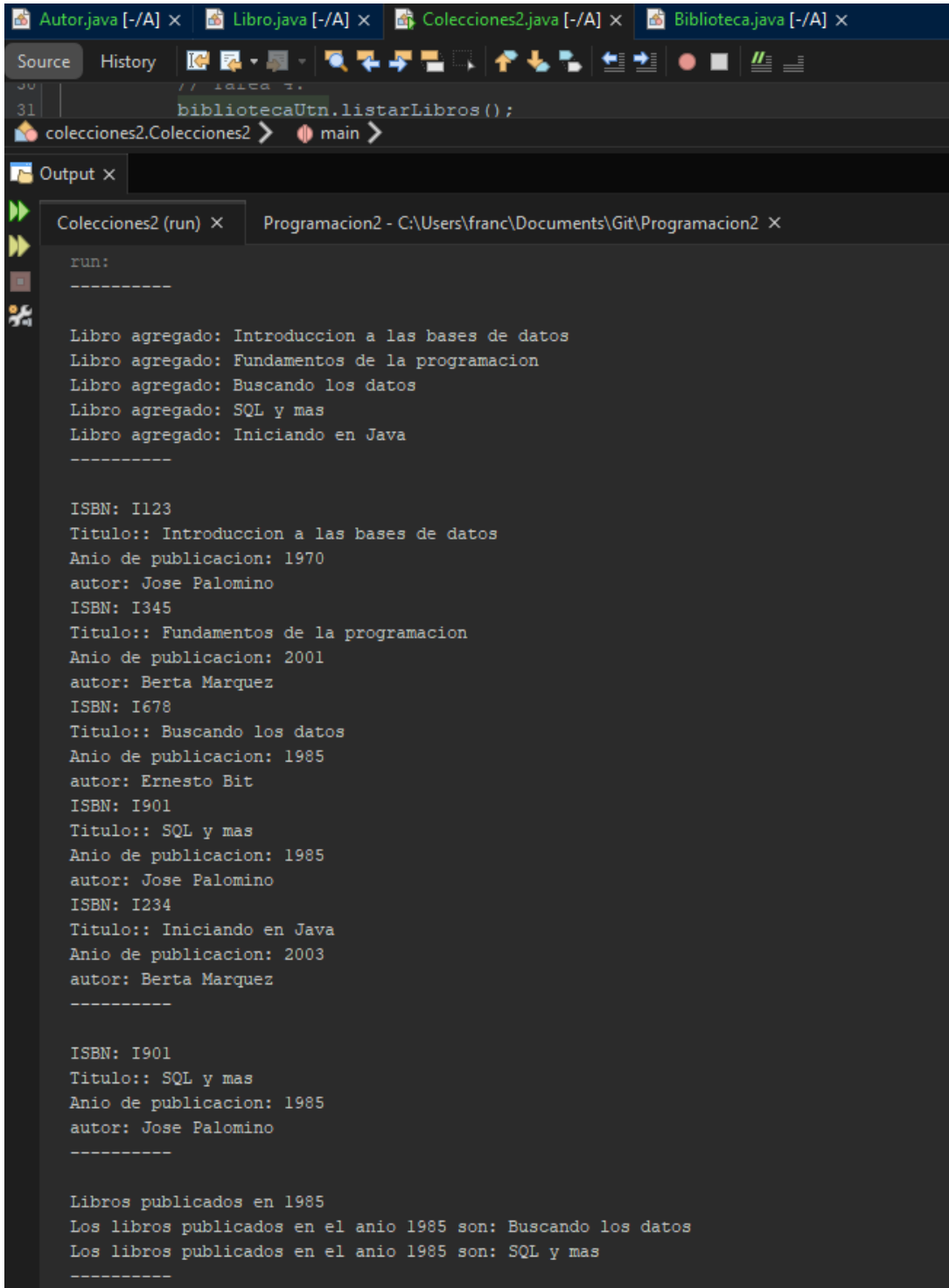
```
Autor.java [-/A] x Libro.java [-/A] x Colecciones2.java [-/A] x Biblioteca.java [-/A] x
Source History
1
2 package colecciones2;
3
4 import java.util.ArrayList;
5
6
7
8 public class Biblioteca {
9     private String nombre;
10    private ArrayList<Libro> libros;
11
12    public Biblioteca(String nombre) {
13        this.nombre = nombre;
14        this.libros = new ArrayList<>();
15    }
16
17    public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
18        boolean isbnExiste = false;
19
20        for (Libro libro : libros) {
21            if (libro.getIsbn().equals(isbn)) {
22                isbnExiste = true;
23            }
24        }
25        if (!isbnExiste) {
26            Libro nuevoLibro = new Libro(isbn, titulo, anioPublicacion, autor);
27            libros.add(nuevoLibro);
28            System.out.println("Libro agregado: " + titulo);
29        } else {
30            System.out.println("Ya existe un libro con el ISBN " + isbn);
31        }
32    }
33
34    public void listarLibros() {
35        if (libros.isEmpty()) {
36            System.out.println("La biblioteca está vacía.");
37        }
38        for (Libro libro : libros) {
39            libro.mostrarInfo();
40        }
41    }
42
```

```
Autor.java [-/A] x Libro.java [-/A] x Colecciones2.java [-/A] x Biblioteca.java [-/A] x
Source History
43 public Libro buscarLibroPorIsbn(String isbn) {
44     int ubicacion = 0;
45     for (Libro libro : libros) {
46         if (libro.getIsbn().equals(isbn)) {
47             libro.mostrarInfo();
48             return libro;
49         }
50         ubicacion++;
51     }
52     System.out.println("Libro con ISBN " + isbn + " no encontrado en la biblioteca.");
53     return null;
54 }
55
56
57 public void eliminarLibro(String isbn) {
58     Libro eliminar = buscarLibroPorIsbn(isbn);
59     if (eliminar != null) {
60         libros.remove(eliminar);
61         System.out.println("Libro " + eliminar.getTitulo() + " ELIMINADO.");
62         System.out.println("\nLos libros disponibles son los siguientes: \n");
63         listarLibros();
64     } else {
65         System.out.println("No se encontro un libro con el ISBN " + isbn + " para eliminar.");
66     }
67 }
68
69 public int obtenerCantidadLibros() {
70     System.out.println("La Biblioteca cuenta con " + libros.size() + " libros");
71     return libros.size();
72 }
73 }
74

73 }
74
75 public void filtrarLibrosPorAnio(int anio) {
76     System.out.println("Libros publicados en " + anio);
77     ArrayList<Libro> filtrados = new ArrayList<>();
78     for (Libro libro : libros) {
79         if (libro.getAnioPublicacion() == anio) {
80             filtrados.add(libro);
81         }
82     }
83     if (filtrados.isEmpty()) {
