

PROGRAMACIÓN II

Trabajo Práctico 6: Colecciones y Sistema de Stock

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) Crear al menos cinco productos con diferentes categorías y agregarlos al inventario

```

20 public static void main(String[] args) {
21     mostrarMensaje("SISTEMA DE GESTION DE STOCK - INICIO");
22
23     // 1. CREAR INVENTARIO
24     Inventario inventario = new Inventario();
25
26     // 2. CREAR PRODUCTOS DE EJEMPLO
27     mostrarMensaje("CREACION DE PRODUCTOS");
28
29     Producto p1 = new Producto("E001", "Apple iPhone11", 899.00, 100, CategoriaProducto.ALIMENTOS);
30     Producto p2 = new Producto("E001", "Smartphone Samsung", 1250.00, 25, CategoriaProducto.ELECTRONICA);
31     Producto p3 = new Producto("E001", "Pasta Arroz", 890.75, 10, CategoriaProducto.ROSA);
32     Producto p4 = new Producto("E001", "Jugo de Naranja", 1500.00, 15, CategoriaProducto.ROSA);
33     Producto p5 = new Producto("E002", "Lechuga de Chile", 600.00, 50, CategoriaProducto.ALIMENTOS);
34     Producto p6 = new Producto("E003", "Aguacates Inmaduros", 350.00, 30, CategoriaProducto.ELECTRONICA);
    
```

Output - Practic01 SistemaStock [out]

```

SISTEMA DE GESTION DE STOCK - INICIO
INVENTARIO CERRADO CORRECTAMENTE
CREACION DE PRODUCTOS
AGREGAR PRODUCTOS AL INVENTARIO
    
```

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

The screenshot shows an IDE with a project named 'Practico6 SistemaStock'. The 'Source' tab is active, displaying the following Java code in 'CategoriaProducto.java':

```

// 3. AGREGAR PRODUCTOS AL INVENTARIO
mostrarSeparador("AGREGAR PRODUCTOS AL INVENTARIO");

inventario.agregarProducto(p1);
inventario.agregarProducto(p2);
inventario.agregarProducto(p3);
inventario.agregarProducto(p4);
inventario.agregarProducto(p5);
inventario.agregarProducto(p6);

// 4. LISTAR TODOS LOS PRODUCTOS
mostrarSeparador("LISTA COMPLETA DE PRODUCTOS");
inventario.listaProductos();
    
```

The 'Output' window at the bottom shows the result of running the program:

```

LISTA COMPLETA DE PRODUCTOS (if productos)
=====
I. =====
ID: A01
Nombre: Arroz Integral
Precio: $888.8
Stock: 100 unidades
Categoría: Productos comestibles
    
```

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

The screenshot shows the same IDE with the 'MainStock.java' file selected in the 'Source' tab. The code is as follows:

```

// 5. BUSCAR PRODUCTO POR ID
mostrarSeparador("BUSQUEDA DE PRODUCTO POR ID (E001)");
Producto encontrado = inventario.buscarProductoPorId("E001");
if (encontrado != null) {
    System.out.println(" Producto encontrado:");
    encontrado.mostrarInfo();
} else {
    System.out.println(" Producto no encontrado");
}
    
```

The 'Output' window shows the result of running the program:

```

=====
BUSQUEDA DE PRODUCTO POR ID (E001)
=====
Producto encontrado:
=====
ID: E001
Nombre: Smartphone Samsung
Precio: $1250.0
Stock: 25 unidades
Categoría: Dispositivos electronicos
    
```


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

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Practico6 SistemaStock' with source packages 'practico6.sistemastock' containing files 'CategoriaProducto.java', 'Inventario.java', 'MainStock.java', and 'Practico6SistemaStock.java'.
- Source Editor:** Displays the code for 'MainStock.java' with the following snippet:


```

59
60 // 6. FILTRAR POR CATEGORÍA
61 mostrarSeparador("FILTRAR POR CATEGORIA: ALIMENTOS");
62 inventario.filtrarPorCategoria(CategoriaProducto.ALIMENTOS);
63
64
65
66
67
68
      
```
- Output Console:** Shows the execution results for 'Practico6 SistemaStock (run)':


```

=====
FILTRAR POR CATEGORIA: ALIMENTOS
=====
PRODUCTOS EN CATEGORIA: Productos comestibles
=====
ID: A001
Nombre: Arroz Integral
Precio: $350.5
Stock: 100 unidades
Categoria: Productos comestibles
=====
ID: A002
Nombre: Aceite de Oliva
      
```

