

## CASO PRÁCTICO 6

### PROGRAMACIÓN II

Profesor/es: Ramiro Huelpa

Alumno/a: Lauk, Karen

#### 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.

##### 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 (doble) → Precio del producto.
- cantidad (int) → Cantidad en stock.
- categoria (CategoriaProducto) → Categoría del producto.

###### Métodos:

- mostrar Info() → Muestra en consola la información del producto.

###### Código:

```

1 package colección_casol;
2 public class Producto {
3     private String id;
4     private String nombre;
5     private double precio;
6     private int cantidad;
7     private CategoriaProducto categoria;
8     // Constructor
9     public Producto(String id, String nombre, double precio, int cantidad, CategoriaProducto categoria) {
10         this.id = id;
11         this.nombre = nombre;
12         this.precio = precio;
13         this.cantidad = cantidad;
14         this.categoria = categoria;
15     }
16     // Método para mostrar la información del producto
17     public void mostrarInfo() {
18         System.out.println("ID: " + id +
19             " | Nombre: " + nombre +
20             " | Precio: $" + precio +
21             " | Cantidad: " + cantidad +
22             " | Categoría: " + categoria + " (" + categoria.getDescripcion() + ")");
23     }
24     // Getters y Setters básicos
25     public String getId() {
26         return id;
27     }
28     public double getPrecio() {
29         return precio;
30     }
31     public int getCantidad() {
32         return cantidad;
33     }
34     public CategoriaProducto getCategoria() {
35         return categoria;
36     }
37     public void setCantidad(int cantidad) {
38         this.cantidad = cantidad;
39     }
40 }

```

### ➤ Enum: CategoriaProducto

Valores:

- ALIMENTOS
- ELECTRÓNICA
- ROPA
- HOGAR

### Método adicional:

```

java public enum
CategoriaProducto {
    ALIMENTOS("Productos comestibles"),
    ELECTRÓNICA("Dispositivos electrónicos"),
    ROPA("Prendas de vestir"),
    HOGAR("Artículos para el hogar");
    private final String descripcion;
    Categoría Producto(String descripcion) {
        this.descripcion = descripcion;
    }
    public String getDescripcion() {
        return descripcion;
    }
}

```

## Código:

```
1 package colección_casol;
2 public enum CategoriaProducto {
3     ALIMENTOS("Productos comestibles"),
4     ELECTRONICA("Dispositivos electrónicos"),
5     ROPA("Prendas de vestir"),
6     HOGAR("Artículos para el hogar");
7     private final String descripcion;
8     CategoriaProducto(String descripcion) {
9         this.descripcion = descripcion;
10    }
11    public String getDescripcion() {
12        return descripcion;
13    }
14 }
```

### ➤ Clase Inventario

Atributo:

- ArrayList productos
- Métodos requeridos:
- agregarProducto(Producto p)
- listarProductos()
- buscarProductoPorId(String id)
- eliminarProducto(String id)
- actualizarStock(String id, int nuevaCantidad)
- filtrarPorCategoria(CategoriaProducto categoria)
- obtener Total Stock()
- obtenerProductoConMayorStock()
- filtrar Productos Por Precio(double min, double max)
- mostrar Categorías Disponibles()

