Trabajo Practico 6

Ejercicio 1)

Sistema de Stock

```
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;
    }
}
```

```
public class Producto {
    private String id;
    private String nombre;
    private double precio;
    private int cantidad;
    private CategoriaProducto categoria;

    public Producto(String id, String nombre, double precio, int cantidad, CategoriaProducto
categoria) {
```

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

    public String getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

    public double getPrecio() {
        return precio;
    }

    public int getCantidad() {
        return cantidad;
    }

    public CategoriaProducto getCategoria() {
        return categoria;
    }

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

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

```
    }
}
```

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

public class Inventario {
    private ArrayList<Producto> productos = new ArrayList<>();

    public void agregarProducto(Producto p) {
        if (p != null) {
            productos.add(p);
        }
    }

    public void listarProductos() {
        for (Producto p : productos) {
            p.mostrarInfo();
        }
    }

    public Producto buscarProductoPorId(String id) {
        for (Producto p : productos) {
            if (p.getId().equals(id)) {
                return p;
            }
        }
        return null;
    }
```

```
public boolean eliminarProducto(String id) {
    Producto p = buscarProductoPorId(id);
    if (p != null) {
        productos.remove(p);
        return true;
    }
    return false;
}

public boolean actualizarStock(String id, int nuevaCantidad) {
    Producto p = buscarProductoPorId(id);
    if (p != null) {
        p.setCantidad(nuevaCantidad);
        return true;
    }
    return false;
}

public List<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
    ArrayList<Producto> res = new ArrayList<>();
    for (Producto p : productos) {
        if (p.getCategoria() == categoria) {
            res.add(p);
        }
    }
    return res;
}

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

```
    }

    public Producto obtenerProductoConMayorStock() {
        if (productos.isEmpty()) return null;
        return Collections.max(productos, Comparator.comparingInt(Producto::getCantidad));
    }

    public List<Producto> filtrarProductosPorPrecio(double min, double max) {
        ArrayList<Producto> res = new ArrayList<>();
        for (Producto p : productos) {
            if (p.getPrecio() >= min && p.getPrecio() <= max) {
                res.add(p);
            }
        }
        return res;
    }

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

```
public class MainStock {
    public static void main(String[] args) {
        Inventario inventario = new Inventario();

        Producto p1 = new Producto("P001", "Arroz", 800.0, 50,
CategoriaProducto.ALIMENTOS);
        Producto p2 = new Producto("P002", "Televisor 42", 25000.0, 10,
CategoriaProducto.ELECTRONICA);
        Producto p3 = new Producto("P003", "Remera", 1500.0, 30, CategoriaProducto.ROPA);
```

```
        Producto p4 = new Producto("P004", "Licuadora", 4500.0, 5,
CategoriaProducto.HOGAR);
        Producto p5 = new Producto("P005", "Fideos", 500.0, 100,
CategoriaProducto.ALIMENTOS);

        inventario.agregarProducto(p1);
        inventario.agregarProducto(p2);
        inventario.agregarProducto(p3);
        inventario.agregarProducto(p4);
        inventario.agregarProducto(p5);

        System.out.println("Listado de productos:");
        inventario.listarProductos();

        System.out.println("\nBuscar producto por ID P003:");
        Producto busc = inventario.buscarProductoPorId("P003");
        if (busc != null) busc.mostrarInfo();

        System.out.println("\nFiltrar por categoria ALIMENTOS:");
        for (Producto p : inventario.filtrarPorCategoria(CategoriaProducto.ALIMENTOS)) {
            p.mostrarInfo();
        }

        System.out.println("\nEliminar producto P004");
        inventario.eliminarProducto("P004");
        System.out.println("Productos restantes:");
        inventario.listarProductos();

        System.out.println("\nActualizar stock P002 a 15");
        inventario.actualizarStock("P002", 15);
        inventario.buscarProductoPorId("P002").mostrarInfo();

        System.out.println("\nTotal de stock disponible: " + inventario.obtenerTotalStock());
```

```
        System.out.println("\nProducto con mayor stock:");
        Producto mayor = inventario.obtenerProductoConMayorStock();
        if (mayor != null) mayor.mostrarInfo();