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

The screenshot shows the same IDE with the following components:

- Project Explorer:** Similar to the previous screenshot, but with 'Producto.java' also visible in the source packages.
- Source Editor:** Displays the code for 'MainStock.java' with the following snippet:


```

63
64 // 7. ELIMINAR UN PRODUCTO
65 mostrarSeparador("ELIMINAR PRODUCTO (R001)");
66 inventario.eliminarProducto("R001");
67 inventario.listarProductos();
68
69
70
71
72
73
74
      
```
- Output Console:** Shows the execution results for 'Practico6 SistemaStock (run)':


```

=====
ELIMINAR PRODUCTO (R001)
=====
Producto eliminado: Jeans Azul
      
```

6) Actualizar el stock de un producto existente.

```

62
63
64 // 4. ACTUALIZAR STOCK
65 mostrarSeparador("ACTUALIZAR STOCK DE ARROZ (A001)");
66 inventario.actualizarStock("A001",150);
67
68 // 5. MOSTRAR TOTAL DE STOCK
69 mostrarSeparador("TOTAL DE STOCK DISPONIBLE");
70 System.out.println(" Total de unidades en stock: " + inventario.obtenerTotalStock());
71
72

```

Output - Practico6 SistemaStock (run)

```

=====
ACTUALIZAR STOCK DE ARROZ (A001)
Stock actualizado: Arroz Integral -> 150 unidades
=====
TOTAL DE STOCK DISPONIBLE
Total de unidades en stock: 300

```

7) Mostrar el total de stock disponible.

```

73
74 // 5. MOSTRAR TOTAL DE STOCK
75 mostrarSeparador("TOTAL DE STOCK DISPONIBLE");
76 System.out.println(" Total de unidades en stock: " + inventario.obtenerTotalStock());
77
78
79
80
81
82
83

```

Output - Practico6 SistemaStock (run)

```

=====
TOTAL DE STOCK DISPONIBLE
Total de unidades en stock: 300

```

8) Obtener y mostrar el producto con mayor stock.

```

73
74
75
76
77 // 10. PRODUCTO CON MAYOR STOCK
78 mostrarSeparador("PRODUCTO CON MAYOR STOCK");
79 Producto mayorStock = inventario.obtenerProductoConMayorStock();
80 if (mayorStock != null) {
81     System.out.println(" Producto con mayor stock:");
82     mayorStock.mostrarInfo();
83 }

```

Output - Practico6 SistemaStock (run)

```

=====
PRODUCTO CON MAYOR STOCK
=====
Producto con mayor stock:
=====
ID: A001
Nombre: Arroz Integral
Precio: $350.5
Stock: 150 unidades
Categoria: Productos comestibles
=====

```

9) Filtrar productos con precios

The screenshot shows an IDE with the following components:

- Projects:** JavaApp
- Source Packages:** practico6.sistemastock
 - CategoriaProducto.java
 - Inventario.java
 - MainStock.java (selected)
 - Practico6SistemaStock.java
 - Producto.java
- Code Editor:**

```

80  if (mayorStock != null) {
81
82
83  // 11. FILTRAR POR PRECIO
84  mostrarSeparador("FILTRAR PRODUCTOS ENTRE $500 Y $1200");
85  inventario.filtrarProductosPorPrecio(500, 1200);
86
87
88  // 12. ORDENAR PRODUCTOS
89  mostrarSeparador("ORDENAR PRODUCTOS POR PRECIO");
90  inventario.ordenarProductosPorPrecio();
91  inventario.listarProductos();
92
93
94
95

```
- Output - Practico6 SistemaStock (run):**

```

>>> FILTRAR PRODUCTOS ENTRE $500 Y $1200
=====
>>> PRODUCTOS ENTRE $500.0 Y $1200.0
=====
ID: A002
Nombre: Aceite de Oliva
Precio: $680.0
Stock: 80 unidades
Categoria: Productos comestibles

```

10) . Mostrar las categorías disponibles con sus descripciones

The screenshot shows an IDE with the following components:

- Projects:** JavaApp
- Source Packages:** practico6.sistemastock
 - CategoriaProducto.java
 - Inventario.java
 - MainStock.java (selected)
 - Practico6SistemaStock.java
 - Producto.java
- Code Editor:**

```

85
86
87
88
89 // 14. MOSTRAR CATEGORÍAS DISPONIBLES
90 mostrarSeparador("CATEGORÍAS DISPONIBLES");
91 CategoriaProducto.mostrarTodasCategorias();
92
93
94
95

```
- Output:**

```

>>> CATEGORÍAS DISPONIBLES
=====
>>> SISTEMA FINALIZADO - TODAS LAS PRUEBAS COMPLETADAS

```


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.

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. Creamos una biblioteca.

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Practico0SistemaBiblioteca' with source packages 'Autor.java', 'Biblioteca.java', 'Libro.java', and 'MainBiblioteca.java'.
- Source Editor:** Displays the code for 'Biblioteca.java'. The code includes imports for 'ArrayList' and 'List', and defines a 'Biblioteca' class with a private 'nombre' field, a private 'libros' field (a list of 'Libro' objects), and a constructor that initializes the 'nombre' and 'libros' fields. The constructor also prints a confirmation message.
- Output Console:** Shows the output of the program, which includes the message 'SISTEMA DE GESTION DE BIBLIOTECA - INICIO' and 'Biblioteca 'Biblioteca Central UDLA' creada correctamente'.

2. Crear al menos tres autores

The screenshot shows an IDE with a project named 'Practico6SistemaBiblioteca'. The source code in 'MainBiblioteca.java' creates three authors: Gabriel Garcia Marquez (Colombiana), Julio Cortazar (Argentina), and Isabel Allende (Chilena). The output window shows the execution results, confirming the creation of the authors and displaying their details.

```

// 1. CREAR AUTORES (3 autores como solicita el ejercicio)
mostrarSeparador("CREACION DE AUTORES");

Autor autor1 = new Autor("A001", "Gabriel Garcia Marquez", "Colombiana");
Autor autor2 = new Autor("A002", "Julio Cortazar", "Argentina");
Autor autor3 = new Autor("A003", "Isabel Allende", "Chilena");

System.out.println("Autores creados correctamente");
autor1.mostrarInfo();
autor2.mostrarInfo();
autor3.mostrarInfo();
    
```

Output - Practico6SistemaBiblioteca (run)

```

=====
CREACION DE AUTORES
=====
Autores creados correctamente:
=====
ID: A001
Nombre: Gabriel Garcia Marquez
Nacionalidad: Colombiana
=====
ID: A002
Nombre: Julio Cortazar
Nacionalidad: Argentina
    
```

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

The screenshot shows the same IDE with updated code in 'MainBiblioteca.java' to add five books to the library. The books are associated with the authors created in the previous step. The output window shows the execution results, confirming the addition of the books and displaying their details.

```

// 2. AGREGAR 5 LIBROS (como solicita el ejercicio)
mostrarSeparador("AGREGAR LIBROS A LA BIBLIOTECA");

biblioteca.agregarLibro("ISBN-001", "Cien años de soledad", 1967, autor1);
biblioteca.agregarLibro("ISBN-002", "Rayuela", 1963, autor2);
biblioteca.agregarLibro("ISBN-003", "La casa de los espiritus", 1982, autor3);
biblioteca.agregarLibro("ISBN-004", "Cronica de una muerte anunciada", 1981, autor1);
biblioteca.agregarLibro("ISBN-005", "Bestiario", 1981, autor2);

// 3. LISTAR TODOS LOS LIBROS CON SU INFORMACION
mostrarSeparador("LISTA COMPLETA DE LIBROS");
    
```

Output - Practico6SistemaBiblioteca (run)

```

=====
AGREGAR LIBROS A LA BIBLIOTECA
=====
Libro agregado: Cien años de soledad
Libro agregado: Rayuela
Libro agregado: La casa de los espiritus
Libro agregado: Cronica de una muerte anunciada
Libro agregado: Bestiario
    
```

