Trabajo Práctico 6:

Colecciones y Sistema de Stock

Alumna: Eugenia Demarchi

Comisón: 7

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.

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.

Codigo:

```
package practico_6;


public class Producto {

    private String ID;
    private String nombre;
    private double precio;
    private int cantidad;
    private CategoriaProducto categoria;

       public Producto(String ID, String nombre, double precio, int cantidad,
    CategoriaProducto categoria) {
        this.ID = ID;
        this.nombre = nombre;
```

```
        this.precio = precio;
        this.cantidad = cantidad;
        this.categoria = categoria;
    }

    public String getID() {
        return ID;
    }

    public String getNombre() {
        return nombre;
    }

    public double getPrecio() {
        return precio;
    }

    public int getCantidad() {
        return cantidad;
    }

    public CategoriaProducto getCategoria() {
        return categoria;
    }

    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }

public void mostrarInfo(){
    System.out.println("ID: " + ID);
    System.out.println("Nombre: " + nombre);
    System.out.println("Precio: $" + precio);
      System.out.println("Cantidad: " + cantidad);
    System.out.println("Categoria: " + categoria + " (" + categoria.getDescripcion() + ") ");
    System.out.println("----------------------------------------");
}
}


package practico_6;
public enum CategoriaProducto {
ALIMENTOS("Productos Comestibles"),
ELECTRONICA("Dispositivos electronicos"),
ROPA("Prendas de vestir"),
```

```
HOGAR("Articulos para el hogar");

private final String descripcion;


CategoriaProducto(String descripcion){
   this.descripcion =descripcion;
}

   public String getDescripcion() {
      return descripcion;
   }
}

package practico_6;

import java.util.ArrayList;

public class Inventario {
private ArrayList<Producto> productos;

//Constructor que inicializa la lista
public Inventario(){
   productos = new ArrayList<>();
}

public void agregarProducto(Producto p){
   if (p!= null && !productos.contains(p)){
      productos.add(p);
   }
}
public void listarProductos(){
   if (productos.isEmpty()){
      System.out.println("No hay productos en el inventario");
   } else {
      for (Producto p : productos) {
         p.mostrarInfo();
      }
   }
}
public Producto buscarProductoPorId(String Id){
   for (Producto p : productos) {
      if (p.getID().equalsIgnoreCase(Id)){
         return p;
      }
```

```
    }
    return null;
}
public void eliminarProducto(String Id){
    Producto p =buscarProductoPorId(Id);
    if (p != null){
        productos.remove(p);
        System.out.println("Producto eliminado: " + p.getNombre());
    } else {
        System.out.println("No se encontro el producto con ese ID: ");
    }
}
public void actualizarStock(String id, int nuevaCantidad){
     Producto p =buscarProductoPorId(id);
     if (p != null){
         p.setCantidad(nuevaCantidad);
         System.out.println("Stock actualizado. Nuevo stock: " + nuevaCantidad);
     } else {
         System.out.println("No se encontro el producto cobn ese id");
     }
}
public void filtrarPorCategoria(CategoriaProducto categoria){
    System.out.println("Productos de la categoria: "+ categoria);
    for (Producto p : productos) {
        if (p.getCategoria() == categoria){
            p.mostrarInfo();
        }
    }
}
public int obtenerTotalStock(){
    int total= 0;
    for (Producto p : productos){
        total+=p.getCantidad();
    }
    return total;
}
public Producto obtenerProductoConMayortock(){
    if (productos.isEmpty()) return null;
    Producto max = productos.get(0);
    for (Producto p : productos) {
        if (p.getCantidad() > max.getCantidad()){
            max = p;
        }
    }
    return max;
```

```
}
public void filtrarProductosPorPrecio(double min, double max){
   System.out.println("Productos con precio entre: " + min + " y $" + max );
   for (Producto p : productos){
      if (p.getPrecio() >= min && p.getPrecio() <= max){
         p.mostrarInfo();
      }
   }
}

public void mostrarCategoriasDisponibles(){
   System.out.println("Categorias disponibles: ");
   for (CategoriaProducto c: CategoriaProducto.values()){
      System.out.println("- " + c + ": " + c.getDescripcion());
   }
}
}


package practico_6;

public class Principal {


   public static void main(String[] args) {
  Inventario inv = new Inventario();

  //1)Crear 5 productos con diferentes categorias y agregarlos:
  inv.agregarProducto(new Producto("P1", "Chocolate", 1000, 3,
CategoriaProducto.ALIMENTOS));
   inv.agregarProducto(new Producto("P2", "Parlante", 5000, 5,
CategoriaProducto.ELECTRONICA));
    inv.agregarProducto(new Producto("P3", "Tapado", 9800, 13,
CategoriaProducto.ROPA));
     inv.agregarProducto(new Producto("P4", "Alfombra", 77000, 7,
CategoriaProducto.HOGAR));
      inv.agregarProducto(new Producto("P5", "Platos", 33800, 5,
CategoriaProducto.HOGAR));

      //2)Listar productos
      inv.listarProductos();

   }

}
```