        System.out.println("\nFiltrar productos con precio entre 1000 y 30000:");
        for (Producto p : inventario.filtrarProductosPorPrecio(1000, 30000)) {
            p.mostrarInfo();
        }

        System.out.println("\nCategorias disponibles:");
        inventario.mostrarCategoriasDisponibles();
    }
}
```

Ejercicio 2)

Biblioteca y Libros (Composicion 1:N)

```
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;
    }

    public String getNacionalidad() {
        return nacionalidad;
    }

    public void mostrarInfo() {
        System.out.println("Autor ID: " + id + " | Nombre: " + nombre + " | Nacionalidad: " +
nacionalidad);
    }
}
```

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

    public String getTitulo() {
```

```
        return titulo;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    public Autor getAutor() {
        return autor;
    }

    public void mostrarInfo() {
        System.out.println("ISBN: " + isbn + " | Titulo: " + titulo + " | Anio: " + anioPublicacion + "
| Autor: " + autor.getNombre());
    }
}
```

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

public class Biblioteca {
    private String nombre;
    private List<Libro> libros = new ArrayList<>();

    public Biblioteca(String nombre) {
        this.nombre = nombre;
    }

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

    public void listarLibros() {
```

```
    for (Libro l : libros) {
        l.mostrarInfo();
    }
}

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

public boolean eliminarLibro(String isbn) {
    Libro l = buscarLibroPorIsbn(isbn);
    if (l != null) {
        libros.remove(l);
        return true;
    }
    return false;
}

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

public List<Libro> filtrarLibrosPorAnio(int anio) {
    List<Libro> res = new ArrayList<>();
    for (Libro l : libros) {
        if (l.getAnioPublicacion() == anio) {
            res.add(l);
        }
```

```
        }
        return res;
    }

    public void mostrarAutoresDisponibles() {
        List<String> ids = new ArrayList<>();
        for (Libro l : libros) {
            Autor a = l.getAutor();
            if (!ids.contains(a.getId())) {
                ids.add(a.getId());
                a.mostrarInfo();
            }
        }
    }
}
```

```
public class MainBiblioteca {
    public static void main(String[] args) {
        Biblioteca biblioteca = new Biblioteca("Biblioteca Central");

        Autor a1 = new Autor("A001", "Gabriel Garcia Marquez", "Colombia");
        Autor a2 = new Autor("A002", "Jorge Luis Borges", "Argentina");
        Autor a3 = new Autor("A003", "Isabel Allende", "Chile");

        biblioteca.agregarLibro("ISBN001", "Cien Anios de Soledad", 1967, a1);
        biblioteca.agregarLibro("ISBN002", "El Aleph", 1949, a2);
        biblioteca.agregarLibro("ISBN003", "La Casa de los Espiritus", 1982, a3);
        biblioteca.agregarLibro("ISBN004", "Cronica de una Muerte Anunciada", 1981, a1);
        biblioteca.agregarLibro("ISBN005", "Ficciones", 1944, a2);

        System.out.println("Listado de libros:");
        biblioteca.listarLibros();
```

```
        System.out.println("\nBuscar libro por ISBN ISBN003:");
        Libro busc = biblioteca.buscarLibroPorIsbn("ISBN003");
        if (busc != null) busc.mostrarInfo();

        System.out.println("\nFiltrar libros por anio 1949:");
        for (Libro l : biblioteca.filtrarLibrosPorAnio(1949)) {
            l.mostrarInfo();
        }

        System.out.println("\nEliminar libro ISBN004");
        biblioteca.eliminarLibro("ISBN004");
        System.out.println("Libros restantes:");
        biblioteca.listarLibros();

        System.out.println("\nCantidad total de libros: " + biblioteca.obtenerCantidadLibros());

        System.out.println("\nAutores disponibles en la biblioteca:");
        biblioteca.mostrarAutoresDisponibles();
    }
}
```

Ejercicio 3)

Universidad, Profesor y Curso (Bidireccional 1:N)

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

public class Profesor {
    private String id;
    private String nombre;
    private String especialidad;
    private List<Curso> cursos = new ArrayList<>();
```

