Trabajo Practico N°6

de

Programación II

Estudiante: Tobias Leiva

Unidad: Colecciones

Universidad Tecnológica Nacional

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. 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.

Clase Producto:

```
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 CategoriaProducto getCategoria() {
        return categoria;
    }
    public void setCantidad(int cantidad) {
        this.cantidad = cantidad;
    }
    public int getCantidad() {
        return cantidad;
    }
    public double getPrecio() {
        return precio;
    }

    public void mostrarInfo(){
        CategoriaProducto[] categoria = CategoriaProducto.values();
        for (int i = 0; i < categoria.length; i++) {
            System.out.println();
        }
    }
    @Override
    public String toString() {
        return "Producto{" + "id=" + id + ", nombre=" + nombre + ",
precio=" + precio + ", cantidad=" + cantidad + ", categoria=" +
categoria + '}';
    }
}
```

Clase Inventario:

```
public class Inventario {
```

```
    private ArrayList<Producto> productos;

    public Inventario(){
        this.productos = new ArrayList<>();
    }

    public void agregarProducto(Producto p){
        this.productos.add(p);
    }

    public void listarProductos(){
        for (Producto producto : productos) {
            System.out.println(producto);
        }
    }

    public Producto buscarProductoPorId(String id){
        int i = 0;
        Producto idEncontrar = null;
        while (i < productos.size() &&
!this.productos.get(i).getId().equals(id)) {
            i++;
        }
        if (i < productos.size()) {
            idEncontrar = this.productos.get(i);
        }
        return idEncontrar;
    }

    public void filtrarPorCategoria(CategoriaProducto categoria){
        for (Producto p : productos) {
            if (p.getCategoria() == categoria) {
                System.out.println(p);
```

```
            }
        }
    }

    public Producto eliminarProducto(String id){
        int i = 0;
        Producto eliminarProd = null;
        while (i < productos.size() &&
!this.productos.get(i).getId().equals(id)) {
            i++;
        }
        if (i < productos.size()) {
            eliminarProd = this.productos.remove(i);
        }
        return eliminarProd;
    }

    public void actualizarStock(String id, int nuevaCantidad){
        for (Producto p : productos) {
            if(p.getId().equals(id)){
                p.setCantidad(nuevaCantidad);
                System.out.println("Stock de " + p.getId() + "
actualizado a " +  nuevaCantidad);
                return;
            }
        }
    }

    public int obtenerTotalStock(){
        int total = 0;
        for (Producto producto : productos) {
            total = total + producto.getCantidad();
        }
```

```
        return total;
    }
    public ArrayList<Producto> obtenerProductoConMayorStock(){
        ArrayList<Producto> listaMax = new ArrayList<>();
        int max = -1;
        for (Producto p : productos) {
            if (p.getCantidad() == max) {
                listaMax.add(p);
            }else if (p.getCantidad() > max) {
                max = p.getCantidad();
                listaMax.clear();
                listaMax.add(p);
            }
        }
        return listaMax;
    }

    public ArrayList<Producto> filtrarProductosPorPrecios(double min,
double max){
        ArrayList<Producto> listaP = new ArrayList<>();
        for (Producto p : productos) {
            if (p.getPrecio() >= min && p.getPrecio() <= max) {
                listaP.add(p);
            }
        }
        return listaP;
    }
    public void mostrarCategoriasDisponibles(){
        CategoriaProducto[] categoria = CategoriaProducto.values();
        for (CategoriaProducto c : categoria) {
            System.out.println(c);
        }
    }
```

```
}
```

Clase enum CategoriaProducto:

```
public enum CategoriaProducto {
    ALIMENTOS("Productos comestibles"),
    ELECTRONICA("Dispositivos electronicos"),
    ROPA("Prendas de vestir"),
    HOGAR("Articulos para el hogar");

    private final String descripcion;

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


    public String getDescripcion() {
        return descripcion;
    }

    @Override
    public String toString() {
        return "CategoriaProducto{" + "ordinal=" + ordinal() + ",
name=" + name() + ", descripcion=" + descripcion + '}';
    }
}
```