//SALIDA POR CONSOLA: run:
ID: P1
Nombre: Chocolate
Precio: $1000.0
Cantidad: 3
Categoria: ALIMENTOS (Productos Comestibles)

ID: P2
Nombre: Parlante
Precio: $5000.0
Cantidad: 5
Categoria: ELECTRONICA (Dispositivos electronicos)

ID: P3
Nombre: Tapado
Precio: $9800.0
Cantidad: 13
Categoria: ROPA (Prendas de vestir)

ID: P4
Nombre: Alfombra
Precio: $77000.0
Cantidad: 7
Categoria: HOGAR (Articulos para el hogar)

ID: P5
Nombre: Platos
Precio: $33800.0
Cantidad: 5
Categoria: HOGAR (Articulos para el hogar)

BUILD SUCCESSFUL (total time: 0 seconds)

```
    //3)bUSCAR PRODUCTOS POR id:
      System.out.println("\nBuscando producto con ID P3: ");
      Producto buscado = inv.buscarProductoPorId("P3");
      if (buscado !=null) buscado.mostrarInfo();
```

//SALIDA POR CONSOLA:

Buscando producto con ID P3:
ID: P3
Nombre: Tapado
Precio: $9800.0

Cantidad: 13
Categoria: ROPA (Prendas de vestir)

BUILD SUCCESSFUL (total time: 0 seconds)

//4)Filtrar por categoria:
inv.filtrarPorCategoria(CategoriaProducto.ALIMENTOS);

SALIDA POR CONSOLA:
Productos de la categoria: ALIMENTOS
ID: P1
Nombre: Chocolate
Precio: $1000.0
Cantidad: 3
Categoria: ALIMENTOS (Productos Comestibles)

BUILD SUCCESSFUL (total time: 0 seconds)

//5) Eliminar un producto por su ID y listar los productos restantes.
inv.eliminarProducto("P5");
inv.listarProductos();

SALIDA POR CONSOLA:
Producto eliminado: Platos
ID: P1
Nombre: Chocolate
Precio: $1000.0
Cantidad: 3
Categoria: ALIMENTOS (Productos Comestibles)

ID: P2
Nombre: Parlante
Precio: $5000.0
Cantidad: 5
Categoria: ELECTRONICA (Dispositivos electronicos)

ID: P3
Nombre: Tapado
Precio: $9800.0
Cantidad: 13
Categoria: ROPA (Prendas de vestir)

ID: P4
Nombre: Alfombra
Precio: $77000.0

Cantidad: 7
Categoria: HOGAR (Articulos para el hogar)

BUILD SUCCESSFUL (total time: 0 seconds)

```
   //6. Actualizar el stock de un producto existente.

  inv.agregarProducto(new Producto("P1", "Chocolate", 1000, 3,
CategoriaProducto.ALIMENTOS));

   inv.actualizarStock("P1", 10);
```