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

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Practico6SistemaBiblioteca' with a package 'practico6sistemabiblioteca' containing files: 'Autor.java', 'Biblioteca.java', 'Libro.java', 'MainBiblioteca.java', and 'Practico6SistemaBiblioteca.java'.
- Source Editor:** Displays the code for 'Practico6SistemaBiblioteca.java' with the following content:


```
// 4. LISTAR TODOS LOS LIBROS CON SU INFORMACIÓN
mostrarSeparador("LISTA COMPLETA DE LIBROS");
biblioteca.listarLibros();
```
- Output Console:** Shows the execution results for 'Practico6SistemaBiblioteca (run)'. The output is:


```
LISTA COMPLETA DE LIBROS
=====
CATALOGO COMPLETO - Biblioteca Central UNLa (5 libros)
=====
1. =====
ISBN: ISBN-001
Titulo: Cien años de soledad
Año: 1967
Autor: Gabriel Garcia Marquez
=====
2. =====
ISBN: ISBN-002
Titulo: Rayuela
Año: 1963
Autor: Julio Cortazar
```

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

The screenshot shows an IDE with the following components:

- Project Explorer:** Similar to the previous screenshot, showing the project structure.
- Source Editor:** Displays the code for 'Practico6SistemaBiblioteca.java' with the following content:


```
// 5. BUSCAR LIBRO POR ISBN
mostrarSeparador("BUSQUEDA DE LIBRO POR ISBN (ISBN-002)");
Libro libroEncontrado = biblioteca.buscarLibroPorIsbn("ISBN-002");
if (libroEncontrado != null) {
    System.out.println("Libro encontrado");
    libroEncontrado.mostrarInfo();
} else {
    System.out.println("Libro no encontrado");
}
```
- Output Console:** Shows the execution results for 'Practico6SistemaBiblioteca (run)'. The output is:


```
BUSQUEDA DE LIBRO POR ISBN (ISBN-002)
=====
Libro encontrado:
=====
ISBN: ISBN-002
Titulo: Rayuela
Año: 1963
Autor: Julio Cortazar
```

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

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Practico6SistemaBiblioteca' with source packages 'practico6sistemabiblioteca' containing files 'Autor.java', 'Biblioteca.java', 'Libro.java', 'MainBiblioteca.java', and 'Practico6SistemaBiblioteca.java'.
- Source Editor:** Displays the code for 'MainBiblioteca.java' with the following content:


```

83 // 6. FILTRAR LIBROS POR AÑO
84 public void filtrarLibrosPorAnio(int anio) {
85     System.out.println("\n LIBROS PUBLICADOS EN " + anio);
86     System.out.println("=".repeat(50));
87
88     boolean encontrado = false;
89     for (Libro libro : libros) {
90         if (libro.getAnioPublicacion() == anio) {
91             libro.mostrarInfo();
92             encontrado = true;
93         }
94     }
95
96     if (!encontrado) {
97         System.out.println("No hay libros publicados en " + anio);
98     }

```
- Output Console:** Shows the execution results for 'Practico6SistemaBiblioteca (run)':


```

=====
FILTRAR LIBROS DEL ANIO 1967
=====

LIBROS PUBLICADOS EN 1967
=====
ISBN: ISBN-001
Titulo: Cien años de soledad
Anio: 1967
Autor: Gabriel Garcia Marquez

```

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

The screenshot shows an IDE with the following components:

- Project Explorer:** Similar to the previous screenshot, showing the project structure.
- Source Editor:** Displays the code for 'MainBiblioteca.java' with the following content:


```

83 // 4. ELIMINAR LIBRO POR ISBN
84 public boolean eliminarLibro(String isbn) {
85     Libro libro = buscarLibroPorIsbn(isbn);
86     if (libro != null) {
87         libros.remove(libro);
88         System.out.println(" Libro eliminado: " + libro.getTitulo());
89         return true;
90     } else {
91         System.out.println(" No se encontro libro con ISBN: " + isbn);
92         return false;
93     }

```
- Output Console:** Shows the execution results for 'Practico6SistemaBiblioteca (run)':


```

ELIMINAR LIBRO (ISBN-003)
=====
Libro eliminado: La casa de los espíritus

```


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

```

77
78 // 5. OBTENER CANTIDAD DE LIBROS
79 public int obtenerCantidadLibros() {
80     return libros.size();
81 }
82
83
84
85
86
87
    
```

Output - Practico6SistemaBiblioteca (run)

```

=====
CANTIDAD TOTAL DE LIBROS
=====
Total de libros en biblioteca: 4
    
```