Clase Principal:

```
public class Principal {
    public static void main(String[] args) {
        System.out.println("-----Tarea 1 y 2----");
        Inventario i = new Inventario();
```

```
        i.agregarProducto(new Producto("VGTR13B","Sillon", 7500, 1,
CategoriaProducto.HOGAR));

        i.agregarProducto(new Producto("54664B","Celular", 12999, 1,
CategoriaProducto.ELECTRONICA));

        i.agregarProducto(new Producto("425FDC","Manzana", 1000, 12,
CategoriaProducto.ALIMENTOS));

        i.agregarProducto(new Producto("KHJK75","Pantalon", 3000, 10,
CategoriaProducto.ROPA));

        i.agregarProducto(new Producto("ASDA312","Remera", 1000, 5,
CategoriaProducto.ROPA));


        i.listarProductos();



////////////////////////////////////////////////////////////////////////
/////////////

        System.out.println("==============================");

        System.out.println("-----Tarea 3-----");

        Producto id = i.buscarProductoPorId("425FDC");

        if (id == null) {

            System.out.println("No se encontro");

        } else {

            System.out.println("Se encontro: " + id );

        }



////////////////////////////////////////////////////////////////////////
//

        System.out.println("==============================");

        System.out.println("-----Tarea 4-----");

        System.out.println("Filtracion por Ropa");

        i.filtrarPorCategoria(CategoriaProducto.ROPA);

        System.out.println("------------------");

        System.out.println("Filtracion por Alimento");

        i.filtrarPorCategoria(CategoriaProducto.ALIMENTOS);



////////////////////////////////////////////////////////////////////////
//////////////////////
```

```
        System.out.println("================================");

        System.out.println("-----Tarea 5-----");

        i.eliminarProducto("KHJK75");

        i.listarProductos();



////////////////////////////////////////////////////////////////////////////
/////////////////

        System.out.println("================================");

        System.out.println("-----Tarea 6-----");

        i.actualizarStock("425FDC", 100);

        System.out.println("Se actualizo este producto: " +
i.buscarProductoPorId("425FDC") );



////////////////////////////////////////////////////////////////////////////
//////////////////////

        System.out.println("================================");

        System.out.println("-----Tarea 7-----");

        System.out.println("Total de Stock entre los productos: " +
i.obtenerTotalStock());



////////////////////////////////////////////////////////////////////////////
//////////////////////

        System.out.println("================================");

        System.out.println("-----Tarea 8-----");

        System.out.println("El producto con mayor stock es: " +
i.obtenerProductoConMayorStock());



////////////////////////////////////////////////////////////////////////////
//////////////////////

        System.out.println("================================");

        System.out.println("-----Tarea 9-----");

        System.out.println("Los productos que estan entre $1000 y $3000: ");

        for (Producto p : i.filtrarProductosPorPrecios(1000, 3000)) {

            System.out.println(p);

        //Producto Pantalon fue eliminado antes y no va aparecer
```

```
        }

////////////////////////////////////////////////////////////////////////////
//////////////////////

        System.out.println("==============================");

        System.out.println("-----Tarea 10-----");

        i.mostrarCategoriasDisponibles();

    }

}
```

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. 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.

