PROGRAMACIÓN II

TRABAJO PRÁCTICO Nº6: COLECCIONES

CHRISTIAN EMMANUEL OLIVERO

https://github.com/CHRISTIAN2320/UTN-TUPAD-Programacion2

Caso Práctico 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.

```java
public class Producto {
    private String id ;
    private String nombre ;
    private double precio ;
    private int cantidad;
    private CategoriaProducto categoria;
    //Constructor
    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 void mostrarInfo(Producto producto) {
    System.out.println(producto.toString());
    }
    public String getId() {
        return id;
    }
```

```java
    public CategoriaProducto getCategoria() {
        return categoria;
    }
    public String getNombre() {
        return nombre;
    }
    public double getPrecio() {
        return precio;
    }
    public int getCantidad() {
        return cantidad;
    }
    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }
    @Override
    public String toString() {
        return "Producto{" + "id=" + id + ", nombre=" + nombre + ", precio=" + precio + ", cantidad=" + cantidad + ",
categoria=" + categoria + '}' ;
    }
```

```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;
    }
}
```

```java
import java.util.ArrayList;
import java.util.Iterator;

public class Inventario {
    private ArrayList<Producto> productos;
    //Constructor
    public Inventario() {
        this.productos = new ArrayList<>();
    }
    public void agregarProducto(Producto p) {
        this.productos.add(p);
    }
    public void listarProducto() {
        for (Producto producto : productos) {
            System.out.println(producto);
        }
    }
    public Producto buscarProductoPorId(String id) {
        for (Producto p : productos) {
            if (id.equalsIgnoreCase(p.getId())) {
                return p;
            }
        }
        return null;
    }
    public void eliminarProducto(String id) {
        Iterator<Producto> it = productos.iterator();
        while (it.hasNext()) {
            Producto p = it.next();
            if (id.equals(p.getId())) {
                it.remove();
            }
        }
    }
```

```java
public void actualizarStock(String id, int nuevaCantidad) {
    for (int i = 0; i < productos.size(); i++) {
        Producto p = productos.get(i);
        if (id.equalsIgnoreCase(p.getId())) {
            p.setCantidad(nuevaCantidad); //
        }
    }
}
public ArrayList<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
    ArrayList<Producto> instPorCategoria = new ArrayList<>();
    for (Producto producto : productos) {
        if (producto.getCategoria() == categoria) {
            instPorCategoria.add(producto);
        }
    }
    return instPorCategoria;
}
```

```java
public int obtenerTotalStock(String id) {
    int total = 0;
    for (Producto producto : productos) {
        if (producto.getId().equalsIgnoreCase(id)) {
            total += producto.getCantidad();
        }
    }
    return total;
}
public Producto obtenerProductoConMayorStock() {
    if (productos.isEmpty()) {
        return null;
    }
    Producto prod = productos.get(0); // asumimos el primero como el mayor al inicio
    for (Producto producto : productos) {
        if (producto.getCantidad() > prod.getCantidad()) {
            prod = producto;
        }
    }
    return prod;
public ArrayList<Producto> filtrarProductosPorPrecio(double min, double max) {
    ArrayList<Producto> FiltroProducto = new ArrayList<>();
    for (Producto producto : productos) {
        if (producto.getPrecio() > min && producto.getPrecio() < max) {

            FiltroProducto.add(producto);
        }
    }
    return FiltroProducto;
}
public ArrayList<CategoriaProducto> mostrarCategoriasDisponibles() {
    ArrayList<CategoriaProducto> categoriasDisponibles = new ArrayList<>();
    for (Producto producto : productos) {
        CategoriaProducto cat = producto.getCategoria();
        if (cat != null && !categoriasDisponibles.contains(cat)) {
            categoriasDisponibles.add(cat);
        }
    }
    return categoriasDisponibles;
}
```

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

```
public static void main(String[] args) {
    Producto p1 = new Producto("ABC123", "MiniPimer", 87500, 10,CategoriaProducto.ELECTRONICA);
    Producto p2 = new Producto("FGH456", "Parlante", 350000, 5 , CategoriaProducto.ELECTRONICA);
    Producto p3 = new Producto("QWR789", "Remera", 50000, 20, CategoriaProducto.ROPA);
    Producto p4 = new Producto("TYU369", "Sillon", 180000, 5,CategoriaProducto.HOGAR);
    Producto p5= new Producto("JIL426", "Harina", 700, 250 , CategoriaProducto.ALIMENTOS);

    Inventario inventario1 = new Inventario();

    inventario1.agregarProducto(p1);
    inventario1.agregarProducto(p2);
    inventario1.agregarProducto(p3);
    inventario1.agregarProducto(p4);
    inventario1.agregarProducto(p5);
```