## Código:

```
1 package colección_casol;
2 import java.util.ArrayList;
3 public class Inventario {
4     private ArrayList<Producto> productos;
5     public Inventario() {
6         productos = new ArrayList<>();
7     }
8     // Agregar producto
9     public void agregarProducto(Producto p) {
10        productos.add(p);
11    }
12    // Listar todos los productos
13    public void listarProductos() {
14        for (Producto p : productos) {
15            p.mostrarInfo();
16        }
17    }
18    // Buscar por ID
19    public Producto buscarProductoPorId(String id) {
20        for (Producto p : productos) {
21            if (p.getId().equalsIgnoreCase(id)) {
22                return p;
23            }
24        }
25        return null; // si no lo encuentra
26    }
27 }
```

```

27 // Eliminar producto
28 public void eliminarProducto(String id) {
29     Producto p = buscarProductoPorId(id);
30     if (p != null) {
31         productos.remove(p);
32         System.out.println("Producto eliminado correctamente.");
33     } else {
34         System.out.println("Producto no encontrado.");
35     }
36 }
37 // Actualizar stock
38 public void actualizarStock(String id, int nuevaCantidad) {
39     Producto p = buscarProductoPorId(id);
40     if (p != null) {
41         p.setCantidad(nuevaCantidad);
42         System.out.println("Stock actualizado correctamente.");
43     } else {
44         System.out.println("Producto no encontrado.");
45     }
46 }
47 // Filtrar por categoría
48 public void filtrarPorCategoria(CategoriaProducto categoria) {
49     for (Producto p : productos) {
50         if (p.getCategoria() == categoria) {
51             p.mostrarInfo();
52         }
53     }
54 }
55 // Obtener total de stock (sumar todas las cantidades)
56 public int obtenerTotalStock() {
57     int total = 0;
58     for (Producto p : productos) {
59         total += p.getCantidad();
60     }
61     return total;
62 }
63 // Producto con mayor stock
64 public Producto obtenerProductoConMayorStock() {
65     if (productos.isEmpty()) return null;
66     Producto mayor = productos.get(0);
67     for (Producto p : productos) {
68         if (p.getCantidad() > mayor.getCantidad()) {
69             mayor = p;
70         }
71     }
72     return mayor;
73 }

```

```

74 // Filtrar por rango de precios
75 public void filtrarProductosPorPrecio(double min, double max) {
76     for (Producto p : productos) {
77         if (p.getPrecio() >= min && p.getPrecio() <= max) {
78             p.mostrarInfo();
79         }
80     }
81 }
82 // Mostrar todas las categorías con descripción
83 public void mostrarCategoriasDisponibles() {
84     for (CategoriaProducto c : CategoriaProducto.values()) {
85         System.out.println(c + " → " + c.getDescripcion());
86     }
87 }
88 }

```

### 3. Tareas a realizar

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

```

6 // Crear productos
7 Producto p1 = new Producto("A1", "Pan", 500, 10, CategoriaProducto.ALIMENTOS);
8 Producto p2 = new Producto("E1", "Celular", 2500, 5, CategoriaProducto.ELECTRONICA);
9 Producto p3 = new Producto("R1", "Campera", 1500, 8, CategoriaProducto.ROPA);
10 Producto p4 = new Producto("H1", "Silla", 2000, 3, CategoriaProducto.HOGAR);
11 Producto p5 = new Producto("A2", "Leche", 700, 15, CategoriaProducto.ALIMENTOS);
12 // Agregarlos al inventario
13 inventario.agregarProducto(p1);
14 inventario.agregarProducto(p2);
15 inventario.agregarProducto(p3);
16 inventario.agregarProducto(p4);
17 inventario.agregarProducto(p5);

```

- b. Listar todos los productos mostrando su información y categoría.

```

18 // Listar productos
19 System.out.println("LISTA DE PRODUCTOS");
20 inventario.listarProductos();

```

- c. Buscar un producto por ID y mostrar su información.

```

21 // Buscar producto
22 System.out.println("\nBUSCAR PRODUCTO (A1)");
23 Producto buscado = inventario.buscarProductoPorId("A1");
24 if (buscado != null) buscado.mostrarInfo();

```

- d. Filtrar y mostrar productos que pertenezcan a una categoría específica.

```

25 // Filtrar por categoría
26 System.out.println("\nFILTRAR POR CATEGORÍA ALIMENTOS");
27 inventario.filtrarPorCategoria(CategoriaProducto.ALIMENTOS);

```

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

```

31 // Eliminar producto y volver a listar
32 System.out.println("\nELIMINAR PRODUCTO CON ID 'H1'");
33 inventario.eliminarProducto("H1");
34 System.out.println("\nLISTA ACTUALIZADA DE PRODUCTOS");
35 inventario.listarProductos();

```

f. Actualizar el stock de un producto existente.

```
36 // Total de stock
37 System.out.println("\nTOTAL DE STOCK");
38 System.out.println("Total: " + inventario.obtenerTotalStock());
```

g. Mostrar el total de stock disponible.

```
39 // Producto con mayor stock
40 System.out.println("\nPRODUCTO CON MAYOR STOCK");
41 Producto mayor = inventario.obtenerProductoConMayorStock();
42 if (mayor != null) mayor.mostrarInfo();
```

h. Obtener y mostrar el producto con mayor stock.

```
39 // Producto con mayor stock
40 System.out.println("\nPRODUCTO CON MAYOR STOCK");
41 Producto mayor = inventario.obtenerProductoConMayorStock();
42 if (mayor != null) mayor.mostrarInfo();
```