Clase 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;
    }

    public void mostrarInfo(){
        System.out.println("Identificacion del libro: " + isbn);
        System.out.println("Titulo: " + titulo);
        System.out.println("Publicado en: " + anioPublicacion);
        System.out.println("Autor del libro: " + autor.mostrarInfo());
    }

    public String getIsbn() {
        return isbn;
    }
    public int getAnioPublicacion() {
        return anioPublicacion;
    }
    public Autor getAutor() {
        return autor;
    }

    @Override
    public String toString() {
        return "Libro{" + "isbn=" + isbn + ", titulo=" + titulo + ",
anioPublicacion=" + anioPublicacion + ", autor=" + autor + '}';
    }
}
```

Clase Biblioteca:

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

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

    public void agregarLibro(String isbn, String titulo, int anioPublicacion,
Autor autor){
        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
    }
    public void listarLibros(){
        for (Libro libro : libros) {
            libro.mostrarInfo();
            System.out.println("========================");
        }
    }
    public Libro buscarLibroPorisbn(String isbn){
        Libro libroBuscado = null;
        int i = 0;
        while(i < libros.size() &&
!this.libros.get(i).getIsbn().equalsIgnoreCase(isbn)){
            i++;
        }
        if (i < libros.size()) {
            libroBuscado = this.libros.get(i);
        }
        return libroBuscado;
    }
    public void filtrarLibrosPorAnio(int anio){
        for (Libro libro : libros) {
            if (libro.getAnioPublicacion() == anio) {
                System.out.println("Libros encontrados por anio: " + libro);
            }
        }
    }
```

```
    public boolean eliminarLibro(String isbn){
        for (Libro libro : libros) {
            if (libro.getIsbn().equalsIgnoreCase(isbn)) {
                libros.remove(libro);
                System.out.println("El libro removido fue: " + libro);
                System.out.println("---------------------------------");
                return true;
            }
        }
        return false;
    }
    public int obtenerCantidadLibros(){
        int total = 0;
        for (Libro libro : libros) {
            total = libros.size();
        }
        return total;
    }

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

    @Override
    public String toString() {
        return "Biblioteca{" + "nombre=" + nombre + ", libros=" + libros +
'}';
    }
}
```

Clase 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;
    }
    public String mostrarInfo(){
        return "\nId: " + id + "\nNombre: " + nombre + "\nNacionalidad: " +
nacionalidad;
    }
    @Override
    public String toString() {
        return "Autor{" + "id=" + id + ", nombre=" + nombre + ",
nacionalidad=" + nacionalidad + '}';
    }
}
```

Clase Principal:

```
public class Principal {
    public static void main(String[] args) {
        //Tarea 1: Creamos una Biblioteca
        Biblioteca b = new Biblioteca("Biblioteca A");

        //Tarea 2: Crear al menos tres autores.
        Autor cervantes = new Autor("1", "Miguel de Cervantes", "Espanol");
        Autor aaron = new Autor("2", "Aaron", "Estados Unidos");
        Autor tobias = new Autor("3", "Tobias", "Argentina");

        //Tarea 3: Agregar 5 libros a la biblioteca y asociar cada libro a un
Autor
        b.agregarLibro("BAAS22", "Don Quijote", 1605, cervantes);
        b.agregarLibro("KCB123", "The Dragon Prince", 2030, aaron);
```

```
        b.agregarLibro("BCV876", "Esfuerzo", 2030, tobias);
        b.agregarLibro("VBGH123", "La Galatea", 1585, cervantes);
        b.agregarLibro("JUYK33", "La espanola inglesa", 1614, cervantes);

        //Tarea 4: Listar todos los libros con su informacion
        b.listarLibros();

        //Tarea 5: Buscar libro por ISBN y mostrar su informacion
        Libro libro = b.buscarLibroPorisbn("JUYK33");
        if (libro == null) {
            System.out.println("No se encontro");
        } else {
            System.out.println("Se encontro: " + libro);
            System.out.println("==============================");
        }

        //Tarea 6: Filtrar y mostrar los libros mediante un año especifico
        b.filtrarLibrosPorAnio(2030);
        System.out.println("==============================");

        //Tarea 7: Eliminar un libro por su ISBN y listar los libros
restantes
        b.eliminarLibro("JUYK33");
        b.listarLibros();

        //Tarea 8: Mostrar la cantidad total de libros en la biblioteca
        System.out.println("La cantidad total de libros: " +
b.obtenerCantidadLibros());
        System.out.println("==============================");