SALIDA POR CONSOLA:
run:
Stock actualizado. Nuevo stock: 13
BUILD SUCCESSFUL (total time: 0 seconds)

```
   //7. Mostrar el total de stock disponible.
   int totalStock =inv.obtenerTotalStock();
      System.out.println("Total de unidades en stock: " + totalStock);
```

SALIDA POR CONSOLA:
run:
Total de unidades en stock: 33
BUILD SUCCESSFUL (total time: 0 seconds)

```
//8. Obtener y mostrar el producto con mayor stock.
Producto productoMayorStock = inv.obtenerProductoConMayorStock();
  if (productoMayorStock != null){
      System.out.println("Producto con mayor stock: ");
      productoMayorStock.mostrarInfo();
  } else {
     System.out.println("No hay productos en el inventario.");
  }
```

SALIDA POR CONSOLA:
run:
Producto con mayor stock:
ID: P3
Nombre: Tapado
Precio: $9800.0
Cantidad: 13
Categoria: ROPA (Prendas de vestir)

BUILD SUCCESSFUL (total time: 0 seconds)

//9. Filtrar productos con precios entre $1000 y $3000.
inv.filtrarProductosPorPrecio(1000, 3000);

SALIDA POR CONSOLA:
Productos con precio entre: 1000.0 y $3000.0
ID: P1
Nombre: Chocolate
Precio: $1000.0
Cantidad: 3
Categoria: ALIMENTOS (Productos Comestibles)

BUILD SUCCESSFUL (total time: 0 seconds)

//10. Mostrar las categorías disponibles con sus descripciones.
inv.mostrarCategoriasDisponibles();

run:
Categorias disponibles:
- ALIMENTOS: Productos Comestibles
- ELECTRONICA: Dispositivos electronicos
- ROPA: Prendas de vestir
- HOGAR: Articulos para el hogar
BUILD SUCCESSFUL (total time: 0 seconds)

EJERCICIO 2:
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.

```
package Ejercicio_02;

public class Autor {
   private String id;
   private String nombre;
   private String nacionalidad;

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

```
    }


    public void mostrarInfo(){
        System.out.println("El id del autor es: " + id);
        System.out.println("El nombre del autor: " + nombre);
        System.out.println("Nacionalidad: " + nacionalidad);
    }

    public String getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public String getNacionalidad() {
        return nacionalidad;
    }

    @Override
    public String toString() {
        return "Autor{" + "id=" + id + ", nombre=" + nombre + ", nacionalidad=" +
nacionalidad + '}';
    }
}



package Ejercicio_02;

public class Libro {
private String isbn;
private String titulo;
private int anioPublicacion;
private Autor autor;

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

```
public void mostrarInfo(){
   System.out.println("ISBN: " + isbn);
   System.out.println("Titulo: " + titulo);
   System.out.println("Año de publicacion: " + anioPublicacion);
   System.out.println("Autor: " + autor);
}

   public String getIsbn() {
      return isbn;
   }

   public String getTitulo() {
      return titulo;
   }

   public int getAnioPublicacion() {
      return anioPublicacion;
   }

   public Autor getAutor() {
      return autor;
   }
}
```

```
package Ejercicio_02;

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

public class Biblioteca {
private String nombre;
private List<Libro> libros;

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

   public String getNombre() {
      return nombre;
   }
public void agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor){
   libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
```

```
}

public void listarLibros(){
   if (libros.isEmpty()){
      System.out.println("No hay libros en la biblioteca");
   } else {
      for (Libro l: libros) {
         l.mostrarInfo();
         System.out.println("--------------------------");
      }
   }
}
public Libro buscarLibroPorIsbn(String isbn){
   for (Libro l : libros) {
      if (l.getIsbn().equalsIgnoreCase(isbn)){
         return l;
      }
   }
   return null;
}
public void eliminarLibro(String isbn) {
   Libro l = buscarLibroPorIsbn(isbn);
   if (l != null){
      libros.remove(l);
      System.out.println("Libro eliminado " + l.getTitulo());
   } else {
      System.out.println("No se encontro el libro con ese isbn");
   }
}
public int obtenerCantidadLibros(){
 return libros.size();
   }

public void filtrarLibrosPorAnio(int anio) {
   System.out.println("Libros publicados en el año " + anio + ":");
   for (Libro l : libros) {
      if (l.getAnioPublicacion() == anio){
         l.mostrarInfo();
      }
   }
}
public void mostrarAutoresDisponibles() {
   System.out.println("Autores disponibles:");

   libros.stream()
```

```
        .map(Libro::getAutor)
        .distinct()
          .forEach(Autor::mostrarInfo);

}
}
```