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

```
43 // Filtrar por precio
44 System.out.println("\nFILTRAR POR PRECIO ENTRE $1000 Y $3000");
45 inventario.filtrarProductosPorPrecio(1000, 3000);
```

j. Mostrar las categorías disponibles con sus descripciones.

```
46 // Categorías disponibles
47 System.out.println("\nCATEGORÍAS DISPONIBLES");
48 inventario.mostrarCategoriasDisponibles();
49 }
```

## Resultado:

Output - Colección\_caso1 (run)

```
FILTRAR POR PRECIO ENTRE $1000 Y $3000
ID: E1 | Nombre: Celular | Precio: $2500.0 | Cantidad: 5 | Categoría: ELECTRONICA (Dispositivos electrónicos)
ID: R1 | Nombre: Campera | Precio: $1500.0 | Cantidad: 20 | Categoría: ROPA (Prendas de vestir)

CATEGORÍAS DISPONIBLES
ALIMENTOS ? Productos comestibles
ELECTRONICA ? Dispositivos electrónicos
ROPA ? Prendas de vestir
HOGAR ? Artículos para el hogar
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Caso práctico 2

### 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:

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

**Código:**

```
1 package colecciones_caso2;
2 public class Autor {
3     private String id;
4     private String nombre;
5     private String nacionalidad;
6     // Constructor
7     public Autor(String id, String nombre, String nacionalidad) {
8         this.id = id;
9         this.nombre = nombre;
10        this.nacionalidad = nacionalidad;
11    }
12    // Mostrar información
13    public void mostrarInfo() {
14        System.out.println("Autor: " + nombre + " | ID: " + id + " | Nacionalidad: " + nacionalidad);
15    }
16    // Getters
17    public String getId() {
18        return id;
19    }
20    public String getNombre() {
21        return nombre;
22    }
23    public String getNacionalidad() {
24        return nacionalidad;
25    }
26 }
```

#### ➤ Clase Libro

Atributos:

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

Métodos:

- mostrar Info() → Muestra título, ISBN, año y autor.

**Código:**

```

1 package colecciones_caso2;
2 public class Libro {
3     private String isbn;
4     private String titulo;
5     private int anioPublicacion;
6     private Autor autor;
7     // Constructor
8     public Libro(String isbn, String titulo, int anioPublicacion, Autor autor) {
9         this.isbn = isbn;
10        this.titulo = titulo;
11        this.anioPublicacion = anioPublicacion;
12        this.autor = autor;
13    }
14    // Mostrar información
15    public void mostrarInfo() {
16        System.out.println("Titulo: " + titulo +
17            " | ISBN: " + isbn +
18            " | Año: " + anioPublicacion +
19            " | Autor: " + autor.getNombre() +
20            " (" + autor.getNacionalidad() + ")");
21    }
22    // Getters
23    public String getIsbn() {
24        return isbn;
25    }
26    public int getAnioPublicacion() {
27        return anioPublicacion;
28    }
29    public Autor getAutor() {
30        return autor;
31    }
32 }

```

### ➤ Clase Biblioteca

Atributo:

- String nombre
- List libros → Colección de libros de la biblioteca

Métodos requeridos:

- agregar Libro(String isbn, String titulo,int anio Publicación, Autor autor)
- listar Libros()
- buscar Libros Por Isbn(String isbn)
- eliminar Libro(String isbn)
- obtener Cantidad Libros()
- filtrar Libros Por Anio(int anio)
- mostrar Autores Disponibles()

**Código:**



```

1 package colecciones_caso2;
2 import java.util.ArrayList;
3 import java.util.List;
4 public class Biblioteca {
5     private String nombre;
6     private List<Libro> libros;
7     // Constructor
8     public Biblioteca(String nombre) {
9         this.nombre = nombre;
10        this.libros = new ArrayList<>();
11    }
12    // Agregar libro
13    public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
14        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
15    }
16    // Listar libros
17    public void listarLibros() {
18        if (libros.isEmpty()) {
19            System.out.println("No hay libros registrados.");
20            return;
21        }
22        for (Libro l : libros) {
23            l.mostrarInfo();
24        }
25    }
26    // Buscar libro por ISBN
27    public Libro buscarLibroPorIsbn(String isbn) {
28        for (Libro l : libros) {
29            if (l.getIsbn().equalsIgnoreCase(isbn)) {
30                return l;
31            }
32        }
33        return null;
34    }

```

