Resolución Práctico 6:

Ejercicio 1: Colección y Sistema de Stock

En este ejercicio desarrollamos un **sistema de gestión de stock** en Java, compuesto por dos clases y un enum, cuya estructura y funcionalidades son las siguientes:

1. **Clase Producto**
    - **Atributos**
        - id (String): identificador único del producto.
        - nombre (String): denominación del producto.
        - precio (double): precio unitario.
        - cantidad (int): unidades disponibles en stock.
        - categoria (CategoriaProducto): categoría a la que pertenece.

```
// ATRIBUTOS DE PRODUCTO
private String id;
private String nombre;
private double precio;
private int cantidad;
private CategoriaProducto categoria;

// Definimos el constructor con argumentos
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;
}
```

    - **Método**
        - mostrarInfo(): imprime en consola todos los datos del producto (id, nombre, precio, cantidad y categoría).

```
/**
 * Muestra por consola toda la información del producto, incluyendo su ID,
 * nombre, precio, cantidad y categoría.
 */
public void mostrarInfo() {
    System.out.println("Producto{" + "id=" + id + ", nombre=" + nombre + ", precio=" + precio + ", cantidad=" + cantidad + ", categoria=" + categoria + '}');
}
```

2. **Enum CategoriaProducto**
   Define las categorías de producto y su descripción asociada:
    - ALIMENTOS (“Productos comestibles”)
    - ELECTRONICA (“Dispositivos electrónicos”)

- ROPA (“Prendas de vestir”)
- HOGAR (“Artículos para el hogar”)

```java
public enum CategoriaProducto {
    // Definimos los valores del enum con una descripcion.
    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;
    }

}
```

3. **Clase Inventario**
   - **Atributo**
     - productos (ArrayList<Producto>): colección dinámica donde se almacenan los productos.

```java
public class Inventario {

    // Instanciamos una coleccion para guardar los productos
    private ArrayList<Producto> productos;

    // Inicilizamos la coleccion en el constructor
    public Inventario() {
        this.productos = new ArrayList();
    }
```

   - **Métodos**
     - agregarProducto(Producto p): añade un producto al inventario.

```java
/**
 * Agrega un nuevo producto a la lista de productos.
 *
 * @param p El producto que se desea agregar.
 */
public void agregarProducto(Producto p) {
    productos.add(p);
}
```

- listarProductos(): recorre y muestra todos los productos.

```java
/**
 * Lista todos los productos disponibles mostrando su información por
 * consola.
 */
public void listarProductos() {
    for (Producto p : productos) {
        p.mostrarInfo();
    }
}
```

- buscarProductoPorId(String id): devuelve el producto con el ID indicado o null si no existe.

```java
/**
 * Busca un producto en la lista por su ID.
 *
 * @param id El identificador del producto a buscar.
 * @return El producto si se encuentra, o null si no existe.
 */
public Producto buscarProductoPorId(String id) {
    Producto productoEncontrado = null;
    int i = 0;

    while (i < productos.size() && !this.productos.get(i).getId().equalsIgnoreCase(id)) {
        i++;
    }

    if (i < this.productos.size()) {
        productoEncontrado = this.productos.get(i);
    }

    return productoEncontrado;
}
```

- eliminarProducto(String id): elimina del inventario el producto con el ID dado.

```java
/**
 * Elimina un producto de la lista según su ID.
 *
 * @param id El identificador del producto a eliminar.
 */
public void eliminarProducto(String id) {
    Producto p = buscarProductoPorId(id);

    if (p != null) {
        this.productos.remove(p);
        System.out.println("El producto se elimino correctamente...");
    } else {
        System.out.println("ID no encontrado!");
    }
}
```

- actualizarStock(String idProducto, int nuevaCantidad): modifica la cantidad de un producto si existe y el valor es positivo.

```java
/**
 * Actualiza el stock de un producto existente.
 *
 * @param id El identificador del producto a actualizar.
 * @param nuevaCantidad La nueva cantidad de stock (debe ser mayor a 0).
 */
public void actualizarStock(String id, int nuevaCantidad) {
    Producto p = buscarProductoPorId(id);

    if (p != null && nuevaCantidad > 0) {
        p.setCantidad(nuevaCantidad);
        System.out.println("El stock se actualizo con exito!");
    } else if (nuevaCantidad < 0) {
        System.out.println("Debe ingresar una cantidad positiva...");
    } else if (p == null) {
        System.out.println("ID no encontrado!");
    }
}
```