Clase Principal:

```
//  1. Creamos una biblioteca.
Biblioteca b1 = new Biblioteca("Universo");

//2. Crear al menos tres autores
Autor a1 = new Autor("a1", "Gabriel Garcia Marquez", "Colombia");
Autor a2 = new Autor("a2", "Julio Cortazar", "Argentina");
Autor a3 = new Autor("a3", "Jane Austen", "Gran Bretaña");


//3. Agregar 5 libros asociados a alguno de los Autores a la biblioteca.
b1.agregarLibro("ISBN001", "Cien años de soledad", 1967, a1);
b1.agregarLibro("ISBN002", "El amor en los tiempos de colera", 1985, a1);
b1.agregarLibro("ISBN003", "Rayuela", 1963, a2);
b1.agregarLibro("ISBN004", "Orgullo y prejuicio", 1813, a3);
b1.agregarLibro("ISBN005", "Emma", 1815, a3);

//4. Listar todos los libros con su información y la del autor.
b1.listarLibros();
```

SALIDA POR CONSOLA:

```
run:
ISBN: ISBN001
Titulo: Cien a�os de soledad
A�o de publicacion: 1967
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
---------------------------
ISBN: ISBN002
Titulo: El amor en los tiempos de colera
A�o de publicacion: 1985
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
---------------------------
ISBN: ISBN003
Titulo: Rayuela
A�o de publicacion: 1963
Autor: Autor{id=a2, nombre=Julio Cortazar, nacionalidad=Argentina}
```

```
--------------------------
ISBN: ISBN004
Titulo: Orgullo y prejuicio
A�o de publicacion: 1813
Autor: Autor{id=a3, nombre=Jane Austen, nacionalidad=Gran Breta�a}
--------------------------
ISBN: ISBN005
Titulo: Emma
A�o de publicacion: 1815
Autor: Autor{id=a3, nombre=Jane Austen, nacionalidad=Gran Breta�a}
--------------------------
BUILD SUCCESSFUL (total time: 0 seconds)

//5. Buscar un libro por su ISBN y mostrar su información.
Libro encontrado = b1.buscarLibroPorIsbn("ISBN001");
    if (encontrado != null){
       encontrado.mostrarInfo();
    } else {
       System.out.println("Libro no encontrado");
    }

SALIDA POR CONSOLA:
run:
ISBN: ISBN001
Titulo: Cien a�os de soledad
A�o de publicacion: 1967
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
BUILD SUCCESSFUL (total time: 0 seconds)

//6. Filtrar y mostrar los libros publicados en un año específico.
b1.filtrarLibrosPorAnio(1985);

SALIDA POR CONSOLA:
run:
Libros publicados en el a�o 1985:
ISBN: ISBN002
Titulo: El amor en los tiempos de colera
A�o de publicacion: 1985
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
BUILD SUCCESSFUL (total time: 0 seconds)



//7. Eliminar un libro por su ISBN y listar los libros restantes.
b1.eliminarLibro("ISBN005");
    System.out.println("||||||||||||||||||||||||||||||");
```

```
b1.listarLibros();

SALIDA POR CONSOLA:
 run:
Libro eliminado Emma
||||||||||||||||||||||||||||||
ISBN: ISBN001
Titulo: Cien a�os de soledad
A�o de publicacion: 1967
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
---------------------------
ISBN: ISBN002
Titulo: El amor en los tiempos de colera
A�o de publicacion: 1985
Autor: Autor{id=a1, nombre=Gabriel Garcia Marquez, nacionalidad=Colombia}
---------------------------
ISBN: ISBN003
Titulo: Rayuela
A�o de publicacion: 1963
Autor: Autor{id=a2, nombre=Julio Cortazar, nacionalidad=Argentina}
---------------------------
ISBN: ISBN004
Titulo: Orgullo y prejuicio
A�o de publicacion: 1813
Autor: Autor{id=a3, nombre=Jane Austen, nacionalidad=Gran Breta�a}
---------------------------
BUILD SUCCESSFUL (total time: 0 seconds)

//8. Mostrar la cantidad total de libros en la biblioteca.
int cantidadTotal = b1.obtenerCantidadLibros();
    System.out.println("La cantidad total de libros en " + b1.getNombre() + " es: " +
cantidadTotal);

SALIDA POR CONSOLA:
run:
La cantidad total de libros en Universo es: 5
BUILD SUCCESSFUL (total time: 0 seconds)

//9. Listar todos los autores de los libros disponibles en la biblioteca.
  b1.mostrarAutoresDisponibles();

SALIDA POR CONSOLA:
run:
Autores disponibles:
----------------------
```