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

```

100 // 7. MOSTRAR AUTORES DISPONIBLES
101 public void mostrarAutoresDisponibles() {
102     System.out.println("\n AUTORES EN LA BIBLIOTECA");
103     System.out.println("\n".repeat(50));
104
105     if (libros.isEmpty()) {
106         System.out.println("No hay autores disponibles");
107         return;
108     }
109
110     // Se crea un ArrayList para almacenar los autores disponibles
111     List<String> autoresMostrados = new ArrayList<>();
112
113     for (Libro libro : libros) {
114         String autorInfo = libro.getAutor().getNombre() + " - " + libro.getAutor().getNacionalidad();
115         if (!autoresMostrados.contains(autorInfo)) {
116             System.out.println(autorInfo);
117             autoresMostrados.add(autorInfo);
118         }
119     }
120 }
    
```

Output - Practico6SistemaBiblioteca (run)

```

=====
AUTORES DISPONIBLES EN LA BIBLIOTECA
=====
AUTORES EN LA BIBLIOTECA
Gabriel Garcia Marquez - Colombiana
Julia Cortazar - Argentina
ESTADISTICAS DE BIBLIOTECA CENTRAL ONLA
    
```

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. Mostrar un reporte: cantidad de cursos por profesor.

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.

1. Crear al menos 3 profesores y 5 cursos.

```

19 public static void main(String[] args) {
20     mostrarSeparador("SISTEMA UNIVERSIDAD - RELACION BIDIRECCIONAL 1 A N");
21
22     // 1. CREAMS UNIVERSIDAD
23     Universidad uni = new Universidad("Universidad Nacional de Programación");
24
25     // 2. CREAM 3 PROFESORES Y 5 CURSOS (como siempre el ejercicio)
26     mostrarSeparador("CREACIÓN DE PROFESORES Y CURSOS");
27
28     // CREAM 3 profesores
29     Profesor prof1 = new Profesor("P001", "Dr. Carlos Garcia", "Matemáticas");
30     Profesor prof2 = new Profesor("P002", "Dra. Veronica Guirin", "Física");
31     Profesor prof3 = new Profesor("P003", "Dr. David Martinez", "Programación");
32
33     // CREAM 5 cursos
34     Curso curso1 = new Curso("MATE1", "Cálculo I");
35     Curso curso2 = new Curso("FIS101", "Física I");
36     Curso curso3 = new Curso("PROG101", "Programación II");
37     Curso curso4 = new Curso("MATE2", "Álgebra lineal");
38     Curso curso5 = new Curso("PROG201", "Estructuras de Datos");
39 }
    
```

Output - Practico6 SistemaUniversidad (run)

```

=====
SISTEMA UNIVERSIDAD - RELACION BIDIRECCIONAL 1 A N
=====
UNIVERSIDAD 'Universidad Nacional de Programación' CREAMA CORRECTAMENTE
=====
CREACION DE PROFESORES Y CURSOS
=====
Profesor creado: Dr. Carlos Garcia
Profesor creado: Dra. Veronica Guirin
Profesor creado: Dr. David Martinez
    
```

2. Agregar profesores y cursos a la universidad.

```

40 // 3. AGREGAR PROFESORES Y CURSOS A LA UNIVERSIDAD
41 mostrarSeparador("AGREGAR A LA UNIVERSIDAD");
42
43 uni.agregarProfesor(prof1);
44 uni.agregarProfesor(prof2);
45 uni.agregarProfesor(prof3);
46
47 uni.agregarCurso(curso1);
48 uni.agregarCurso(curso2);
49 uni.agregarCurso(curso3);
50 uni.agregarCurso(curso4);
51 uni.agregarCurso(curso5);
52
    
```

Output - Practico6 SistemaUniversidad (run)

```

=====
AGREGAR A LA UNIVERSIDAD
=====
Profesor agregado a la universidad: Dr. Carlos Garcia
    
```