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

```
        inventario1.listarProducto();
```

```
tput ×
Programacion II - C:\Users\Pikachu-PC\Desktop\TUP\2° Cuatrimestre\Programacion II ×   Ejercicio1 (run) ×
run:
Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA}
Producto{id=FGH456, nombre=Parlante, precio=350000.0, cantidad=5, categoria=ELECTRONICA}
Producto{id=QWR789, nombre=Remera, precio=50000.0, cantidad=20, categoria=ROPA}
Producto{id=TYU369, nombre=Sillon, precio=180000.0, cantidad=5, categoria=HOGAR}
Producto{id=JIL426, nombre=Harina, precio=700.0, cantidad=250, categoria=ALIMENTOS}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
        System.out.println( "EL producto Buscado por id es: \n"
                + inventario1.buscarProductoPorId("abc123"));
```

```
EL producto Buscado por id es:
Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
        System.out.println("La categoria Mostrada es" + inventario1.filtrarPorCategoria(CategoriaProducto.
ELECTRONICA));
    }
```

```
La categoria Mostrada es[
 Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA},
 Producto{id=FGH456, nombre=Parlante, precio=350000.0, cantidad=5, categoria=ELECTRONICA}]
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
inventario1.eliminarProducto("QWR789");
inventario1.listarProducto();
```

```
Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA}

Producto{id=FGH456, nombre=Parlante, precio=350000.0, cantidad=5, categoria=ELECTRONICA}

Producto{id=TYU369, nombre=Sillon, precio=180000.0, cantidad=5, categoria=HOGAR}

Producto{id=JIL426, nombre=Harina, precio=700.0, cantidad=250, categoria=ALIMENTOS}
BUILD SUCCESSFUL (total time: 0 seconds)
```

6. Actualizar el stock de un producto existente.

```
inventario1.actualizarStock("jil426", 600);
inventario1.listarProducto();
```

```
Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA}
Producto{id=FGH456, nombre=Parlante, precio=350000.0, cantidad=5, categoria=ELECTRONICA}
Producto{id=TYU369, nombre=Sillon, precio=180000.0, cantidad=5, categoria=HOGAR}
Producto{id=JIL426, nombre=Harina, precio=700.0, cantidad=600, categoria=ALIMENTOS}
BUILD SUCCESSFUL (total time: 0 seconds)
```

7. Mostrar el total de stock disponible.

```
int StockDisponibleid = inventario1.obtenerTotalStock("abc123");
System.out.println("\nEl Stock diponible es Para el producto id abc123 es: "
        +StockDisponibleid + "\n");
```

```
El Stock diponible es Para el producto id abc123 es: 10

BUILD SUCCESSFUL (total time: 0 seconds)
```

8. Obtener y mostrar el producto con mayor stock.

```
System.out.println(inventario1.obtenerProductoConMayorStock());
```

```
Producto{id=JIL426, nombre=Harina, precio=700.0, cantidad=600, categoria=ALIMENTOS}
BUILD SUCCESSFUL (total time: 0 seconds)
```

9. Filtrar productos con precios entre $100 y $300000.

```
System.out.println("\nProductos con precios entre $100 y $300000:");
for (Producto p : inventario1.filtrarProductosPorPrecio(100, 300000)) {
    System.out.println(p);
}
```

```
Productos con precios entre $100 y $300000:
Producto{id=ABC123, nombre=MiniPimer, precio=87500.0, cantidad=10, categoria=ELECTRONICA}
Producto{id=TYU369, nombre=Sillon, precio=180000.0, cantidad=5, categoria=HOGAR}
Producto{id=JIL426, nombre=Harina, precio=700.0, cantidad=600, categoria=ALIMENTOS}
BUILD SUCCESSFUL (total time: 0 seconds)
```

