



## PROGRAMACIÓN II

### Trabajo Práctico 6: Colecciones y Sistema de Stock

Alumno: Victor Barroeta

DNI: 95805903

Comisión: M2025 – 2

Matricula: 100393

Link de Github:

[https://github.com/victormbar/UTN\\_Prog2\\_VictorBarroeta/tree/main/javaUTN\\_TPs/src/TPs\\_JAVA\\_TP6\\_1](https://github.com/victormbar/UTN_Prog2_VictorBarroeta/tree/main/javaUTN_TPs/src/TPs_JAVA_TP6_1)

[https://github.com/victormbar/UTN\\_Prog2\\_VictorBarroeta/tree/main/javaUTN\\_TPs/src/TPs\\_JAVA\\_TP6\\_2](https://github.com/victormbar/UTN_Prog2_VictorBarroeta/tree/main/javaUTN_TPs/src/TPs_JAVA_TP6_2)

[https://github.com/victormbar/UTN\\_Prog2\\_VictorBarroeta/tree/main/javaUTN\\_TPs/src/TPs\\_JAVA\\_TP6\\_3](https://github.com/victormbar/UTN_Prog2_VictorBarroeta/tree/main/javaUTN_TPs/src/TPs_JAVA_TP6_3)

### OBJETIVO GENERAL

Desarrollar estructuras de datos dinámicas en Java mediante el uso de colecciones ([ArrayList](#)) y enumeraciones ([enum](#)), implementando un sistema de stock con funcionalidades progresivas que refuerzan conceptos clave de la programación orientada a objetos..

### MARCO TEÓRICO

Concepto	Aplicación en el proyecto
ArrayList	Estructura principal para almacenar productos en el inventario.
Enumeraciones (enum)	Representan las categorías de productos con valores predefinidos.
Relaciones 1 a N	Relación entre Inventario (1) y múltiples Productos (N).



Métodos en enum	Inclusión de descripciones dentro del enum para mejorar legibilidad.
Ciclo for-each	Recorre colecciones de productos para listado, búsqueda o filtrado.
Búsqueda y filtrado	Por ID y por categoría, aplicando condiciones.
Ordenamientos y reportes	Permiten organizar la información y mostrar estadísticas útiles.
Encapsulamiento	Restringir el acceso directo a los atributos de una clase

## Caso Práctico 1

### 1. Descripción general

Se debe desarrollar un sistema de stock que permita gestionar productos en una tienda, controlando su disponibilidad, precios y categorías. La información se modelará utilizando clases, colecciones dinámicas y enumeraciones en Java.

### 2. Clases a implementar **Clase Producto**

#### Atributos:

- **id (String)** → Identificador único del producto.
- **nombre (String)** → Nombre del producto.
- **precio (double)** → Precio del producto.
- **cantidad (int)** → Cantidad en stock.
- **categoria (CategoriaProducto)** → Categoría del producto.

#### Métodos:

- **mostrarInfo()** → Muestra en consola la información del producto.

#### Enum CategoriaProducto

#### Valores:

- ALIMENTOS
- ELECTRONICA
- ROPA
- HOGAR

**Método adicional:**

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



## Clase Inventario

Atributo:

- **ArrayList<Producto> productos** Métodos requeridos:
- **agregarProducto(Producto p)**
- **listarProductos()**
- **buscarProductoPorId(String id)**
- **eliminarProducto(String id)**
- **actualizarStock(String id, int nuevaCantidad)**
- **filtrarPorCategoria(CategoríaProducto categoria)**
- **obtenerTotalStock()**
- **obtenerProductoConMayorStock()**
- **filtrarProductosPorPrecio(double min, double max)**
- **mostrarCategoríasDisponibles()**

### 3. Tareas a realizar

1. Crear al menos cinco productos con diferentes categorías y agregarlos al inventario.
2. Listar todos los productos mostrando su información y categoría.
3. Buscar un producto por ID y mostrar su información.
4. Filtrar y mostrar productos que pertenezcan a una categoría específica.
5. Eliminar un producto por su ID y listar los productos restantes.
6. Actualizar el stock de un producto existente.
7. Mostrar el total de stock disponible.
8. Obtener y mostrar el producto con mayor stock.
9. Filtrar productos con precios entre \$1000 y \$3000.
10. Mostrar las categorías disponibles con sus descripciones.

## CONCLUSIONES ESPERADAS

- Comprender el uso de **this** para acceder a atributos de instancia.
- Aplicar constructores sobrecargados para flexibilizar la creación de objetos.
- Implementar métodos con el mismo nombre y distintos parámetros.
- Representar objetos con **toString()** para mejorar la depuración.
- Diferenciar y aplicar atributos y métodos estáticos en Java.



- Reforzar el diseño modular y reutilizable mediante el paradigma orientado a objetos.