El id del autor es: a1
El nombre del autor: Gabriel Garcia Marquez
Nacionalidad: Colombia
El id del autor es: a2
El nombre del autor: Julio Cortazar
Nacionalidad: Argentina
El id del autor es: a3
El nombre del autor: Jane Austen
Nacionalidad: Gran Breta�a
BUILD SUCCESSFUL (total time: 0 seconds)

EJERCICIO 3:

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:

```
package Ejercicio03;

public class Curso {
   private String codigo;
   private String nombre;
   private Profesor profesor;

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

   public String getCodigo() {
      return codigo;
   }

   public String getNombre() {
      return nombre;
   }

   public Profesor getProfesor() {
      return profesor;
   }

   public void setProfesor(Profesor p) {
      if (this.profesor == p) {
```

```
            return;
        }

        // Si ya tenía un profesor, lo desvinculo
        if (this.profesor != null) {
            this.profesor.eliminarCurso(this);
        }

        this.profesor = p; // ← corrección

        // Vinculo este curso al nuevo profesor, si no lo tenía
        if (p != null && !p.getCursos().contains(this)) {
            p.agregarCurso(this);
        }
    }

    public void mostrarInfo() {
        System.out.println("Curso: " + nombre + " (" + codigo + ")");
        if (profesor != null) {
            System.out.println("Profesor: " + profesor.getNombre());
        } else {
            System.out.println("Profesor: [Sin asignar]");
        }
        System.out.println("----------------------------------------");
    }
}

package Ejercicio03;

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

public class Profesor {
    private String id;
    private String nombre;
    private String especialidad;
    private List<Curso> cursos;

    public Profesor(String id, String nombre, String especialidad) {
        this.id = id;
```

```
    this.nombre = nombre;
    this.especialidad = especialidad;
    this.cursos = new ArrayList<>();
}

public String getId() {
    return id;
}

public String getNombre() {
    return nombre;
}

public String getEspecialidad() {
    return especialidad;
}

public List<Curso> getCursos() {
    return Collections.unmodifiableList(cursos);
}

public void agregarCurso(Curso c) {
    if (c != null && !cursos.contains(c)) {
        cursos.add(c);
        if (c.getProfesor() != this) {
            c.setProfesor(this);
        }
    }
}

public void eliminarCurso(Curso c) {
    if (c != null && cursos.remove(c)) {
        if (c.getProfesor() == this) {
            c.setProfesor(null);
        }
    }
}

public void listarCursos() {
    if (cursos.isEmpty()) {
```

```
            System.out.println("No hay cursos disponibles asignados al profesor " +
nombre);
        } else {
            System.out.println("Cursos dictados por " + nombre + ":");
            for (Curso c : cursos) {
                System.out.println("- " + c.getCodigo() + ": " + c.getNombre());
            }
        }
    }

    public void mostrarInfo() {
        System.out.println("Profesor: " + nombre);
        System.out.println("Especialidad: " + especialidad);
        System.out.println("Cantidad de cursos: " + cursos.size());
    }
}



package Ejercicio03;

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


public class Universidad {
private String nombre;
private List<Profesor> profesores;
private List<Curso> cursos;

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

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