- filtrarPorCategoria(CategoriaProducto categoria): retorna una lista de productos de la categoría especificada.

```java
/**
 * Filtra y devuelve una lista de productos que pertenecen a una categoría
 * específica.
 *
 * @param categoria La categoría a filtrar.
 * @return Una lista de productos que pertenecen a la categoría
 * especificada.
 */
public ArrayList<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
    ArrayList<Producto> productosFiltrados = new ArrayList();

    for (Producto p : productos) {
        if (p.getCategoria() == categoria) {
            productosFiltrados.add(p);
        }
    }

    return productosFiltrados;
}
```

- obtenerTotalStock(): calcula la suma total de unidades en stock.

```java
/**
 * Calcula y devuelve el total de stock de todos los productos.
 *
 * @return La suma total de unidades en stock.
 */
public int obtenerTotalStock() {
    int totalStock = 0;
    for (Producto p : productos) {
        totalStock += p.getCantidad();
    }

    return totalStock;
}
```

- productoConMayorStock(): identifica y devuelve el producto con la mayor cantidad disponible.

```java
/**
 * Busca y devuelve el producto con mayor cantidad en stock. Si hay varios
 * con el mismo stock máximo, devuelve el último encontrado.
 *
 * @return El producto con mayor stock, o null si la lista está vacía.
 */
public Producto productoConMayorStock() {
    int maxStock = -1;
    Producto productoConMayorStock = null;
    for (Producto p : productos) {
        if (p.getCantidad() >= maxStock) {
            maxStock = p.getCantidad();
            productoConMayorStock = p;
        }
    }

    return productoConMayorStock;
}
```

- filtrarProductosPorPrecio(double min, double max): devuelve los productos cuyo precio está entre los valores min y max (inclusive).

```java
/**
 * Filtra los productos cuyo precio se encuentra entre un mínimo y un máximo
 * (inclusive).
 *
 * @param min El precio mínimo.
 * @param max El precio máximo.
 * @return Una lista de productos cuyo precio está dentro del rango
 * especificado.
 */
public ArrayList<Producto> filtrarProductosPorPrecio(double min, double max) {
    ArrayList<Producto> productosFiltrados = new ArrayList();

    for (Producto p : productos) {
        if (p.getPrecio() >= min && p.getPrecio() <= max) {
            productosFiltrados.add(p);
        }
    }

    return productosFiltrados;
}
```

- mostrarCategoriasDisponibles(): imprime todas las categorías definidas en el enum, junto a su descripción.

```java
/**
 * Muestra por consola todas las categorías de producto disponibles junto a
 * su descripción.
 */
public void mostrarCategoriasDisponibles() {
    System.out.println("Las categorias disponibles son: ");
    for (CategoriaProducto c : CategoriaProducto.values()) {
        System.out.println("Categoria: " + c);
        System.out.println("Descripcion: " + c.getDescripcion());
    }
}
```

4. **Clase Main**

   En el método main se realiza una secuencia de pruebas para validar el funcionamiento completo del sistema:

   - Se crea una instancia de Inventario.

```java
public static void main(String[] args) {
    // Instanciamos un inventario
    Inventario inv = new Inventario();
```

- Se agregan 5 productos distintos (con distintas categorías y valores).

```
// Agregamos 5 productos al inventario
inv.agregarProducto(new Producto("001","Smart TV 50",200000,6,CategoriaProducto.ELECTRONICA));
inv.agregarProducto(new Producto("002","Hamburguesas x4",2500,10,CategoriaProducto.ALIMENTOS));
inv.agregarProducto(new Producto("003","Pantalon negro hombre",45000,4,CategoriaProducto.ROPA));
inv.agregarProducto(new Producto("004","Pan kg",1500,8,CategoriaProducto.ALIMENTOS));
inv.agregarProducto(new Producto("005","Silla de madera UN",50000,7,CategoriaProducto.HOGAR));
```