```

35    // Eliminar libro
36    public void eliminarLibro(String isbn) {
37        Libro encontrado = buscarLibroPorIsbn(isbn);
38        if (encontrado != null) {
39            libros.remove(encontrado);
40            System.out.println("Libro eliminado correctamente.");
41        } else {
42            System.out.println("Libro no encontrado.");
43        }
44    }
45    // Cantidad total de libros
46    public int obtenerCantidadLibros() {
47        return libros.size();
48    }
49    // Filtrar por año
50    public void filtrarLibrosPorAnio(int anio) {
51        boolean encontrado = false;
52        for (Libro l : libros) {
53            if (l.getAnioPublicacion() == anio) {
54                l.mostrarInfo();
55                encontrado = true;
56            }
57        }
58        if (!encontrado) {
59            System.out.println("No se encontraron libros del año " + anio);
60        }
61    }

```

```

62 // Mostrar autores disponibles
63 public void mostrarAutoresDisponibles() {
64     System.out.println("Autores en la biblioteca:");
65     for (Libro l : libros) {
66         System.out.println("- " + l.getAutor().getNombre() + " (" + l.getAutor().getNacionalidad() + ")");
67     }
68 }
69 }
70

```

### 3. Tareas a realizar

a. Creamos una biblioteca.

```

4 // 1. Crear la biblioteca
5 Biblioteca biblioteca = new Biblioteca("Biblioteca Central");

```

b. Crear al menos tres autores

```

6 // 2. Crear autores
7 Autor a1 = new Autor("A1", "Gabriel García Márquez", "Colombia");
8 Autor a2 = new Autor("A2", "Jane Austen", "Reino Unido");
9 Autor a3 = new Autor("A3", "J.K. Rowling", "Reino Unido");

```

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

```

10 // 3. Agregar libros
11 biblioteca.agregarLibro("ISBN001", "Cien años de soledad", 1967, a1);
12 biblioteca.agregarLibro("ISBN002", "Orgullo y prejuicio", 1813, a2);
13 biblioteca.agregarLibro("ISBN003", "Harry Potter y la piedra filosofal", 1997, a3);
14 biblioteca.agregarLibro("ISBN004", "El amor en los tiempos del cólera", 1985, a1);
15 biblioteca.agregarLibro("ISBN005", "Harry Potter y el prisionero de Azkaban", 1999, a3);

```

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

```

16 // 4. Listar libros
17 System.out.println("LISTA DE LIBROS");
18 biblioteca.listarLibros();

```

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

```

19 // 5. Buscar por ISBN
20 System.out.println("\nBUSCAR LIBRO POR ISBN (ISBN003)");
21 Libro buscado = biblioteca.buscarLibroPorIsbn("ISBN003");
22 if (buscado != null) buscado.mostrarInfo();

```

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

```

23 // 6. Filtrar por año
24 System.out.println("\nLIBROS PUBLICADOS EN 1997");
25 biblioteca.filtrarLibrosPorAño(1997);

```

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

```

26 // 7. Eliminar libro
27 System.out.println("\nELIMINAR LIBRO (ISBN002)");
28 biblioteca.eliminarLibro("ISBN002");

```

```

29 // 8. Mostrar lista actualizada
30 System.out.println("\nLISTA ACTUALIZADA DE LIBROS");
31 biblioteca.listarLibros();

```

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