```
public void agregarCurso(Curso c){
   if (c !=null && !cursos.contains(c)){
      cursos.add(c);
   }
}
 public Profesor buscarProfesorPorId(String id) {
      for (Profesor p : profesores) {
         if (p.getId().equals(id)) {
            return p;
         }
      }
      return null;
   }

   public Curso buscarCursoPorCodigo(String codigo) {
      for (Curso c : cursos) {
         if (c.getCodigo().equals(codigo)) {
            return c;
         }
      }
      return null;
   }

   public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
      Curso c = buscarCursoPorCodigo(codigoCurso);
      Profesor p = buscarProfesorPorId(idProfesor);
      if (c != null && p != null) {
         c.setProfesor(p);
      }
   }

   public void eliminarCurso(String codigo) {
     Curso c = buscarCursoPorCodigo(codigo);
      if (c != null) {
         if (c.getProfesor() != null) {
            c.setProfesor(null); // romper relación
         }
         cursos.remove(c);
      }
   }
```

```
    public void eliminarProfesor(String id) {
        Profesor p = buscarProfesorPorId(id);
        if (p != null) {
            // dejar sin profesor los cursos que dictaba
            for (Curso c : new ArrayList<>(p.getCursos())) {
                c.setProfesor(null);
            }
            profesores.remove(p);
        }
    }

    public void listarProfesores() {
        for (Profesor p : profesores) {
            p.mostrarInfo();
            p.listarCursos();
            System.out.println();
        }
    }

    public void listarCursos() {
        for (Curso c : cursos) {
            c.mostrarInfo();
        }
    }
}
```

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.

```
package Ejercicio03;
public class Principal {
```

```
public static void main(String[] args) {
        Universidad u = new Universidad("UNLa");

      Profesor p1 = new Profesor("P001", "Ana García", "Matemática");
      Profesor p2 = new Profesor("P002", "Luis Pérez", "Programación");
      Profesor p3 = new Profesor("P003", "Carla Gómez", "Base de Datos");

      Curso c1 = new Curso("C001", "Álgebra");
      Curso c2 = new Curso("C002", "POO");
      Curso c3 = new Curso("C003", "SQL");
      Curso c4 = new Curso("C004", "Redes");
      Curso c5 = new Curso("C005", "Taller Web");

      u.agregarProfesor(p1);
      u.agregarProfesor(p2);
      u.agregarProfesor(p3);

      u.agregarCurso(c1);
      u.agregarCurso(c2);
      u.agregarCurso(c3);
      u.agregarCurso(c4);
      u.agregarCurso(c5);

      u.asignarProfesorACurso("C002", "P002");
      u.asignarProfesorACurso("C003", "P003");
      u.asignarProfesorACurso("C005", "P002");

      System.out.println("=== Listado inicial ===");
      u.listarCursos();
      u.listarProfesores();

      System.out.println("\n=== Cambio de profesor ===");
      c3.setProfesor(p1); // cambiamos el profe de SQL
      u.listarCursos();

      System.out.println("\n=== Eliminamos curso C005 ===");
      u.eliminarCurso("C005");
      u.listarProfesores();

      System.out.println("\n=== Eliminamos profesor P002 ===");
      u.eliminarProfesor("P002");
```

```
        u.listarCursos();
    }
}
```