84         System.out.println(" No se encontraron libros publicados en el año " + anio + " en nuestra Biblioteca.");
85         return;
86     }
87     for (Libro libro : filtrados) {
88         System.out.println("Los libros publicados en el anio " + anio + " son: " + libro.getTitulo());
89     }
90 }
91
92 public void mostrarAutoresDisponibles() {
93     ArrayList<String> nombresAutoresUnicos = new ArrayList<>();
94
95     for (Libro libro : libros) {
96         String nombreAutor = libro.getAutor().getNombre();
97         if (!nombresAutoresUnicos.contains(nombreAutor)) {
98             nombresAutoresUnicos.add(nombreAutor);
99         }
100     }
101
102     System.out.println("Contamos con " + nombresAutoresUnicos.size() + " autores distintos en la biblioteca.");
103
104     for (String nombreAutor : nombresAutoresUnicos) {
105         System.out.println(nombreAutor);
106     }
107 }
108 }
109 }
```

```
Autor.java [-/A] × Libro.java [-/A] × Colecciones2.java [-/A] × Biblioteca.java [-/A] ×
Source History
1
2 package colecciones2;
3
4
5 public class Colecciones2 {
6
7
8     public static void main(String[] args) {
9
10
11         // Tarea 1.
12         Biblioteca bibliotecaUtn = new Biblioteca("Biblioteca UTN");
13
14         // Tarea 2.
15         Autor autor1 = new Autor("A001", "Jose Palomino", "Nicaragua");
16         Autor autor2 = new Autor("A002", "Berta Marquez", "Argentina");
17         Autor autor3 = new Autor("A003", "Ernesto Bit", "Groenlandia");
18         System.out.println("----- \n");
19
20
21         // Tarea 3.
22
23         bibliotecaUtn.agregarLibro("I123", "Introduccion a las bases de datos", 1970, autor1);
24         bibliotecaUtn.agregarLibro("I345", "Fundamentos de la programacion", 2001, autor2);
25         bibliotecaUtn.agregarLibro("I678", "Buscando los datos", 1985, autor3);
26         bibliotecaUtn.agregarLibro("I901", "SQL y mas", 1985, autor1);
27         bibliotecaUtn.agregarLibro("I234", "Iniciando en Java", 2003, autor2);
28         System.out.println("----- \n");
29
30         // Tarea 4.
31         bibliotecaUtn.listarLibros();
32         System.out.println("----- \n");
33
34
35         // Tarea 5.
36
37         bibliotecaUtn.buscarLibroPorIsbn("I901");
38         System.out.println("----- \n");
39
40         // Tarea 5.
41         bibliotecaUtn.filtrarLibrosPorAnio(1985);
42         System.out.println("----- \n");
43
44 }
```

```
43  
44     // Tarea 7.  
45     bibliotecaUtn.eliminarLibro("I678");  
46     System.out.println("----- \n");  
47  
48  
49     // Tarea 8.  
50     bibliotecaUtn.obtenerCantidadLibros();  
51     System.out.println("----- \n");  
52  
53     // Tarea 9.  
54     bibliotecaUtn.mostrarAutoresDisponibles();  
55  
56     }  
57 }  
58  
59
```