- Se listan todos los productos.

```
// Listamos los productos agregados
inv.listarProductos();
```

- Se busca un producto por su ID y se muestra su información.

```
// Listamos el producto con id 001
System.out.println("\nProducto id 001: ");
Producto productoPorId = inv.buscarProductoPorId("001"); // Utilizamos el metodo de inventario para buscar por id
productoPorId.mostrarInfo(); // Mostramos la info del producto encontrado
```

- Se filtran y muestran los productos de una categoría concreta.

```
// Filtramos productos por categoria ROPA
System.out.println("\nProductos filtrados");
ArrayList<Producto> productosPorCategoria = inv.filtrarPorCategoria(CategoriaProducto.ROPA); // Llamamos al metodo de inventario para filtrar por categoria
for(Producto p : productosPorCategoria) { // Recorremos los productos filtrados
    p.mostrarInfo(); // Mostramos los resultados
}
```

- Se elimina un producto por ID y se muestra el inventario actualizado.

```
// Eliminamos un producto por id
System.out.println("\nEliminado producto id 005");
inv.eliminarProducto("005"); // Llamamos al metodo de inventario para eliminar por id
System.out.println("\nInventario actualizado: ");
inv.listarProductos(); // Mostramos el inventario actualizado
```

- Se actualiza el stock de un producto existente.

```
// Actualizamos el stock del producto con id 003
inv.actualizarStock("003", 10);
```

- Se muestra el producto con mayor cantidad en stock.

```
// Mostramos el producto con mayor stock
System.out.println("\nProducto con mayor Stock: ");
inv.productoConMayorStock().mostrarInfo();
```

- Se filtran los productos con precio entre $1000 y $3000 y se muestran.

```
// Filtramos por precio entre 1000 y 3000
System.out.println("\nProductos entre $1000 y $3000");
ArrayList<Producto> productosFiltradosPorPrecio = inv.filtrarProductosPorPrecio(1000, 3000); // Utilizamos el metodo de filtrar por precio de la clase inventario
for(Producto p: productosFiltradosPorPrecio){ // Recorremos los productos filtrados
    p.mostrarInfo(); // Mostramos la info de cada producto
}
```

- Por último, se listan todas las categorías disponibles junto con sus descripciones.

```
// Mostramos categorias disponibles
System.out.println("\nCategorias disponibles: ");
inv.mostrarCategoriasDisponibles();
```

Salida por pantalla:

```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ Ejercitacion3 ---
Producto{id=001, nombre=Smart TV 50, precio=200000.0, cantidad=6, categoria=ELECTRONICA}
Producto{id=002, nombre=Hamburguesas x4, precio=2500.0, cantidad=10, categoria=ALIMENTOS}
Producto{id=003, nombre=Pantalon negro hombre, precio=45000.0, cantidad=4, categoria=ROPA}
Producto{id=004, nombre=Pan kg, precio=1500.0, cantidad=8, categoria=ALIMENTOS}
Producto{id=005, nombre=Silla de madera UN, precio=50000.0, cantidad=7, categoria=HOGAR}

Producto id 001:
Producto{id=001, nombre=Smart TV 50, precio=200000.0, cantidad=6, categoria=ELECTRONICA}

Productos filtrados
Producto{id=003, nombre=Pantalon negro hombre, precio=45000.0, cantidad=4, categoria=ROPA}

Eliminado producto id 005
El producto se elimino correctamente...

Inventario actualizado:
Producto{id=001, nombre=Smart TV 50, precio=200000.0, cantidad=6, categoria=ELECTRONICA}
Producto{id=002, nombre=Hamburguesas x4, precio=2500.0, cantidad=10, categoria=ALIMENTOS}
Producto{id=003, nombre=Pantalon negro hombre, precio=45000.0, cantidad=4, categoria=ROPA}
Producto{id=004, nombre=Pan kg, precio=1500.0, cantidad=8, categoria=ALIMENTOS}
El stock se actualizo con exito!

Stock disponible: 34

Producto con mayor Stock:
Producto{id=003, nombre=Pantalon negro hombre, precio=45000.0, cantidad=10, categoria=ROPA}

Productos entre $1000 y $3000
Producto{id=002, nombre=Hamburguesas x4, precio=2500.0, cantidad=10, categoria=ALIMENTOS}
Producto{id=004, nombre=Pan kg, precio=1500.0, cantidad=8, categoria=ALIMENTOS}

Categorias disponibles:
Las categorias disponibles son:
Categoria: ALIMENTOS
Descripcion: Productos comestibles
Categoria: ELECTRONICA
Descripcion: Dispositivos electrónicos
Categoria: ROPA
Descripcion: Prendas de vestir
Categoria: HOGAR
Descripcion: Artículos para el hogar
```