        //Tarea 9: Listar todos los autores de los libros disponibles en la
biblioteca
        b.mostrarAutoresDisponibles();
    }
}
```

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:

2.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.

Clase Profesor:

```
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;
        cursos = new ArrayList<>();
    }

    public String getNombre() {
        return nombre;
    }

    public String getId() {
```

```
    return id;
}

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

public void mostrarInfo(){
    System.out.println("Id profesor: "+ id);
    System.out.println("Nombre: "+ nombre);
    System.out.println("Especialiadad: " + especialidad );
    System.out.println("Cursos asignados: "+ cursos.size());
    System.out.println("==================================");
}
public void agregarCurso(Curso c){
    if (!cursos.contains(c)) {
        cursos.add(c);
        if (c.getProfesor() != this) {
            c.setProfesor(this);
        }
    }
}
public void eliminarCurso(Curso c){
    if (cursos.contains(c)) {
        cursos.remove(c);
        if (c.getProfesor() == this) {
            c.setProfesor(null);
        }
    }
}
public void listarCursos(){
    for (Curso c : cursos) {
        System.out.println("- " + c.getCodigo() + " | " + c.getNombre());
    }
```

```
    }

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

Clase Universidad:

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

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

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

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

    public void asignarProfesorACurso(String codigoCurso, String
idProfesor){
        Curso curso = buscarCursoPorCodigo(codigoCurso);
        Profesor profe = buscarProfesorPorId(idProfesor);
        if (curso != null && profe != null) {
```

```
            curso.setProfesor(profe);
        }
    }

    public Profesor buscarProfesorPorId(String id){
        Profesor p = null;
        int i = 0;
            while(i < profesores.size() &&
!this.profesores.get(i).getId().equalsIgnoreCase(id)){
            i++;
        }
        if (i < profesores.size()) {
            p = this.profesores.get(i);
        }
        return p;
    }
    public Curso buscarCursoPorCodigo(String codigo){
        Curso c = null;
        int i = 0;
            while(i < cursos.size() &&
!this.cursos.get(i).getCodigo().equalsIgnoreCase(codigo)){
            i++;
        }
        if (i < cursos.size()) {
            c = this.cursos.get(i);
        }
        return c;
    }
    public void listarProfesor(){
        for (Profesor p : profesores) {
            p.mostrarInfo();
        }
    }
```

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

        }
    }
    public void eliminarProfesor(String id){
        Profesor p = buscarProfesorPorId(id);
        if (p != null) {
            for (Curso c : new ArrayList<>(p.getCursos())) {
                c.setProfesor(null);
            }
            profesores.remove(p);
            System.out.println("Profesor eliminado: "+ id);
        }
    }
    public void eliminarCurso(String codigo) {
        Curso c = buscarCursoPorCodigo(codigo);
        if (c != null) {
            if (c.getProfesor() != null) {
                c.setProfesor(null);
            }
            cursos.remove(c);
            System.out.println("Curso eliminado: " + codigo);
        }
    }
    public void mostrarReporteCursosPorProfesor() {
        System.out.println("\n--- Reporte: Cursos por Profesor ---");
        for (Profesor p : profesores) {
            System.out.println(p.getNombre() + " dicta " +
p.getCursos().size() + " curso(s).");
        }
    }
```

```
    @Override
    public String toString() {
        return "Universidad{" + "nombre=" + nombre + ", profesores=" +
profesores + ", cursos=" + cursos + '}';
    }
}
```