10. Mostrar las categorías disponibles con sus descripciones.

```
System.out.println("\nCategorias Disponibles:");
System.out.println(inventario1.mostrarCategoriasDisponibles());
```

```
Categorias Disponibles:
[ELECTRONICA, HOGAR, ALIMENTOS]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ejercicio Propuesto 2: Biblioteca y Libros

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.

CLASE BIBLIOTECA:

```
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Set;

public class Biblioteca {

    private String nombre;
    private ArrayList<Libro> libros;

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

    public void agregarLibros(Libro libros) {
        this.libros.add(libros);

    }

    public void listarLibros() {
        for (Libro libro : libros) {
            System.out.println(libro);
            System.out.println("----------");
```

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

public Libro filtrarLibrosPorAnio(int anio) {
    for (Libro libro : libros) {
        if (libro.getAnioPublicacion() == anio) {
            return libro;
        }
    }
    return null;
}

public void eliminarLibro(String isbn) {
    libros.removeIf(libro -> isbn.equalsIgnoreCase(libro.getIsbn()));
}
```

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

 //Para evitar duplicados, lo ideal es usar un Set (como HashSet), que no permite elementos repetidos:
public ArrayList<Autor> mostrarAutoresDisponibles() {
    Set<Autor> autoresUnicos = new HashSet<>();
    for (Libro libro : libros) {
        if (libro.getAutor() != null) {
            autoresUnicos.add(libro.getAutor());
        }
    }
    return new ArrayList<>(autoresUnicos);
}
```

CLASE AUTOR:

```java
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 String getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }
```

```java
public String getNacionalidad() {
    return nacionalidad;
}

public void mostrarInfo() {
    System.out.println("La informacion del autor es: "
            + "\nId: " + getId()
            + "\nNombre: " + getNombre()
            + "\nNacionalidad: " + getNacionalidad());
}

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

CLASE LIBRO:

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

```java
public String getIsbn() {
    return isbn;
}

public String getTitulo() {
    return titulo;
}

public int getAnioPublicacion() {
    return anioPublicacion;
}

public void mostrarInfo() {
    System.out.println("La informacion del libro es: "
            + "\nIsbn: " + getIsbn()
            + "\nTitulo: " + getTitulo()
            + "\nAnio de Publicacion: " + getAnioPublicacion());
}
```

```java
    @Override
    public String toString() {
        return "Libro{" + "\nisbn=" + isbn + ", \ntitulo=" + titulo + ", \nanioPublicacion=" +
anioPublicacion + ", \n" + autor + '}';
    }
```

Tareas a realizar

1. Creamos una biblioteca.

```java
public static void main(String[] args) {
    Biblioteca b2 = new Biblioteca("Paramous");
```

2. Crear al menos tres autores

```java
Autor a1 = new Autor("abc1", "Frank Herbert", "EEUU");
Autor a2 = new Autor("fgh123", "Stephen King", "EEUU");
Autor a3 = new Autor("jkl345", "Dan Brown", "EEUU");
```

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

```java
Libro l1 = new Libro("AGFG1239", "Origen", 2017 , a3);
Libro l2 = new Libro("AJKLÑ123", "Dune", 1965, a1);
Libro l3 = new Libro("QWER123", "Doctor Sueño ", 2013, a2);
Libro l4 = new Libro("TYUUI567", "El Resplandor", 1977, a2);
Libro l5 = new Libro("HJKL566", "El simbolo perdido", 2009, a3);
```

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

```
b1.listarLibros();
```

```
run:
Libro{
isbn=AGFG1239,
titulo=Origen,
anioPublicacion=2017,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
----------
Libro{
isbn=AJKL123,
titulo=Dune,
anioPublicacion=1965,
Autor{
id=abc1,
nombre=Frank Herbert,
nacionalidad=EEUU}}
----------
Libro{
isbn=QWER123,
titulo=Doctor Sue o ,
anioPublicacion=2013,
Autor{
id=fgh123,
nombre=Stephen King,
nacionalidad=EEUU}}
----------
Libro{
isbn=TYUUI567,
titulo=El Resplandor,
anioPublicacion=1977,
Autor{
id=fgh123,
nombre=Stephen King,
nacionalidad=EEUU}}
----------
Libro{
isbn=HJKL566,
titulo=El simbolo perdido,
anioPublicacion=2009,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
----------
```