Ejercicio 2: Biblioteca y Libros

En este ejercicio desarrollamos un **sistema de gestión de biblioteca** en Java con composición, compuesto por tres clases, cuya estructura y funcionalidades son las siguientes:

1. **Clase Autor**

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

```
*/
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;
    }
```

- Metodos
    - public String mostrarInfo()

```
public String mostrarInfo(){
    return "Autor{id: "+id+
                "\nnombre: "+nombre+
                "\nnacionalidad: "+nacionalidad+
                "}";
}
```

Ademas de Getters y Setters

2. Clase Libro:
    - Atributos:

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

```
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;
    }
```

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

```
public void mostrarInfo(){
    System.out.println("Libro {isbn: "+isbn+
                       "\ntitulo: "+titulo+
                       "\n anio de Publicacion: "+anioPublicacion+
                       "\n "+autor.mostrarInfo()+
                       "}");
}
```

3. Clase Biblioteca

- **Atributo:**
  - private String nombre;
  - **private List<Libro> libros** → Colección de libros de la biblioteca.

```
public class Biblioteca {

    private String nombre;
    private List<Libro> libros;

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

    }
```

- Metodos:

- agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor)-> debemos implementar la composición.

```
public void agregarLibro(String isbn, String titulo,
                         int anioPublicacion, Autor autor) {
    if (!isbn.equals("") && isbn != null && !titulo.equals("")
            && titulo != null && anioPublicacion > 0 && autor != null) {
        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
    }

}
```

- listarLibros()

```
public void listarLibros() {
    System.out.println("La lista de libros de la biblioteca" + nombre + " es: ");
    for (Libro libro : libros) {
        libro.mostrarInfo();
    }
}
```

- buscarLibroPorIsbn(String isbn)

```
public Libro buscarLibroPorIsbn(String isbn) {
    for (Libro libro : libros) {
        if (libro.getIsbn().equals(isbn)) {
            return libro;
        }
    }
    return null;
}
```

- eliminarLibro(String isbn)

```
public void eliminarLibro(String isbn) {
    if (isbn != null) {
        Libro libroEliminar = this.buscarLibroPorIsbn(isbn);
        if (libroEliminar != null) {
            libros.remove(libroEliminar);
        }
    }
}
```

- obtenerCantidadLibros()

```
public int obtenerCantidadLibros(){
    return libros.size();
}
```

- filtrarLibrosPorAnio(int anio)

```
public List<Libro> filtrarLibrosPorAnio(int anio){
    List<Libro> libroPorAnio=new ArrayList();
    for(Libro libro:libros){
        if(libro.getAnioPublicacion()==anio){
            libroPorAnio.add(libro);
        }
    }
    return Collections.unmodifiableList(libroPorAnio);
}
```

- mostrarAutoresDisponibles()

```
public void mostrarAutoresDisponibles(){
    for(Libro libro: libros){
        System.out.println(libro.getAutor().mostrarInfo());
    }
}
```

- Clase Main

1 Creamos la biblioteca:

```
public static void main(String[] args) {
    // Creamos una biblioteca
    Biblioteca biblioteca=new Biblioteca("Biblioteca Municipal");
```