```

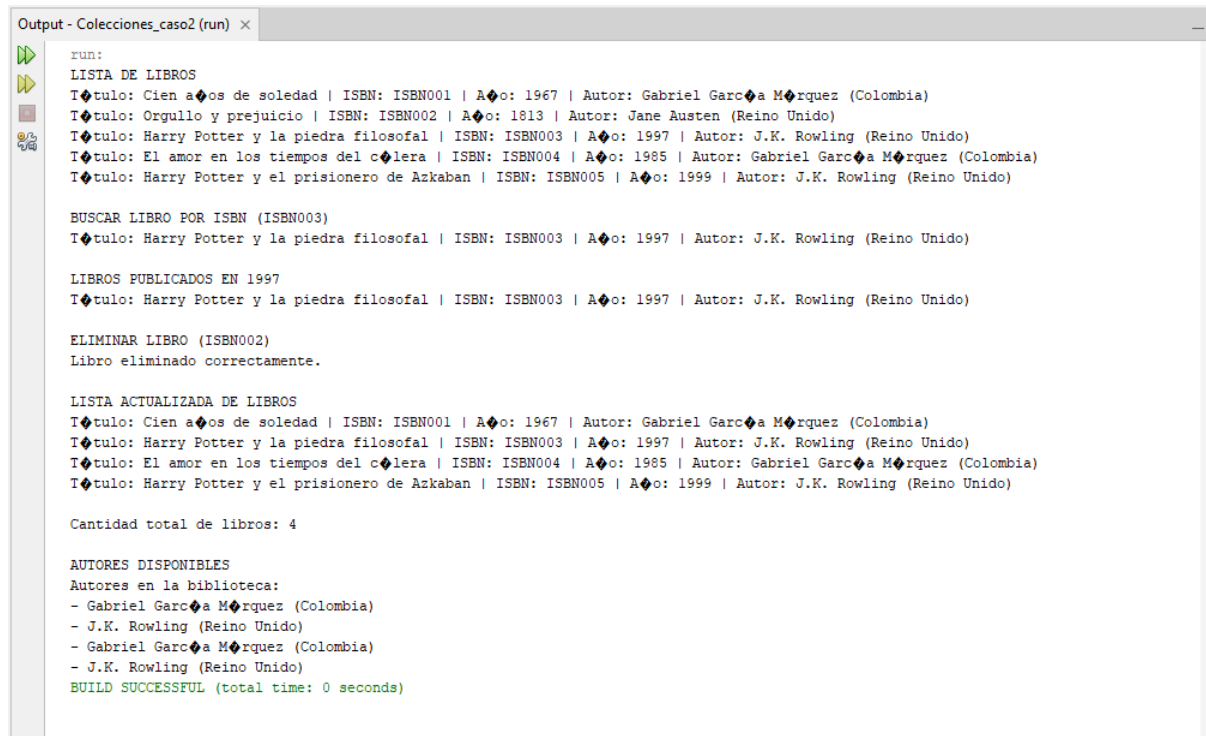
32 // 9. Mostrar cantidad total de libros
33 System.out.println("\nCantidad total de libros: " + biblioteca.obtenerCantidadLibros());

```

- i. Listar todos los autores de los libros disponibles en la biblioteca.

```
34 // 10. Mostrar autores disponibles
35     System.out.println("\nAUTORES DISPONIBLES");
36     biblioteca.mostrarAutoresDisponibles();
37 }
```

## Resultados:



```
Output - Colecciones_caso2 (run) x
run:
LISTA DE LIBROS
Título: Cien años de soledad | ISBN: ISBN001 | Año: 1967 | Autor: Gabriel García Márquez (Colombia)
Título: Orgullo y prejuicio | ISBN: ISBN002 | Año: 1813 | Autor: Jane Austen (Reino Unido)
Título: Harry Potter y la piedra filosofal | ISBN: ISBN003 | Año: 1997 | Autor: J.K. Rowling (Reino Unido)
Título: El amor en los tiempos del cólera | ISBN: ISBN004 | Año: 1985 | Autor: Gabriel García Márquez (Colombia)
Título: Harry Potter y el prisionero de Azkaban | ISBN: ISBN005 | Año: 1999 | Autor: J.K. Rowling (Reino Unido)

BUSCAR LIBRO POR ISBN (ISBN003)
Título: Harry Potter y la piedra filosofal | ISBN: ISBN003 | Año: 1997 | Autor: J.K. Rowling (Reino Unido)

LIBROS PUBLICADOS EN 1997
Título: Harry Potter y la piedra filosofal | ISBN: ISBN003 | Año: 1997 | Autor: J.K. Rowling (Reino Unido)

ELIMINAR LIBRO (ISBN002)
Libro eliminado correctamente.