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

```
System.out.println(b1.buscarLibroPorIsbn("AGFG1239"));
```

```
Libro{
isbn=AGFG1239,
titulo=Origen,
anioPublicacion=2017,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
System.out.println(b1.filtrarLibrosPorAnio(2009));
```

```
Libro{
isbn=HJKL566,
titulo=El simbolo perdido,
anioPublicacion=2009,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
b1.eliminarLibro("TYUUI567");
b1.listarLibros();
```

```
Libro{
isbn=AGFG1239,
titulo=Origen,
anioPublicacion=2017,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
----------
Libro{
isbn=AJKL123,
titulo=Dune,
anioPublicacion=1965,
Autor{
id=abc1,
nombre=Frank Herbert,
nacionalidad=EEUU}}
```

```
----------
Libro{
isbn=QWER123,
titulo=Doctor Sueño ,
anioPublicacion=2013,
Autor{
id=fgh123,
nombre=Stephen King,
nacionalidad=EEUU}}
----------
Libro{
isbn=HJKL566,
titulo=El simbolo perdido,
anioPublicacion=2009,
Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}}
----------
BUILD SUCCESSFUL (total time: 0 se
```

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

```
System.out.println("La biblioteca tiene " +
b1.obtenerCantidadLibros() + " libros" );
```

```
Ejercicio 8
La biblioteca tiene 4 libros
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
System.out.println("Ejercicio 9");
System.out.println("Los autores Disponibles son: \n");
for (Autor a : b1.mostrarAutoresDisponibles()) {
    System.out.println(a + "\n");
}
```

```
Ejercicio 9
Los autores Disponibles son:

Autor{
id=abc1,
nombre=Frank Herbert,
nacionalidad=EEUU}

Autor{
id=fgh123,
nombre=Stephen King,
nacionalidad=EEUU}

Autor{
id=jkl345,
nombre=Dan Brown,
nacionalidad=EEUU}
```

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

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).

Clases a implementar

PROFESOR:

```java
public class Profesor {

    private String id;
    private String nombre;
    private String especialidad;
    private ArrayList<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 getEspecialidad() {
        return especialidad;
    }


    public String getId() {
        return id;
    }

    public void agregarCurso(Curso c) {
        if (!cursos.contains(c)) {
            cursos.add(c);
        }
    }
```

```java
public void eliminarCurso(Curso c) {
    cursos.remove(c);
}

public String getNombre() {
    return nombre;
}

public ArrayList<Curso> getCursos() {
    return cursos;
}

public void listarCursos() {
    for (Curso curso : cursos) {
        System.out.println("Nombre: " + curso.getNombre()
                + "\nCodigo: " + curso.getCodigo());
    }
}

public int cantidadCursos() {
    return cursos.size();
}

public void mostrarInfo() {
    for (Curso curso : cursos) {
        System.out.println(curso.getProfesor()
                + "\nCantidad de Cursos: " + cantidadCursos());
    }
}

@Override
public String toString() {
    return "Profesor{" + "\nid=" + id + ", \nnombre=" + nombre + ", \nespecialidad=" + especialidad + '}';
```

CURSO:

```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;
    }

    public void setProfesor(Profesor profesor) {
        this.profesor = profesor;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
```

```java
public Profesor getProfesor() {
    return profesor;
}

public void mostrarInfo() {
    System.out.println("Codigo del curso: " + getCodigo()
            + "\nNombre del Curso: " + getNombre()
            + "\n" + getProfesor());
}

@Override
public String toString() {
    return "Curso{" + "\ncodigo=" + codigo + ", \nnombre=" + nombre
            + ", profesor=" + (profesor != null ? profesor.getNombre() : "sin profesor")
            + '}';
}
```