2 Crear al menos tres autores

```
//Creamos Autores
Autor laura=new Autor("1","Laura Mendez","Argentina");
Autor diego= new Autor("2","Diego Salazar","Mexico");
Autor sofia= new Autor("3","Sofia Pereira","Uruguay");
```

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

```
//Agregamos los libros por composicion
biblioteca.agregarLibro("A234","Logica de Programacion Moderna", 2021, laura);
biblioteca.agregarLibro("H345", "Sistemas Distribuidos Esenciales", 2023, diego);
biblioteca.agregarLibro("S234", "Arquitectura de Software", 2021, laura );
biblioteca.agregarLibro("C234", "Patron de diseño con Java", 2020, sofia );
biblioteca.agregarLibro("F234", "Estructura de Datos en Profundidad", 2024, diego);
```

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

```
//Listamos los libros de la biblioteca
System.out.println("Los libros de la biblioteca "+biblioteca.getNombre()+"son: ");
biblioteca.listarLibros();
System.out.println("-----------------------------------------------");
```

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

```
//buscamos un libro por ISBN
System.out.println("Buscamos un libro por isbn A234");
biblioteca.buscarLibroPorIsbn("A234").mostrarInfo();
System.out.println("..................................................");
```

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

```
//buscamos los libros del mismo anio de publicacion

int anioFiltro=2021;
List<Libro> librosPorAnio=biblioteca.filtrarLibrosPorAnio(anioFiltro);

System.out.println("Los libros encontrados del anio "+anioFiltro+" son: ");
for(Libro libro:librosPorAnio){
    libro.mostrarInfo();
}
System.out.println("////////////////////////////////////////77");
```

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

```
//eliminamos un libro por isbn

biblioteca.eliminarLibro("S234");
```

8 Mostrar la cantidad total de libros en la biblioteca.

```
//Listamos los libros despues de eliminar uno por isbn
System.out.println("Imprimimos los libros despues de eliminar uno");
biblioteca.listarLibros();
```

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

```
//Listamos los autores de los libros de la biblioteca.
System.out.println("Los autores de los libros de la biblioteca son: ");
biblioteca.mostrarAutoresDisponibles();
```

Salida por pantalla:

```
run:
Los libros de la biblioteca Biblioteca Municipalson:
La lista de libros de la bibliotecaBiblioteca Municipal es:
Libro {isbn: A234
titulo: Logica de Programacion Moderna
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}}
Libro {isbn: H345
titulo: Sistemas Distribuidos Esenciales
 anio de Publicacion: 2023
 Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}}
Libro {isbn: S234
titulo: Arquitectura de Software
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}}
```

```
Libro {isbn: C234
titulo: Patron de diseño con Java
 anio de Publicacion: 2020
 Autor{id: 3
nombre: Sofia Pereira
nacionalidad: Uruguay}}
Libro {isbn: F234
titulo: Estructura de Datos en Profundidad
 anio de Publicacion: 2024
 Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}}
-----------------------------------------------
Buscamos un libro por isbn A234
Libro {isbn: A234
titulo: Logica de Programacion Moderna
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}}
.....................................................
Los libros encontrados del anio 2021 son:
Libro {isbn: A234
titulo: Logica de Programacion Moderna
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}}
Libro {isbn: S234
titulo: Arquitectura de Software
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}}
//////////////////////////////////////////77
Imprimimos los libros despues de eliminar uno
La lista de libros de la bibliotecaBiblioteca Municipal es:
Libro {isbn: A234
titulo: Logica de Programacion Moderna
 anio de Publicacion: 2021
 Autor{id: 1
nombre: Laura Mendez
```

```
nacionalidad: Argentina}}
Libro {isbn: H345
titulo: Sistemas Distribuidos Esenciales
 anio de Publicacion: 2023
 Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}}
Libro {isbn: C234
titulo: Patron de diseño con Java
 anio de Publicacion: 2020
 Autor{id: 3
nombre: Sofia Pereira
nacionalidad: Uruguay}}
Libro {isbn: F234
titulo: Estructura de Datos en Profundidad
 anio de Publicacion: 2024
 Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}}
La cantidad total de libros en la biblioteca Biblioteca Municipalson: 4
Los autores de los libros de la biblioteca son:
Autor{id: 1
nombre: Laura Mendez
nacionalidad: Argentina}
Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}
Autor{id: 3
nombre: Sofia Pereira
nacionalidad: Uruguay}
Autor{id: 2
nombre: Diego Salazar
nacionalidad: Mexico}
BUILD SUCCESSFUL (total time: 1 second)
```