The screenshot shows an IDE with four tabs: `Autor.java`, `Libro.java`, `Colecciones2.java`, and `Biblioteca.java`. The `Source` view is active, showing line 31 of `Colecciones2.java` with the code `bibliotecaUtn.listarLibros();`. The `Output` view is also open, displaying the execution results of the `Colecciones2` class. The output is as follows:

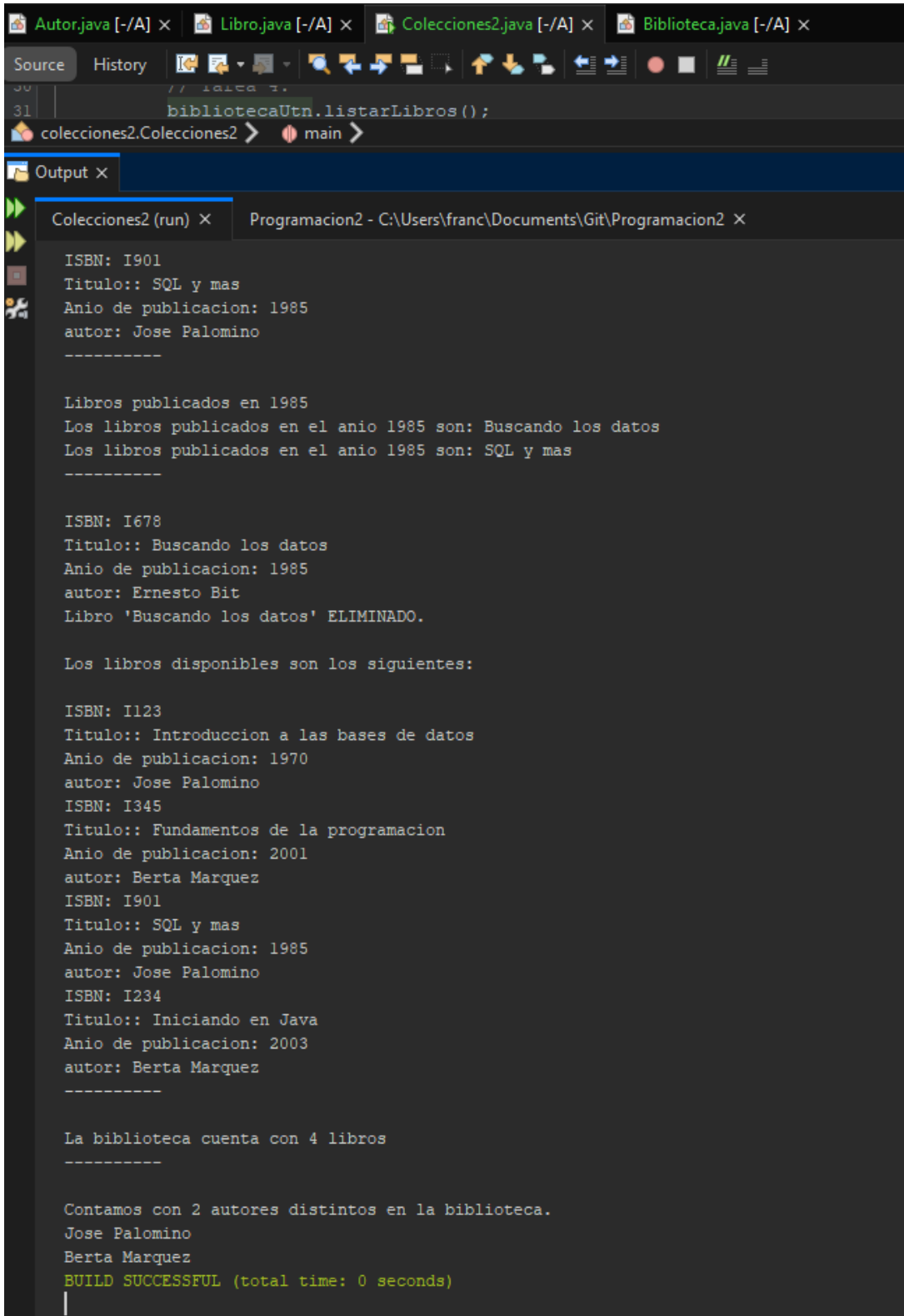
```
run:
-----

Libro agregado: Introduccion a las bases de datos
Libro agregado: Fundamentos de la programacion
Libro agregado: Buscando los datos
Libro agregado: SQL y mas
Libro agregado: Iniciando en Java
-----

ISBN: I123
Titulo:: Introduccion a las bases de datos
Anio de publicacion: 1970
autor: Jose Palomino
ISBN: I345
Titulo:: Fundamentos de la programacion
Anio de publicacion: 2001
autor: Berta Marquez
ISBN: I678
Titulo:: Buscando los datos
Anio de publicacion: 1985
autor: Ernesto Bit
ISBN: I901
Titulo:: SQL y mas
Anio de publicacion: 1985
autor: Jose Palomino
ISBN: I234
Titulo:: Iniciando en Java
Anio de publicacion: 2003
autor: Berta Marquez
-----

ISBN: I901
Titulo:: SQL y mas
Anio de publicacion: 1985
autor: Jose Palomino
-----

Libros publicados en 1985
Los libros publicados en el anio 1985 son: Buscando los datos
Los libros publicados en el anio 1985 son: SQL y mas
-----
```