UNIVERSIDAD:

```java
public class Universidad {

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

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

    public void agregarProfesor(Profesor p) {
        profesores.add(p);
    }

    public void agregarCurso(Curso c) {
        cursos.add(c);
    }
```

```java
public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
    Curso cursoEncontrado = null;
    Profesor profesorEncontrado = null;
    for (Curso curso : cursos) {
        if (curso.getCodigo().equalsIgnoreCase(codigoCurso)) {
            cursoEncontrado = curso;
            break;
        }
    }
    for (Profesor profesor : profesores) {
        if (profesor.getId().equalsIgnoreCase(idProfesor)) {
            profesorEncontrado = profesor;
            break;
        }
    }
    if (cursoEncontrado != null && profesorEncontrado != null) {
        cursoEncontrado.setProfesor(profesorEncontrado);
        profesorEncontrado.agregarCurso(cursoEncontrado);
    } else {
        System.out.println("Curso o profesor no encontrado.");
    }
}
```

```java
public void listarProfesores() {
    for (Profesor profesore : profesores) {
        System.out.println(profesore);

    }
}

public void listarCursos() {
    for (Curso curso : cursos) {
        System.out.println(curso);
    }
}

public Profesor buscarProfesorPorId(String id) {
    for (Profesor profesore : profesores) {
        if (id.equalsIgnoreCase(profesore.getId())) {
            return profesore;
        }
    }
    return null;
}
```

```java
public void eliminarProfesor(String id) {
    for (Curso curso : cursos) {
        if (curso.getProfesor() != null && curso.getProfesor().getId().equals(id)) {
            curso.setProfesor(null);
        }
    }
    Profesor p = buscarProfesorPorId(id);
    profesores.remove(p);
}
public Curso buscarCursoPorCodigo(String codigo) {
    for (Curso curso : cursos) {
        if (codigo.equalsIgnoreCase(curso.getCodigo())) {
            return curso;
        }
    }
    return null;
}
public void eliminarCurso(String codigo) {
    Curso curso = buscarCursoPorCodigo(codigo);
    for (Profesor profesor : profesores) {
        profesor.getCursos().remove(curso);
    }
    cursos.remove(curso);
}
```

```java
public void listarCursosPorProfesor() {
    System.out.println("---- Profesores -----");
    for (Profesor profesor : profesores) {
        System.out.println("Profesor: " + profesor.getNombre() + " (" + profesor.getId() + ")");
        for (Curso curso : profesor.getCursos()) {
            System.out.println("  - Curso: " + curso.getNombre() + " [" + curso.getCodigo() + "]");
        }
        System.out.println(); // Línea en blanco para separar
    }
}

public void mostrarReporte() {
    System.out.println("---- Reporte: Cantidad de cursos por profesor ----");
    for (Profesor profesor : profesores) {
        System.out.println("Profesor: " + profesor.getNombre() + " (" + profesor.getEspecialidad() + ")");
        System.out.println("Id : " +  profesor.getId());
        System.out.println("Cantidad de cursos asignados: " + profesor.cantidadCursos() + " cursos");
        for (Curso curso : profesor.getCursos()) {
            System.out.println("Nombre del Curso: "+ curso.getNombre());
            System.out.println("Codigo del curso: " + curso.getCodigo());
            System.out.println("------");
        }
        System.out.println(); // Línea en blanco para separar
```

Tareas a realizar

1. Crear al menos 3 profesores y 5 cursos.

```java
Profesor p1 = new Profesor("qwert123", "Carlos", "Electronica");
Profesor p2 = new Profesor("asdf345", "Juan", "Informatica");
Profesor p3 = new Profesor("jkl987", "Elvio", "Base de datos I");

Curso c1 = new Curso("aa00", "Programacion I");
Curso c2 = new Curso("ab01", "Electronica");
Curso c3 = new Curso("fa03", "Intro. Informatica");
Curso c4 = new Curso("a204", "BD I");
Curso c5 = new Curso("a709", "Analisis");
```

2. Agregar profesores y cursos a la universidad.

```
Universidad u1 = new Universidad("UTN");
u1.agregarProfesor(p1);
u1.agregarProfesor(p2);
u1.agregarProfesor(p3);

u1.agregarCurso(c1);
u1.agregarCurso(c2);
u1.agregarCurso(c3);
u1.agregarCurso(c4);
u1.agregarCurso(c5);
```

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

```
u1.asignarProfesorACurso("aa00", "asdf345");
u1.asignarProfesorACurso("fa03", "asdf345");
u1.asignarProfesorACurso("ab01", "qwert123");
u1.asignarProfesorACurso("a204", "jkl987");
u1.asignarProfesorACurso("a709", "asdf345");
```