Ejercicio 2: Universidad, Profesor y Curso (bidireccional 1 a N)

En este ejercicio desarrollamos un **sistema de gestión de universidad** en Java , compuesto por tres clases, cuya estructura y funcionalidades son las siguientes:

1. **Clase Profesor**

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

```
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<>();
    }
```

Metodos:

- **agregarCurso(Curso c)** → Agrega el curso a su lista si no está y sincroniza el lado del curso.

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

- **eliminarCurso(Curso c)** → Quita el curso y sincroniza el lado del curso (dejar `profesor` en `null` si corresponde).

```
public void eliminarCurso(Curso curso) {

    if (curso != null && cursos.contains(curso)) {

        cursos.remove(curso);

        if (curso.getProfesor() == this) {
            curso.setProfesor(null);

        }
    }
}
```

- **listarCursos()** → Muestra códigos y nombres.

```
public void listarCursos(){
    System.out.println("Los cursos del profesor "+nombre+" son: ");
    for(Curso curso: cursos){
        curso.mostrarInfo();
    }
}
```

- **mostrarInfo()** → Imprime datos del profesor y cantidad de cursos.

```
public void mostrarInfo(){
    System.out.println("El profesor con id: "+id+
            "\nnombre: "+nombre+
            "\nespecialidad: "+especialidad+
            "\nejerse en "+cursos.size()+" cursos.");
}
```

2 Clase Curso

Atributos;

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

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

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

Metodos:

- **setProfesor(Profesor p)** → Asigna/cambia el profesor **sincronizando ambos lados**:
  - Si tenía profesor previo, quitarse de su lista.

```java
public void setProfesor(Profesor profesor) {
   //Si paso el mismo profesor ya asignado no hago nada
    if(profesor == this.profesor){
    return;
}
    //Si el profesor seteado es distinto a null
    //primero nos borramos de su lista
   if(this.profesor != null){
      this.profesor.eliminarCurso(this);

   }

   //Asignamos al nuevo profesor
   this.profesor=profesor;

   //Si el nuevo profesor es distinto de null y no agrego a este curso
   //Lo agregamos
   if(profesor!=null && !profesor.getCursos().contains(this)){
       profesor.agregarCurso(this);
   }
```

- **mostrarInfo()** → Muestra código, nombre y nombre del profesor (si tiene).

```
public void mostrarInfo(){
    System.out.println("El codigo del curso es: "+codigo+
            "\nel nombre del curso es: "+nombre+
            "\nel profesor asignado al curso es: "+profesor.getNombre());
}
```

3 Clase Universidad

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

```
public class Universidad {

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

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

Metodos:

- **agregarProfesor(Profesor p)**

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

}
```

- **agregarCurso(Curso c)**

```
public void agregarCurso(Curso curso) {
    if (curso != null && !cursos.contains(curso)) {
        cursos.add(curso);
    }
}
```

- **asignarProfesorACurso(String codigoCurso, String idProfesor)** → Usa setProfesor del curso.

```
public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
        Profesor profesor= this.buscarProfesorPorId(idProfesor);
        Curso curso=this.buscarCursoPorCodigo(codigoCurso);
        if(profesor!=null && curso!=null){
        curso.setProfesor(profesor);
        }else{
            System.out.println("O el docente o el curso no existen "
                    + "en la universidad "+nombre);
        }
}
```

- **listarProfesores() / listarCursos()**

```
public void listarProfesor() {
    System.out.println("Los profsores de la universidad " + nombre
            + " cuenta con " + profesores.size() + " docentes: ");
    for (Profesor profesor : profesores) {
        profesor.mostrarInfo();
        System.out.println("--------------------");
    }
}

public void listarCursos() {
    System.out.println("La unidversidad " + nombre
            + " cuenta con " + cursos.size() + " cursos y son: ");
    for (Curso curso : cursos) {
        curso.mostrarInfo();

    }

}
```