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

public String getId() {
    return id;
}

public String getNombre() {
    return nombre;
}

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

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

public void eliminarCurso(Curso c) {
    if (c == null) return;
    if (cursos.contains(c)) {
        cursos.remove(c);
```

```
            if (c.getProfesor() == this) {
                c.setProfesor(null);
            }
        }
    }

    public void mostrarInfo() {
        System.out.println("Profesor ID: " + id + " | Nombre: " + nombre + " | Especialidad: " +
especialidad + " | Cantidad cursos: " + cursos.size());
    }
}
```

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

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

    public String getCodigo() {
        return codigo;
    }

    public String getNombre() {
        return nombre;
    }

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

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

        Profesor anterior = this.profesor;
        this.profesor = p;

        if (anterior != null) {
            anterior.eliminarCurso(this);
        }

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

    public void mostrarInfo() {
        String nomProf = (profesor != null) ? profesor.getNombre() : "Sin profesor";
        System.out.println("Curso: " + codigo + " | " + nombre + " | Profesor: " + nomProf);
    }
}
```

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

public class Universidad {
    private String nombre;
    private List<Profesor> profesores = new ArrayList<>();
    private List<Curso> cursos = new ArrayList<>();

    public Universidad(String nombre) {
        this.nombre = nombre;
```

```
}

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

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

public Profesor buscarProfesorPorId(String id) {
    for (Profesor p : profesores) {
        if (p.getId().equals(id)) return p;
    }
    return null;
}

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

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

```
    }
}

public void listarProfesores() {
    for (Profesor p : profesores) {
        p.mostrarInfo();
    }
}

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

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

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);
    }
```

```
    }

    public void reporteCantidadCursosPorProfesor() {
        for (Profesor p : profesores) {
            System.out.println(p.getNombre() + " -> " + p.getCursos().size() + " cursos");
        }
    }
}
```

```
public class MainUniversidad {
    public static void main(String[] args) {
        Universidad uni = new Universidad("Universidad Ejemplo");

        Profesor prof1 = new Profesor("PR001", "Ana Martinez", "Matematica");
        Profesor prof2 = new Profesor("PR002", "Carlos Lopez", "Programacion");
        Profesor prof3 = new Profesor("PR003", "Laura Diaz", "Fisica");

        Curso c1 = new Curso("C001", "Algebra");
        Curso c2 = new Curso("C002", "Programacion I");
        Curso c3 = new Curso("C003", "Fisica General");
        Curso c4 = new Curso("C004", "Estructuras de Datos");
        Curso c5 = new Curso("C005", "Calculo");

        uni.agregarProfesor(prof1);
        uni.agregarProfesor(prof2);
        uni.agregarProfesor(prof3);

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

```
        uni.asignarProfesorACurso("C001", "PR001");
        uni.asignarProfesorACurso("C002", "PR002");
        uni.asignarProfesorACurso("C003", "PR003");
        uni.asignarProfesorACurso("C004", "PR002");
        uni.asignarProfesorACurso("C005", "PR001");

        System.out.println("Cursos con su profesor:");
        uni.listarCursos();

        System.out.println("\nProfesores con sus cursos:");
        uni.listarProfesores();

        System.out.println("\nCambiar profesor de C004 a PR003");
        uni.asignarProfesorACurso("C004", "PR003");
        uni.listarCursos();
        System.out.println("\nVerificar sincronizacion:");
        uni.listarProfesores();

        System.out.println("\nEliminar curso C002");
        uni.eliminarCurso("C002");
        uni.listarCursos();
        System.out.println("\nVerificar que C002 ya no aparece en cursos de profesor:");
        uni.listarProfesores();

        System.out.println("\nEliminar profesor PR001");
        uni.eliminarProfesor("PR001");
        uni.listarProfesores();
        System.out.println("\nCursos despues de eliminar profesor PR001:");
        uni.listarCursos();

        System.out.println("\nReporte cantidad de cursos por profesor:");
        uni.reporteCantidadCursosPorProfesor();
    }
```

```
}
```