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

```
u1.listarProfesores();
System.out.println("");
u1.listarCursos();
```

```
Profesor{
id=qwert123,
nombre=Carlos,
especialidad=Electronica}
Profesor{
id=asdf345,
nombre=Juan,
especialidad=Informatica}
Profesor{
id=jkl987,
nombre=Elvio,
especialidad=Base de datos I}
```

```
Curso{
codigo=aa00,
nombre=Programacion I, profesor=Juan}
Curso{
codigo=ab01,
nombre=Electronica, profesor=Carlos}
Curso{
codigo=fa03,
nombre=Intro. Informatica, profesor=Juan}
Curso{
codigo=a204,
nombre=BD I, profesor=Elvio}
Curso{
codigo=a709,
nombre=Analisis, profesor=Juan}
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```
 ---- Cursos -----
Curso{
codigo=aa00,
nombre=Programacion I, profesor=Juan}
Curso{
codigo=ab01,
nombre=Electronica, profesor=Carlos}
Curso{
codigo=fa03,
nombre=Intro. Informatica, profesor=Juan}
Curso{
codigo=a204,
nombre=BD I, profesor=sin profesor}
Curso{
codigo=a709,
nombre=Analisis, profesor=Juan}
 ---- Profesores -----
Profesor{
id=qwert123,
nombre=Carlos,
especialidad=Electronica}
Profesor{
id=asdf345,
nombre=Juan,
especialidad=Informatica}
```

```
u1.eliminarProfesor("jkl987");
System.out.println(" ---- Cursos -----");
u1.listarCursos();
System.out.println(" ---- Profesores -----");
u1.listarProfesores();
```

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

```
u1.eliminarCurso("ab01");
System.out.println(" ---- Cursos -----");
u1.listarCursos();
u1.listarCursosPorProfesor();
```

```
 ---- Cursos -----
Curso{
codigo=aa00,
nombre=Programacion I, profesor=Juan}
Curso{
codigo=fa03,
nombre=Intro. Informatica, profesor=Juan}
Curso{
codigo=a204,
nombre=BD I, profesor=sin profesor}
Curso{
codigo=a709,
nombre=Analisis, profesor=Juan}
---- Profesores -----
Profesor: Carlos (qwert123)

Profesor: Juan (asdf345)
  - Curso: Programacion I [aa00]
  - Curso: Intro. Informatica [fa03]
  - Curso: Analisis [a709]
```

7. Remover un profesor y dejar profesor = null,

```
 ---- Cursos -----
Curso{
codigo=aa00,
nombre=Programacion I, profesor=Juan}
Curso{
codigo=ab01,
nombre=Electronica, profesor=Carlos}
Curso{
codigo=fa03,
nombre=Intro. Informatica, profesor=Juan}
Curso{
codigo=a204,
nombre=BD I, profesor=sin profesor}
Curso{
codigo=a709,
nombre=Analisis, profesor=Juan}
 ---- Profesores -----
Profesor{
id=qwert123,
nombre=Carlos,
especialidad=Electronica}
Profesor{
id=asdf345,
nombre=Juan,
especialidad=Informatica}
```

```
u1.eliminarProfesor("jkl987");
System.out.println(" ---- Cursos -----");
u1.listarCursos();
System.out.println(" ---- Profesores -----");
u1.listarProfesores();
```

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

```
REPORTE INICIAL
---- Reporte: Cantidad de cursos por profesor ----
Profesor: Carlos (Electronica)
Id : qwert123
Cantidad de cursos asignados: 1 cursos
Nombre del Curso: Electronica
Codigo del curso: ab01
------

Profesor: Juan (Informatica)
Id : asdf345
Cantidad de cursos asignados: 3 cursos
Nombre del Curso: Programacion I
Codigo del curso: aa00
------
Nombre del Curso: Intro. Informatica
Codigo del curso: fa03
------
Nombre del Curso: Analisis
Codigo del curso: a709
------

Profesor: Elvio (Base de datos I)
Id : jkl987
Cantidad de cursos asignados: 1 cursos
Nombre del Curso: BD I
Codigo del curso: a204
------
```

```
REPORTE FINAL

---- Reporte: Cantidad de cursos por profesor ----
Profesor: Carlos (Electronica)
Id : qwert123
Cantidad de cursos asignados: 0 cursos

Profesor: Juan (Informatica)
Id : asdf345
Cantidad de cursos asignados: 3 cursos
Nombre del Curso: Programacion I
Codigo del curso: aa00
------
Nombre del Curso: Intro. Informatica
Codigo del curso: fa03
------
Nombre del Curso: Analisis
Codigo del curso: a709
------
```