- **buscarProfesorPorId(String id)**

```
public Profesor buscarProfesorPorId(String id) {
    if (id != null) {
        for (Profesor profesor : profesores) {
            if (profesor.getId().equals(id)) {
                return profesor;
            }
        }

    }
     return null;
}
```

- **buscarCursoPorCodigo(String codigo)**

```
public Curso buscarCursoPorCodigo(String codigo) {
    if (codigo != null) {
        for (Curso curso : cursos) {
            if (curso.getCodigo().equals(codigo)) {
                return curso;
            }
        }

    }
return null;
}
```

- **eliminarCurso(String codigo)** → Debe **romper la relación** con su profesor si la hubiera.

```
public void eliminarProfesor(Profesor profesor) {
    if (profesor != null && profesores.contains(profesor)) {

        for (Curso curso : new ArrayList<>(profesor.getCursos())) {
            curso.setProfesor(null);
        }
        profesores.remove(profesor);
    }
}
```

- **eliminarProfesor(String id)** → Antes de remover, dejar null los cursos que dictaba.

```
public void eliminarCurso(Curso curso) {
    if (curso != null && cursos.contains(curso)) {
        curso.setProfesor(null);
        cursos.remove(curso);
    }
}
```

Clase Main:

1 Crear **al menos 3 profesores** y **5 cursos**.

```
public static void main(String[] args) {
    // Instanciamos 3 profesores
    Profesor profesor1= new Profesor("1","Leandro De Angelis","ADO");
    Profesor profesor2= new Profesor("2","Ariel Enferrel","Programacion");
    Profesor profesor3= new Profesor("3","Carlos Frede","Financiera");

    //Instanciamos 5 cursos
    Curso programacion1= new Curso("1","Programacion1");
    Curso ado= new Curso("2","ADO");
    Curso financiera= new Curso("3","Financiera 1");
    Curso programacion2= new Curso("4","Programacion 2");
    Curso baseDatos= new Curso("5","Base de Datos");
```

2 Agregar profesores y cursos a la universidad.

```
//Creamos la universidad
Universidad uni= new Universidad("UTN");

//Agregamos los profesores a la universidad
uni.agregarProfesor(profesor3);
uni.agregarProfesor(profesor2);
uni.agregarProfesor(profesor1);

//Agregamos los cursos
uni.agregarCurso(ado);
uni.agregarCurso(programacion1);
uni.agregarCurso(financiera);
uni.agregarCurso(programacion2);
uni.agregarCurso(baseDatos);
```

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

```
//Asignamos los profesores con los cursos
uni.asignarProfesorACurso("2", "1");
uni.asignarProfesorACurso("1", "2");
uni.asignarProfesorACurso("4", "2");
uni.asignarProfesorACurso("5", "2");
uni.asignarProfesorACurso("3", "3");
```

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

```
//Listamos profesores con cursos
uni.listarProfesor();

//Listamos cursos
uni.listarCursos();

//Listamos profesores con cursos
uni.listarProfesor();
```

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

```
//Cambiamos un profesor de curso
 System.out.println("////////////////|||||||||||||||||||||||||||||||");
   System.out.println("Reasignamos el curso de Programacion 1 al docente"
           + "Leandro De Angelis");
uni.asignarProfesorACurso("1", "1");
uni.listarCursos();
uni.listarProfesor();
```

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

```
//Removemos un curso y validamos que no lo tiene al profesor
   System.out.println("**********************************");
   System.out.println("Removemos el curso de id=2 y vemos los profesores");
uni.eliminarCurso(financiera);
uni.listarProfesor();
//Vemos que ahora hay 1 curso menos en la universidad
uni.listarCursos();
```

7 Remover un profesor y dejar profesor = null,

8 Mostrar un reporte: cantidad de cursos por profesor.

```
//Removemos un profesor
    System.out.println("||||||||||||||||||||||||||||||||||||||||||||");
    System.out.println("Removemos al profesor de id=1");
    uni.eliminarProfesor(profesor1);

    uni.listarProfesor();
```