## Resultados

```
run:  
Producto Hamburguesa agregado exitosamente  
Producto Pizza agregado exitosamente  
Producto Sandwich de Pollo agregado exitosamente  
Producto iPhone 14 agregado exitosamente  
Producto Remera Adidas T40 Negra agregado exitosamente  
Producto Notebook Asus 16'' agregado exitosamente  
Producto Silla gamer + reposa pies agregado exitosamente  
Producto Escritorio gamer 2x3,5 agregado exitosamente  
Producto Zapatillas Nike SK8 T42 agregado exitosamente  
  
-----Listar Productos-----  
  
{  
id=1  
nombre=Hamburguesa  
precio=10000.0  
cantidad=20  
categoria=ALIMENTOS}  
{  
id=2  
nombre=Pizza  
precio=15000.0  
cantidad=100  
categoria=ALIMENTOS}  
{  
id=3  
nombre=Sandwich de Pollo  
precio=20000.0  
cantidad=20
```



```
{  
    id=3  
    nombre=Sandwich de Pollo  
    precio=20000.0  
    cantidad=20  
    categoria=ALIMENTOS}  
{  
    id=4  
    nombre=iPhone 14  
    precio=1000000.0  
    cantidad=10  
    categoria=ELECTRONICA}  
{  
    id=5  
    nombre=Remera Adidas T40 Negra  
    precio=30000.0  
    cantidad=200  
    categoria=ROPA}  
{  
    id=6  
    nombre=Notebook Asus 16''  
    precio=650000.0  
    cantidad=40  
    categoria=ELECTRONICA}  
{  
    id=7  
    nombre=Silla gamer + reposa pies  
    precio=450000.0  
    cantidad=15
```

```
id=7  
nombre=Silla gamer + reposa pies  
precio=450000.0  
cantidad=15  
categoria=HOGAR}  
{  
    id=8  
    nombre=Escritorio gamer 2x3,5  
    precio=73000.0  
    cantidad=20  
    categoria=HOGAR}  
{  
    id=9  
    nombre=Zapatillas Nike SK8 T42  
    precio=90000.0  
    cantidad=20  
    categoria=ROPA}  
  
-----Buscar Producto ID-----  
  
Producto encontrado:  
{  
    id=4  
    nombre=iPhone 14  
    precio=1000000.0  
    cantidad=10  
    categoria=ELECTRONICA}  
  
-----Filtrar Por Categoría-----
```



```
----Filtrar Por Categoría----  
  
Filtrando por categoría: ROPA  
{  
id=5  
nombre=Remera Adidas T40 Negra  
precio=30000.0  
cantidad=200  
categoria=ROPA}  
{  
id=9  
nombre=Zapatillas Nike SK8 T42  
precio=90000.0  
cantidad=20  
categoria=ROPA}  
  
----Eliminar ID Y Mostrar----  
  
Producto con ID 4 eliminado correctamente.  
{  
id=1  
nombre=Hamburguesa  
precio=10000.0  
cantidad=20  
categoria=ALIMENTOS}  
{  
id=2  
nombre=Pizza  
precio=15000.0
```

```
id=2  
nombre=Pizza  
precio=15000.0  
cantidad=100  
categoria=ALIMENTOS}  
{  
id=3  
nombre=Sandwich de Pollo  
precio=20000.0  
cantidad=20  
categoria=ALIMENTOS}  
{  
id=5  
nombre=Remera Adidas T40 Negra  
precio=30000.0  
cantidad=200  
categoria=ROPA}  
{  
id=6  
nombre=Notebook Asus 16''  
precio=650000.0  
cantidad=40  
categoria=ELECTRONICA}  
{  
id=7  
nombre=Silla gamer + reposa pies  
precio=450000.0  
cantidad=15  
categoria=HOGAR}
```



```
id=8
nombre=Escritorio gamer 2x3,5
precio=73000.0
cantidad=20
categoria=HOGAR}
{
id=9
nombre=Zapatillas Nike SK8 T42
precio=90000.0
cantidad=20
categoria=ROPA}

-----Actualizar Stock-----

Producto encontrado:
Stock del producto Remera Adidas T40 Negra actualizado a 50 unidades.

-----Mostrar Total Stock-----

Stock total disponible: 285

-----Producto Mayor Stock-----

El producto con mayor stock es PIZZA con un total de 100 unidades.

-----Filtrar Precios-----

Productos con precio entre $1000.0 y $3000.0:
No se encontraron productos en ese rango de precios.

-----Mostrar Descripcion Categorias-----

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

## 📝 Nuevo Ejercicio Propuesto 2: Biblioteca y Libros

### 1. Descripción general

Se debe desarrollar un sistema para gestionar una **biblioteca**, en la cual se registren los libros disponibles y sus autores. La relación central es de **composición 1 a N**: una Biblioteca contiene múltiples Libros, y cada Libro pertenece obligatoriamente a una Biblioteca. Si la Biblioteca se elimina, también se eliminan sus Libros.

### 2. Clases a implementar

#### **Clase Autor**

#### **Atributos:**

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

### Métodos:

- **mostrarInfo()** → Muestra la información del autor en consola.

## Clase Libro

### Atributos:

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

### Métodos:

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

## Clase Biblioteca

### Atributo:

- **String nombre**
- **List<Libro> libros** → Colección de libros de la biblioteca.

### Métodos requeridos:

- **agregarLibro(String isbn, String titulo,int anioPublicacion, Autor autor)**
- **listarLibros()**
- **buscarLibroPorIsbn(String isbn)**
- **eliminarLibro(String isbn)**
- **obtenerCantidadLibros()**
- **filtrarLibrosPorAnio(int anio)**
- **mostrarAutoresDisponibles()**

### 3. Tareas a realizar

1. Creamos una biblioteca.
2. Crear al menos tres autores
3. Agregar 5 libros asociados a alguno de los Autores a la biblioteca.
4. Listar todos los libros con su información y la del autor.
5. Buscar un libro por su ISBN y mostrar su información.
6. Filtrar y mostrar los libros publicados en un año específico.
7. Eliminar un libro por su ISBN y listar los libros restantes.
8. Mostrar la cantidad total de libros en la biblioteca.
9. Listar todos los autores de los libros disponibles en la biblioteca.

### Conclusiones esperadas

- Comprender la **composición 1 a N** entre Biblioteca y Libro.
- Reforzar el manejo de **colecciones dinámicas** (ArrayList).
- Practicar el uso de **métodos de búsqueda, filtrado y eliminación**.
- Mejorar la modularidad aplicando el paradigma de **programación orientada a objetos**.

### Resultados obtenidos