The screenshot shows an IDE with four tabs: `Autor.java`, `Libro.java`, `Colecciones2.java`, and `Biblioteca.java`. The `Source` tab is active, showing line 31 of `Colecciones2.java` with the code `bibliotecaUtn.listarLibros();`. The `Output` tab is also open, displaying the execution results of the `Colecciones2` class. The output includes book details for ISBN 1901, ISBN I678, and ISBN I123, followed by a summary of the library's contents.

```
ISBN: I901
Titulo:: SQL y mas
Anio de publicacion: 1985
autor: Jose Palomino
-----

Libros publicados en 1985
Los libros publicados en el anio 1985 son: Buscando los datos
Los libros publicados en el anio 1985 son: SQL y mas
-----

ISBN: I678
Titulo:: Buscando los datos
Anio de publicacion: 1985
autor: Ernesto Bit
Libro 'Buscando los datos' ELIMINADO.

Los libros disponibles son los siguientes:

ISBN: I123
Titulo:: Introduccion a las bases de datos
Anio de publicacion: 1970
autor: Jose Palomino
ISBN: I345
Titulo:: Fundamentos de la programacion
Anio de publicacion: 2001
autor: Berta Marquez
ISBN: I901
Titulo:: SQL y mas
Anio de publicacion: 1985
autor: Jose Palomino
ISBN: I234
Titulo:: Iniciando en Java
Anio de publicacion: 2003
autor: Berta Marquez
-----

La biblioteca cuenta con 4 libros
-----

Contamos con 2 autores distintos en la biblioteca.
Jose Palomino
Berta Marquez
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

1. Descripción general

Se debe modelar un sistema académico donde **un Profesor dicta muchos Cursos** y cada **Curso** tiene exactamente **un Profesor responsable**. La relación **Profesor–Curso** es **bidireccional**:

- Desde **Curso** se accede a su **Profesor**.
 - Desde **Profesor** se accede a la **lista de Cursos** que dicta.
- Además, existe la clase **Universidad** que administra el alta/baja y consulta de profesores y cursos.

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:

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

Métodos sugeridos:

- **agregarCurso(Curso c)** → Agrega el curso a su lista si no está y sincroniza el lado del curso.
- **eliminarCurso(Curso c)** → Quita el curso y sincroniza el lado del curso (dejar `profesor` en `null` si corresponde).
- **listarCursos()** → Muestra códigos y nombres.
- **mostrarInfo()** → Imprime datos del profesor y cantidad de cursos.

Clase Curso

Atributos:

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

Métodos sugeridos:

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

Clase Universidad

Atributos:

- **String nombre**
- **List<Profesor> profesores**
- **List<Curso> cursos**

Métodos requeridos:

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

Tareas a realizar

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

7. Remover un profesor y dejar profesor = null,
8. 8. Mostrar un reporte: cantidad de cursos por profesor.