Salida por pantalla:

```
run:
Los profsores de la universidad UTN cuenta con 3 docentes:
El profesor con id: 3
nombre: Carlos Frede
especialidad: Financiera
ejerse en 1 cursos.
--------------------
El profesor con id: 2
nombre: Ariel Enferrel
especialidad: Programacion
ejerse en 3 cursos.
--------------------
El profesor con id: 1
nombre: Leandro De Angelis
especialidad: ADO
ejerse en 1 cursos.
--------------------
La unidversidad UTN cuenta con 5 cursos y son:
El codigo del curso es: 2
el nombre del curso es: ADO
el profesor asignado al curso es: Leandro De Angelis
```

```
El codigo del curso es: 1
el nombre del curso es: Programacion1
el profesor asignado al curso es: Ariel Enferrel
El codigo del curso es: 3
el nombre del curso es: Financiera 1
el profesor asignado al curso es: Carlos Frede
El codigo del curso es: 4
el nombre del curso es: Programacion 2
el profesor asignado al curso es: Ariel Enferrel
El codigo del curso es: 5
el nombre del curso es: Base de Datos
el profesor asignado al curso es: Ariel Enferrel
////////////////||||||||||||||||||||||||||||
Reasignamos el curso de Programacion 1 al docenteLeandro De Angelis
La unidversidad UTN cuenta con 5 cursos y son:
El codigo del curso es: 2
el nombre del curso es: ADO
el profesor asignado al curso es: Leandro De Angelis
El codigo del curso es: 1
el nombre del curso es: Programacion1
el profesor asignado al curso es: Leandro De Angelis
El codigo del curso es: 3
el nombre del curso es: Financiera 1
el profesor asignado al curso es: Carlos Frede
El codigo del curso es: 4
el nombre del curso es: Programacion 2
el profesor asignado al curso es: Ariel Enferrel
El codigo del curso es: 5
el nombre del curso es: Base de Datos
el profesor asignado al curso es: Ariel Enferrel
Los profsores de la universidad UTN cuenta con 3 docentes:
El profesor con id: 3
nombre: Carlos Frede
especialidad: Financiera
ejerse en 1 cursos.
--------------------
El profesor con id: 2
nombre: Ariel Enferrel
especialidad: Programacion
ejerse en 2 cursos.
--------------------
El profesor con id: 1
```

```
nombre: Leandro De Angelis
especialidad: ADO
ejerse en 2 cursos.
--------------------
**********************************
Removemos el curso de id=2 y vemos los profesores
Los profsores de la universidad UTN cuenta con 3 docentes:
El profesor con id: 3
nombre: Carlos Frede
especialidad: Financiera
ejerse en 0 cursos.
--------------------
El profesor con id: 2
nombre: Ariel Enferrel
especialidad: Programacion
ejerse en 2 cursos.
--------------------
El profesor con id: 1
nombre: Leandro De Angelis
especialidad: ADO
ejerse en 2 cursos.
--------------------
La unidversidad UTN cuenta con 4 cursos y son:
El codigo del curso es: 2
el nombre del curso es: ADO
el profesor asignado al curso es: Leandro De Angelis
El codigo del curso es: 1
el nombre del curso es: Programacion1
el profesor asignado al curso es: Leandro De Angelis
El codigo del curso es: 4
el nombre del curso es: Programacion 2
el profesor asignado al curso es: Ariel Enferrel
El codigo del curso es: 5
el nombre del curso es: Base de Datos
el profesor asignado al curso es: Ariel Enferrel
||||||||||||||||||||||||||||||||||||||
Removemos al profesor de id=1
Los profsores de la universidad UTN cuenta con 2 docentes:
El profesor con id: 3
nombre: Carlos Frede
especialidad: Financiera
ejerse en 0 cursos.
--------------------
```

```
El profesor con id: 2
nombre: Ariel Enferrel
especialidad: Programacion
ejerse en 2 cursos.
--------------------
```