LISTA ACTUALIZADA DE LIBROS
Título: Cien años de soledad | ISBN: ISBN001 | Año: 1967 | Autor: Gabriel García Márquez (Colombia)
Título: Harry Potter y la piedra filosofal | ISBN: ISBN003 | Año: 1997 | Autor: J.K. Rowling (Reino Unido)
Título: El amor en los tiempos del cólera | ISBN: ISBN004 | Año: 1985 | Autor: Gabriel García Márquez (Colombia)
Título: Harry Potter y el prisionero de Azkaban | ISBN: ISBN005 | Año: 1999 | Autor: J.K. Rowling (Reino Unido)

Cantidad total de libros: 4

AUTORES DISPONIBLES
Autores en la biblioteca:
- Gabriel García Márquez (Colombia)
- J.K. Rowling (Reino Unido)
- Gabriel García Márquez (Colombia)
- J.K. Rowling (Reino Unido)
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Caso práctico 3

### Universidad, Profesor y Curso (bidireccional 1 a N)

#### 1. Descripción general

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

- Desde Curso se accede a su Profesor.
- Desde Profesor se accede a la lista de Cursos que dicta.

Además, existe la clase Universidad que administra el alta/baja y consulta de profesores y cursos.

Invariante de asociación: cada vez que se asigne o cambie el profesor de un curso, debe actualizarse en los dos lados (agregar/quitar en la lista del profesor correspondiente).

#### 2. Clases a implementar

##### ➤ Clase Profesor

Atributos:

- id (String) → Identificador único.
- nombre (String) → Nombre completo.
- especialidad (String) → Área principal.

- List <curso> → 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.

Código:

```
1 package colecciones_caso3;
2 import java.util.ArrayList;
3 import java.util.List;
4 public class Profesor {
5     private String id;
6     private String nombre;
7     private String especialidad;
8     private List<Curso> cursos;
9     public Profesor(String id, String nombre, String especialidad) {
10         this.id = id;
11         this.nombre = nombre;
12         this.especialidad = especialidad;
13         this.cursos = new ArrayList<>();
14     }
15     // Agregar curso (sin duplicar y sincronizando)
16     public void agregarCurso(Curso c) {
17         if (!cursos.contains(c)) {
18             cursos.add(c);
19             if (c.getProfesor() != this) {
20                 c.setProfesor(this); // sincroniza el otro lado
21             }
22         }
23     }
24     // Eliminar curso (y romper la relación del otro lado)
25     public void eliminarCurso(Curso c) {
26         if (cursos.contains(c)) {
27             cursos.remove(c);
28             if (c.getProfesor() == this) {
29                 c.setProfesor(null);
30             }
31         }
32     }
33 }
```

```

33 // Listar cursos del profesor
34 public void listarCursos() {
35     if (cursos.isEmpty()) {
36         System.out.println(nombre + " no tiene cursos asignados.");
37     } else {
38         System.out.println("Cursos dictados por " + nombre + ":");
39         for (Curso c : cursos) {
40             System.out.println("- " + c.getCodigo() + " | " + c.getNombre());
41         }
42     }
43 }
44 // Mostrar info del profesor
45 public void mostrarInfo() {
46     System.out.println("Profesor: " + nombre + " | ID: " + id +
47         " | Especialidad: " + especialidad +
48         " | Cantidad de cursos: " + cursos.size());
49 }
50 // Getters
51 public String getId() {
52     return id;
53 }
54 public String getNombre() {
55     return nombre;
56 }
57 public List<Curso> getCursos() {
58     return cursos;
59 }
60 }

```

### ➤ Clase Curso

Atributos:

- código (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).

Código:

```

1 package colecciones_caso3;
2 public class Curso {
3     private String codigo;
4     private String nombre;
5     private Profesor profesor;
6     public Curso(String codigo, String nombre) {
7         this.codigo = codigo;
8         this.nombre = nombre;
9         this.profesor = null;
10    }
11    // Asignar profesor sincronizando ambos lados
12    public void setProfesor(Profesor p) {
13        // Si ya tenia profesor anterior, eliminarse de su lista
14        if (this.profesor != null) {
15            this.profesor.getCursos().remove(this);
16        }
17        this.profesor = p;
18        // Si hay nuevo profesor, agregar este curso a su lista
19        if (p != null && !p.getCursos().contains(this)) {
20            p.getCursos().add(this);
21        }
22    }

```

```

23 public void mostrarInfo() {
24     String nombreProfesor = (profesor != null) ? profesor.getNombre() : "Sin asignar";
25     System.out.println("Curso: " + nombre + " | Código: " + codigo + " | Profesor: " + nombreProfesor);
26 }
27 // Getters
28 public String getCodigo() {
29     return codigo;
30 }
31 public String getNombre() {
32     return nombre;
33 }
34 public Profesor getProfesor() {
35     return profesor;
36 }

```

## ➤ Clase Universidad

Atributos:

- String nombre
- List profesores
- List 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.