```
Colecciones3.java [-/A] x  Profesor.java [-/A] x  Curso.java [-/A] x  Universidad.java [-/A] x
Source  History  [Icons]
1  package colecciones3;
2
3  import java.util.ArrayList;
4
5
6  public class Profesor {
7      private String id;
8      private String nombre;
9      private String especialidad;
10     private ArrayList<Curso> cursos;
11
12     public Profesor(String id, String nombre, String especialidad) {
13         this.id = id;
14         this.nombre = nombre;
15         this.especialidad = especialidad;
16         this.cursos = new ArrayList<>();
17     }
18
19     public String getId() {
20         return id;
21     }
22
23     public String getNombre() {
24         return nombre;
25     }
26
27     public String getEspecialidad() {
28         return especialidad;
29     }
30
31     public ArrayList<Curso> getCursos() {
32         return cursos;
33     }
34
35
36     public void agregarCurso(Curso c) {
37         if (!cursos.contains(c)) {
38             cursos.add(c);
39             if (c.getProfesor() != this) {
40                 c.setProfesor(this);
41             }
42         }
43     }
44 }
```

Programación II

```
Colecciones3.java [-/A] x  Profesor.java [-/A] x  Curso.java [-/A] x  Universidad.java [-/A] x
Source  History  [Icons]
1
2  package colecciones3;
3
4  import java.util.ArrayList;
5
6
7  public class Universidad {
8      private String nombre;
9      private ArrayList<Profesor> profesores;
10     private ArrayList<Curso> cursos;
11
12     public Universidad(String nombre) {
13         this.nombre = nombre;
14         this.profesores = new ArrayList<>();
15         this.cursos = new ArrayList<>();
16     }
17
18     public void agregarProfesor(Profesor profesor) {
19         if (profesor != null && !profesores.contains(profesor)) {
20             profesores.add(profesor);
21         }
22     }
23
24     public void agregarCurso(Curso curso) {
25         if (curso != null && !cursos.contains(curso)) {
26             cursos.add(curso);
27         }
28     }
29
30     public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
31         Curso curso = buscarCursoPorCodigo(codigoCurso);
32         Profesor profesor = buscarProfesorPorId(idProfesor);
33         if (curso != null && profesor != null) {
34             curso.setProfesor(profesor);
35         }
36     }
37
```

```
38 public void listarProfesores() {
39     System.out.println("Lista de Profesores en " + nombre + ":");
40     if (profesores.isEmpty()) {
41         System.out.println(" Ningun profesor registrado.");
42     } else {
43         for (Profesor profesor : profesores) {
44             profesor.mostrarInfo();
45             System.out.println();
46         }
47     }
48 }
49
50 public void listarCursos() {
51     if (cursos.isEmpty()) {
52         System.out.println(" Ningun curso registrado.");
53     } else {
54         for (Curso curso : cursos) {
55             curso.mostrarInfo();
56         }
57     }
58 }
59
60 public Profesor buscarProfesorPorId(String id) {
61     for (Profesor profesor : profesores) {
62         if (profesor.getId().equals(id)) {
63             return profesor;
64         }
65     }
66     return null;
67 }
68
69 public Curso buscarCursoPorCodigo(String codigo) {
70     for (Curso curso : cursos) {
71         if (curso.getCodigo().equals(codigo)) {
72             return curso;
73         }
74     }
75     return null;
76 }
```