Clase Curso:

```
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 p) {
        if (this.profesor == p) {
            return;
        }
        if (this.profesor != null) {
            this.profesor.eliminarCurso(this);
        }

        this.profesor = p;

        if (p != null && !p.getCursos().contains(this)) {
            p.agregarCurso(this);
        }
    }

    public Profesor getProfesor() {
```

```
        return profesor;
    }

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void mostrarInfo(){
        System.out.println("Codigo: "+ codigo);
        System.out.println("Nombre: "+ nombre);
        if (profesor != null) {
            System.out.println("Profesor: " + profesor.getNombre());
        }else{
            System.out.println("Profesor sin asignar");
        }
        System.out.println("================================");

    }

    @Override
    public String toString() {
        return "Curso{" + "codigo=" + codigo + ", nombre=" + nombre + ",
profesor=" + profesor + '}';
    }
}
```

Clase Principal:

```
public class Principal {
    public static void main(String[] args) {
        //Tarea 1: crear al menos 3 profesores y 5 cursos
        Profesor javier = new Profesor("NMCBX", "Javier", "Ciencias");
        Profesor luciano = new Profesor("IOKLJ8", "Luciano", "Matematicas");
```

```
Profesor ale = new Profesor("OTR123", "Alejandro", "Programacion");

Curso a1 = new Curso("A1", "Curso A");
Curso b2 = new Curso("B2", "Curso B");
Curso c3 = new Curso("C3", "Curso C");
Curso d4 = new Curso("D4", "Curso D");
Curso f5 = new Curso("F5", "Curso F");

//Tarea 2: agregar profesores y cursos a la universidad
Universidad utn = new Universidad("UTN");

//Cursos
utn.agregarCurso(a1);
utn.agregarCurso(b2);
utn.agregarCurso(c3);
utn.agregarCurso(d4);
utn.agregarCurso(f5);

//Profesores
utn.agregarProfesor(ale);
utn.agregarProfesor(luciano);
utn.agregarProfesor(javier);

//Tarea 3: Asignar profesores a cursos
utn.asignarProfesorACurso("A1", "OTR123");
utn.asignarProfesorACurso("B2", "IOKLJ8");
utn.asignarProfesorACurso("C3", "NMCBX");
utn.asignarProfesorACurso("D4", "OTR123");
utn.asignarProfesorACurso("F5", "NMCBX");

//Tarea 4: Listar cursos con su profesor y profesores con sus cursos
utn.listarProfesor();
System.out.println("////////////////////////////////");
System.out.println();
```

```
        System.out.println("////////////////////////////////");
        utn.listarCursos();
        System.out.println("////////////////////////////////");
        System.out.println("---------------------------------");
        System.out.println("////////////////////////////////");

        //Tarea 5: Cambiar el profesor de un curso y verificar que ambos
lados quedan sincronizados
        a1.setProfesor(luciano);

        utn.listarProfesor();

        System.out.println("Cursos de Luciano:");
        luciano.listarCursos();
        System.out.println("Cursos de Ale:");
        ale.listarCursos();

        System.out.println("////////////////////////////////");
        System.out.println("---------------------------------");
        System.out.println("////////////////////////////////");

        //Tarea 6: Remover un curso y confirmar que ya no aparece en la lista
del profesor.
        System.out.println("\n--- Eliminando curso A1 ---");
        utn.eliminarCurso("A1");
        System.out.println("");
        utn.listarCursos();

        System.out.println("El curso A1 no pertenece al profesor Luciano");
        luciano.listarCursos();

        System.out.println("////////////////////////////////");
        System.out.println("---------------------------------");
        System.out.println("////////////////////////////////");
```

```
        //Tarea 7: Remover un profesor y dejar profesor = null.
        System.out.println("\n--- Eliminando profesor Javier ---");
        utn.eliminarProfesor("NMCBX");
        System.out.println("");
        utn.listarCursos();
        utn.listarProfesor();

        System.out.println("////////////////////////////////////");
        System.out.println("------------------------------------");
        System.out.println("////////////////////////////////////");

        //Tarea 8: Mostrar un reporte: cantidad de cursos por profesor.
        utn.mostrarReporteCursosPorProfesor();
    }
}
```

Repositorio: https://github.com/Tobias-L7/Trabajos-Practicos-P2.git