```
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
```



```
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
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: 0 seconds)
```

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

### 1. Descripción general

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

- Desde **Curso** se accede a su **Profesor**.
  - Desde **Profesor** se accede a la **lista de Cursos** que dicta.
- Además, existe la clase **Universidad** que administra el alta/baja y consulta de profesores y cursos.

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

## 2. Clases a implementar

### Clase Profesor

**Atributos:**

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

**Métodos sugeridos:**

- **agregarCurso(Curso c)** → Agrega el curso a su lista si no está y sincroniza el lado del curso.
- **eliminarCurso(Curso c)** → Quita el curso y sincroniza el lado del curso (dejar profesor en null si corresponde).
- **listarCursos()** → Muestra códigos y nombres.
- **mostrarInfo()** → Imprime datos del profesor y cantidad de cursos.

### Clase Curso

**Atributos:**

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

**Métodos sugeridos:**

- **setProfesor(Profesor p)** → Asigna/cambia el profesor **sincronizando ambos lados**:
  - Si tenía profesor previo, quitarse de su lista.
- **mostrarInfo()** → Muestra código, nombre y nombre del profesor (si tiene).

### Clase Universidad

**Atributos:**

- **String nombre**



- `List<Profesor> profesores`
- `List<Curso> cursos`

#### Métodos requeridos:

- `agregarProfesor(Profesor p)`
- `agregarCurso(Curso c)`
- `asignarProfesorACurso(String codigoCurso, String idProfesor)` → Usa `setProfesor` del curso.
- `listarProfesores() / listarCursos()`
- `buscarProfesorPorId(String id)`
- `buscarCursoPorCodigo(String codigo)`
- `eliminarCurso(String codigo)` → Debe **romper la relación** con su profesor si la hubiera.
- `eliminarProfesor(String id)` → Antes de remover, dejar null los cursos que dictaba.

#### Tareas a realizar

1. Crear al menos 3 profesores y 5 cursos.
2. Agregar profesores y cursos a la universidad.
3. Asignar profesores a cursos usando `asignarProfesorACurso(...)`.
4. Listar cursos con su profesor y profesores con sus cursos.
5. Cambiar el profesor de un curso y verificar que ambos lados quedan sincronizados.
6. Remover un curso y confirmar que ya **no** aparece en la lista del profesor.
7. Remover un profesor y dejar profesor = null,
8. Mostrar un reporte: cantidad de cursos por profesor.

#### Conclusiones esperadas

- Diferenciar **bidireccionalidad** de una relación unidireccional (navegación desde ambos extremos).
- Mantener **invariantes de asociación** (coherencia de referencias) al **agregar, quitar o reasignar**.
- Practicar colecciones (`ArrayList`), búsquedas y operaciones de alta/baja.
- Diseñar métodos “seguros” que **sincronicen** los dos lados siempre.

#### Resultados obtenidos:



```
Los profesores 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
```



```
//////////  
Reasignamos el curso de Programacion 1 al docente Leandro De Angelis  
La universidad 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 profesores de la universidad UTN cuenta con 3 docentes:  
El profesor con id: 3  
nombre: Carlos Frede  
especialidad: Financiera  
ejercerse en 1 cursos.  
-----  
El profesor con id: 2  
nombre: Ariel Enferrel  
especialidad: Programacion  
ejercerse en 2 cursos.  
-----  
El profesor con id: 1
```



```
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
```



```
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.
```