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

```

Source History
31 uni.agregarCurso(cursos);
32
33 // 4. ASIGNAR PROFESORES A CURSOS
34 mostrarSeparador("ASIGNAR PROFESORES A CURSOS");
35
36 uni.asignarProfesorACurso("MAT101", "P001"); // Garcia - Calculo I
37 uni.asignarProfesorACurso("FIS101", "P002"); // Guirin - Fisica I
38 uni.asignarProfesorACurso("PROG001", "P003"); // Martinez - Programacion II
39 uni.asignarProfesorACurso("MAT201", "P001"); // Garcia - Algebra Lineal
40 uni.asignarProfesorACurso("PROG001", "P003"); // Martinez - Estructuras de Datos
41
Output - Practico6 SistemaUniversidad (run)
ASIGNAR PROFESORES A CURSOS
=====
Curso 'Calculo I' agregado al profesor Dr. Carlos Garcia
Asignacion exitosa: Dr. Carlos Garcia -> Calculo I.
Curso 'Fisica I' agregado al profesor Dra. Veronica Guirin

```

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

```

Source History
61
62 // 5. LISTAR CURSOS CON SU PROFESOR Y PROFESORES CON SUS CURSOS
63 mostrarSeparador("LISTA DE CURSOS CON PROFESORES");
64 uni.listarCursos();
65
66 mostrarSeparador("LISTA DE PROFESORES CON SUS CURSOS");
67 uni.listarProfesores();
68
69 // Mostrar cursos de cada profesor individualmente
70 System.out.println("\n CURSOS POR PROFESOR (DETALLE):");
71 prof1.listarCursos();
72 prof2.listarCursos();
73 prof3.listarCursos();
74
Output - Practico6 SistemaUniversidad (run)
LISTA DE CURSOS CON PROFESORES
=====

LISTA DE CURSOS - Universidad Nacional de Programacion
=====
1. =====
Codigo: MAT101
Nombre: Calculo I
Profesor: Dr. Carlos Garcia
=====

```

```

LISTA DE PROFESORES CON SUS CURSOS
=====

LISTA DE PROFESORES - Universidad Nacional de Programacion
=====
1. =====
ID: P001
Nombre: Dr. Carlos Garcia
Especialidad: Matematicas
Cursos asignados: 2

```


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

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Practico6 SistemaUniversidad' with source packages 'practico6.sistemauniversidad' containing files 'Curso.java', 'MainUniversidad.java', 'Practico6SistemaUniversidad.java', and 'Profesor.java'.
- Source Editor:** Displays the code for 'Practico6SistemaUniversidad.java' (lines 75-86). The code performs the following actions:
 - Comments: "// 6. CAMBIAR EL PROFESOR DE UN CURSO Y VERIFICAR SINCRONIZACION"
 - Method call: `mostrarSeparador("CAMBIAR PROFESOR DE CURSO (VERIFICAR SINCRONIZACION)")`
 - Print statement: `System.out.println("ANTES del cambio:");`
 - Method call: `curso1.mostrarInfoResumida();`
 - Method call: `prof1.listarCursos();`
 - Method call: `prof2.listarCursos();`
 - Print statement: `System.out.println("\n REALIZANDO CAMBIO: Calculo I de Garcia a Guirin...");`
 - Method call: `uni.asignarProfesorACurso("MAT101", "PROG");`
 - Print statement: `System.out.println("\nDESPUES del cambio:");`
- Output Console:** Shows the execution results for 'Practico6 SistemaUniversidad (run)':
 - Separator: `=====`
 - Header: `CAMBIAR PROFESOR DE CURSO (VERIFICAR SINCRONIZACION)`
 - Separator: `=====`
 - Print: `ANTES del cambio:`
 - Output: `MAT101 - Calculo I [Dr. Carlos Garcia]`
 - Print: `CURSOS DEL PROFESOR DR. CARLOS GARCIA`
 - Separator: `=====`
 - Output: `1. MAT101 - Calculo I`
 - Output: `2. MAT201 - Algebra lineal`
 - Print: `CURSOS DEL PROFESOR DRA. VERONICA GUIRIN`
 - Separator: `=====`
 - Output: `1. FIS101 - Fisica I`

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

The screenshot shows an IDE with the following components:

- Project Explorer:** Similar to the previous screenshot, but with 'Curso.java' selected.
- Source Editor:** Displays the code for 'Practico6SistemaUniversidad.java' (lines 91-103). The code performs the following actions:
 - Comments: "// 7. REMOVER UN CURSO Y CONFIRMAR QUE YA NO APARECE EN LA LISTA DEL PROFESOR"
 - Method call: `mostrarSeparador("ELIMINAR CURSO Y VERIFICAR SINCRONIZACION")`
 - Print statement: `System.out.println("ANTES de eliminar curso:");`
 - Method call: `curso3.mostrarInfoResumida();`
 - Method call: `prof3.listarCursos();`
 - Print statement: `System.out.println("\n ELIMINANDO CURSO: Programacion II...");`
 - Method call: `uni.eliminarCurso("PROG201");`
 - Print statement: `System.out.println("\n DESPUES de eliminar curso:");`
 - Method call: `prof3.listarCursos();`
- Output Console:** Shows the execution results for 'Practico6 SistemaUniversidad (run)':
 - Separator: `=====`
 - Header: `ELIMINAR CURSO Y VERIFICAR SINCRONIZACION`
 - Separator: `=====`
 - Print: `ANTES de eliminar curso:`
 - Output: `PROG201 - Programacion II [Dr. David Martinez]`
 - Print: `CURSOS DEL PROFESOR DR. DAVID MARTINEZ`
 - Separator: `=====`
 - Output: `1. PROG201 - Programacion II`
 - Output: `2. PROG301 - Estructuras de Datos`
 - Print: `ELIMINANDO CURSO: Programacion II...`

7. Remover un profesor y dejar profesor = null,

The screenshot shows an IDE with the following components:

- Projects:** JavaApp, Practico 2, Practico 3, Practico 4, Practico 5, Practico6SistemaBiblioteca, Practico6SistemaStock, Practico6SistemaUniversidad.
- Source Packages:** practico6.sistemauniversidad, Curso.java, MainUniversidad.java, Practico6SistemaUniversidad.java, Profesor.java, Universidad.java.
- Source Code (Practico6SistemaUniversidad.java):**

```

103 // 8. REMOVER UN PROFESOR Y DEJAR PROFESOR = NULL EN SUS CURSOS
104 mostrarSeparador("ELIMINAR PROFESOR Y VERIFICAR CURSOS SIN PROFESOR");
105
106 System.out.println("Cursos antes de eliminar profesor:");
107 uni.listarCursos();
108
109 System.out.println("\n ELIMINANDO PROFESOR: " + prof2.getNombre() + "...");
110 uni.eliminarProfesor("PROF");
111
112 System.out.println("\nCursos despues de eliminar profesor:");
113 uni.listarCursos();
114
115

```
- Output - Practico6SistemaUniversidad (run):**

```

=====
ELIMINAR PROFESOR Y VERIFICAR CURSOS SIN PROFESOR
=====
Cursos antes de eliminar profesor:
=====
LISTA DE CURSOS - Universidad Nacional de Programacion

```

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

The screenshot shows an IDE with the following components:

- Projects:** JavaApp, Practico 2, Practico 3, Practico 4, Practico 5, Practico6SistemaBiblioteca, Practico6SistemaStock, Practico6SistemaUniversidad.
- Source Packages:** practico6.sistemauniversidad, Curso.java, MainUniversidad.java, Practico6SistemaUniversidad.java, Profesor.java, Universidad.java.
- Source Code (Practico6SistemaUniversidad.java):**

```

116 // 9. MOSTRAR REPORTE: CANTIDAD DE CURSOS POR PROFESOR
117 mostrarSeparador("REPORTE FINAL: CANTIDAD DE CURSOS POR PROFESOR");
118 uni.mostrarReporteCursosPorProfesor();
119
120 // 10. ESTADISTICAS FINALES
121 mostrarSeparador("ESTADISTICAS FINALES DE LA UNIVERSIDAD");
122 uni.mostrarEstadisticas();
123
124 // VERIFICACIÓN EXTRA DE LA BIDIRECCIONALIDAD
125 mostrarSeparador("VERIFICACION DE RELACION BIDIRECCIONAL");
126
127 System.out.println("Desde Curso -> Profesor:");
128 curso04.mostrarInfoResumida();
129
130 System.out.println("\nDesde Profesor -> Cursos:");
131 prof01.listarCursos();
132
133 mostrarSeparador("SISTEMA UNIVERSIDAD - FINALIZADO CORRECTAMENTE");
134
135 }

```
- Output - Practico6SistemaUniversidad (run):**

```

=====
REPORTE FINAL: CANTIDAD DE CURSOS POR PROFESOR
=====
REPORTE: CURSOS POR PROFESOR

```