```
77
78 public void eliminarCurso(String codigo) {
79     Curso curso = buscarCursoPorCodigo(codigo);
80     if (curso == null) {
81         System.out.println("Curso no encontrado.");
82         return;
83     }
84     if (curso.getProfesor() != null) {
85         curso.setProfesor(null);
86     }
87     cursos.remove(curso);
88     System.out.println("Se elimino el curso: " + curso.getNombre() + " con codigo (" + codigo + ")");
89 }
90
91
92 public void eliminarProfesor(String id) {
93     Profesor profesor = buscarProfesorPorId(id);
94     if (profesor != null) {
95         for (Curso curso : new ArrayList<>(profesor.getCursos())) {
96             curso.setProfesor(null);
97         }
98         profesores.remove(profesor);
99     }
100 }
101
102 public void reporteCursosPorProfesor() {
103     System.out.println("Reporte: Cantidad de cursos por profesor en " + nombre + ":");
104     if (profesores.isEmpty()) {
105         System.out.println(" Ningun profesor registrado.");
106     } else {
107         for (Profesor profesor : profesores) {
108             System.out.println("  " + profesor.getNombre() + ": " + profesor.getCursos().size() + " cursos");
109         }
110     }
111 }
112 }
113
```

Programación II


```
44
45     // Tarea 4:
46
47     universidad.listarCursos();
48     System.out.println();
49     System.out.println("----- \n");
50     universidad.listarProfesores();
51     System.out.println("----- \n");
52
53     // Tarea 5:
54     universidad.asignarProfesorACurso("C001", "P002");
55     cursol.mostrarInfo();
56     prof1.listarCursos();
57     prof2.listarCursos();
58     System.out.println("----- \n");
59
60     // Tarea 6:
61     universidad.eliminarCurso("C005");
62     prof2.listarCursos();
63     universidad.listarCursos();
64     System.out.println("----- \n");
65
66     // Tarea 7:
67     universidad.eliminarProfesor("P003");
68     curso3.mostrarInfo();
69     universidad.listarProfesores();
70     System.out.println("----- \n");
71
72     // Tarea 8: Reporte de cursos por profesor
73     universidad.reporteCursosPorProfesor();
74     System.out.println("----- \n");
75 }
76 }
```

```
Output x
Programacion2 - C:\Users\franc\Documents\Git\Programacion2 x  Colecciones3 (run) x
run:
-----
Codigo: C001
Curso: Algebra I
Profesor: Ana Lopez
Codigo: C002
Curso: Programacion I
Profesor: Carlos Gomez
Codigo: C003
Curso: Fisica I
Profesor: Maria Torres
Codigo: C004
Curso: Algoritmos
Profesor: Ana Lopez
Codigo: C005
Curso: Programacion II
Profesor: Carlos Gomez
-----

Lista de Profesores en UTN:
ID: P001
Profesor: Ana Lopez
Especialidad: Matematicas
Cantidad de cursos: 2

ID: P002
Profesor: Carlos Gomez
Especialidad: Informatica
Cantidad de cursos: 2

ID: P003
Profesor: Maria Torres
Especialidad: Programacion
Cantidad de cursos: 1
-----
```

```
Output x
Programacion2 - C:\Users\franc\Documents\Git\Programacion2 x  Colecciones3 (run) x

Codigo: C001
Curso: Algebra I
Profesor: Carlos Gomez
Cursos dictados por Ana Lopez:
  - C004: Algoritmos
Cursos dictados por Carlos Gomez:
  - C002: Programacion I
  - C005: Programacion II
  - C001: Algebra I
-----

Se elimino el curso: Programacion II con codigo (C005)
Cursos dictados por Carlos Gomez:
  - C002: Programacion I
  - C001: Algebra I
Codigo: C001
Curso: Algebra I
Profesor: Carlos Gomez
Codigo: C002
Curso: Programacion I
Profesor: Carlos Gomez
Codigo: C003
Curso: Fisica I
Profesor: Maria Torres
Codigo: C004
Curso: Algoritmos
Profesor: Ana Lopez
-----

Codigo: C003
Curso: Fisica I
Profesor: Sin profesor
Lista de Profesores en UTN:
ID: P001
Profesor: Ana Lopez
Especialidad: Matematicas
Cantidad de cursos: 1

ID: P002
Profesor: Carlos Gomez
Especialidad: Informatica
Cantidad de cursos: 2

-----
```

```
-----  
Reporte: Cantidad de cursos por profesor en UTN:  
- Ana Lopez: 1 cursos  
- Carlos Gomez: 2 cursos  
-----  
  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Conclusiones esperadas

- Diferenciar **bidireccionalidad** de una relación unidireccional (navegación desde ambos extremos).
- Mantener **invariantes de asociación** (coherencia de referencias) al **agregar, quitar o reasignar**.
- Practicar colecciones (ArrayList), búsquedas y operaciones de alta/baja.
- Diseñar métodos “seguros” que **sincronicen** los dos lados siempre.