Código:

```

1 package colecciones_caso3;
2 import java.util.ArrayList;
3 import java.util.List;
4 public class Universidad {
5     private String nombre;
6     private List<Profesor> profesores;
7     private List<Curso> cursos;
8     public Universidad(String nombre) {
9         this.nombre = nombre;
10        this.profesores = new ArrayList<>();
11        this.cursos = new ArrayList<>();
12    }
13    public void agregarProfesor(Profesor p) {
14        profesores.add(p);
15    }
16    public void agregarCurso(Curso c) {
17        cursos.add(c);
18    }

```

```

19 public Profesor buscarProfesorPorId(String id) {
20     for (Profesor p : profesores) {
21         if (p.getId().equalsIgnoreCase(id)) {
22             return p;
23         }
24     }
25     return null;
26 }
27 public Curso buscarCursoPorCodigo(String codigo) {
28     for (Curso c : cursos) {
29         if (c.getCodigo().equalsIgnoreCase(codigo)) {
30             return c;
31         }
32     }
33     return null;
34 }

```

```

35 // Asignar profesor a curso
36 public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
37     Curso curso = buscarCursoPorCodigo(codigoCurso);
38     Profesor profesor = buscarProfesorPorId(idProfesor);
39
40     if (curso != null && profesor != null) {
41         curso.setProfesor(profesor);
42         System.out.println("Profesor " + profesor.getNombre() +
43             " asignado al curso " + curso.getNombre());
44     } else {
45         System.out.println("Error: profesor o curso no encontrado.");
46     }
47 }
48 public void listarProfesores() {
49     System.out.println("Profesores");
50     for (Profesor p : profesores) {
51         p.mostrarInfo();
52     }
53 }
54 public void listarCursos() {
55     System.out.println("Cursos");
56     for (Curso c : cursos) {
57         c.mostrarInfo();
58     }
59 }
60 public void eliminarCurso(String codigo) {
61     Curso c = buscarCursoPorCodigo(codigo);
62     if (c != null) {
63         if (c.getProfesor() != null) {
64             c.getProfesor().eliminarCurso(c);
65         }
66         cursos.remove(c);
67         System.out.println("Curso eliminado correctamente.");
68     } else {
69         System.out.println("Curso no encontrado.");
70     }
71 }

```

```

72 public void eliminarProfesor(String id) {
73     Profesor p = buscarProfesorPorId(id);
74     if (p != null) {
75         // Dejar profesor = null en todos los cursos que dictaba
76         for (Curso c : new ArrayList<>(p.getCursos())) {
77             c.setProfesor(null);
78         }
79         profesores.remove(p);
80         System.out.println("Profesor eliminado correctamente.");
81     } else {
82         System.out.println("Profesor no encontrado.");
83     }
84 }
85 // Reporte: cantidad de cursos por profesor
86 public void reporteCursosPorProfesor() {
87     System.out.println("\nReporte: Cursos por Profesor");
88     for (Profesor p : profesores) {
89         System.out.println(p.getNombre() + " → " + p.getCursos().size() + " cursos.");
90     }
91 }
92 }

```

## Tareas a realizar

- a. Crear al menos 3 profesores y 5 cursos.

```

5 // 1. Crear profesores
6 Profesor p1 = new Profesor("P1", "Laura Gómez", "Matemática");
7 Profesor p2 = new Profesor("P2", "Carlos Pérez", "Informática");
8 Profesor p3 = new Profesor("P3", "María Torres", "Física");
9 uni.agregarProfesor(p1);
10 uni.agregarProfesor(p2);
11 uni.agregarProfesor(p3);

```

```

12 // 2. Crear cursos
13 Curso c1 = new Curso("C1", "Álgebra");
14 Curso c2 = new Curso("C2", "Programación I");
15 Curso c3 = new Curso("C3", "Física Mecánica");
16 Curso c4 = new Curso("C4", "Bases de Datos");
17 Curso c5 = new Curso("C5", "Cálculo Diferencial");
18 uni.agregarCurso(c1);
19 uni.agregarCurso(c2);
20 uni.agregarCurso(c3);
21 uni.agregarCurso(c4);
22 uni.agregarCurso(c5);

```

- b. Agregar profesores y cursos a la universidad.
- c. Asignar profesores a cursos usando asignarProfesorACurso(...).

```

23 // 3. Asignar profesores a cursos
24 uni.asignarProfesorACurso("C1", "P1");
25 uni.asignarProfesorACurso("C5", "P1");
26 uni.asignarProfesorACurso("C2", "P2");
27 uni.asignarProfesorACurso("C4", "P2");
28 uni.asignarProfesorACurso("C3", "P3");

```



- d. Listar cursos con su profesor y profesores con sus cursos.

```
29 // 4. Listar cursos y profesores
30 System.out.println("\nLISTA DE CURSOS");
31 uni.listarCursos();
32 System.out.println("\nLISTA DE PROFESORES");
33 uni.listarProfesores();
```

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

```
34 // 5. Cambiar el profesor de un curso
35 System.out.println("\nCAMBIAR PROFESOR DEL CURSO C1");
36 uni.asignarProfesorACurso("C1", "P2");
```

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

```
37 // 6. Remover un curso
38 System.out.println("\nELIMINAR CURSO C5");
39 uni.eliminarCurso("C5");
```

- g. Remover un profesor y dejar profesor = null,

```
43 // 8. Mostrar reporte
44 uni.reporteCursosPorProfesor();
```

- h. Mostrar un reporte: cantidad de cursos por profesor.

```
45 // 9. Confirmar cambios
46 System.out.println("\nLISTA FINAL DE CURSOS");
47 uni.listarCursos();
48 System.out.println("\nLISTA FINAL DE PROFESORES");
49 uni.listarProfesores();
```

**Resultados:**

Output - Colecciones\_caso3 (run)



```
run:
Profesor Laura Gómez asignado al curso Álgebra
Profesor Laura Gómez asignado al curso Cálculo Diferencial
Profesor Carlos Pérez asignado al curso Programación I
Profesor Carlos Pérez asignado al curso Bases de Datos
Profesor María Torres asignado al curso Física Mecánica

LISTA DE CURSOS
Cursos
Curso: Álgebra | Código: C1 | Profesor: Laura Gómez
Curso: Programación I | Código: C2 | Profesor: Carlos Pérez
Curso: Física Mecánica | Código: C3 | Profesor: María Torres
Curso: Bases de Datos | Código: C4 | Profesor: Carlos Pérez
Curso: Cálculo Diferencial | Código: C5 | Profesor: Laura Gómez

LISTA DE PROFESORES
Profesores
Profesor: Laura Gómez | ID: P1 | Especialidad: Matemática | Cantidad de cursos: 2
Profesor: Carlos Pérez | ID: P2 | Especialidad: Informática | Cantidad de cursos: 2
Profesor: María Torres | ID: P3 | Especialidad: Física | Cantidad de cursos: 1

CAMBIAR PROFESOR DEL CURSO C1
Profesor Carlos Pérez asignado al curso Álgebra

ELIMINAR CURSO C5
Curso eliminado correctamente.

ELIMINAR PROFESOR P3
Profesor eliminado correctamente.

Reporte: Cursos por Profesor
Laura Gómez ? 0 cursos.
Carlos Pérez ? 3 cursos.

LISTA FINAL DE CURSOS
Cursos
Curso: Álgebra | Código: C1 | Profesor: Carlos Pérez
Curso: Programación I | Código: C2 | Profesor: Carlos Pérez
Curso: Física Mecánica | Código: C3 | Profesor: Sin asignar
Curso: Bases de Datos | Código: C4 | Profesor: Carlos Pérez

LISTA FINAL DE PROFESORES
Profesores
Profesor: Laura Gómez | ID: P1 | Especialidad: Matemática | Cantidad de cursos: 0
Profesor: Carlos Pérez | ID: P2 | Especialidad: Informática | Cantidad de cursos: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

Link de repositorio GitHub:

<https://github.com/karenlauk/Programaci-n2>