PROGRAMACIÓN II

Trabajo Práctico 6: Colecciones

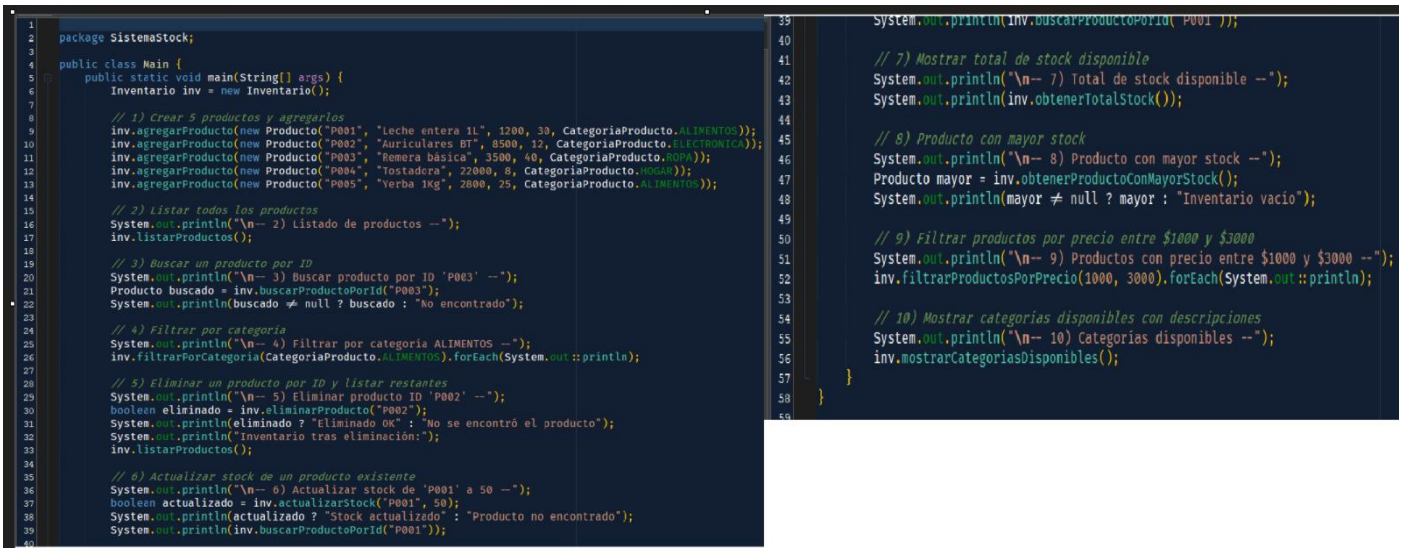
Alumno: Mauro Zavatti

Propuesto 1: Inventario Stock

```java
package SistemaStock;

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 setPrecio(double precio) { this.precio = precio; }

    public void mostrarInfo() {
        System.out.println(this);
    }

    @Override
    public String toString() {
        return "Producto{" +
                "id='" + id + '\'' +
                ", nombre='" + nombre + '\'' +
                ", precio=$" + precio +
                ", cantidad=" + cantidad +
                ", categoria=" + categoria + " (" + categoria.getDescripcion() + ")" +
                '}';
    }
}
```

```java
package SistemaStock;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;

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

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

    public void listarProductos() {
        if (productos.isEmpty()) {
            System.out.println("(Inventario vacío)");
            return;
        }
        productos.forEach(Producto::mostrarInfo);
    }

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

    public boolean eliminarProducto(String id) {
        return productos.removeIf(p -> p.getId().equalsIgnoreCase(id));
    }

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

    public List<Producto> filtrarPorCategoria(CategoriaProducto categoria) {
        return productos.stream()
                .filter(p -> p.getCategoria() == categoria)
                .collect(Collectors.toList());
    }

    public int obtenerTotalStock() {
        return productos.stream().mapToInt(Producto::getCantidad).sum();
    }

    public Producto obtenerProductoConMayorStock() {
        Optional<Producto> max = productos.stream()
                .max(Comparator.comparingInt(Producto::getCantidad));
        return max.orElse(null);
    }

    public List<Producto> filtrarProductosPorPrecio(double min, double max) {
        return productos.stream()
                .filter(p -> p.getPrecio() >= min && p.getPrecio() <= max)
                .collect(Collectors.toList());
    }

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

```java
package SistemaStock;

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
package SistemaStock;

public class Main {
    public stetic void main(String[] args) {
        Inventario inv = new Inventario();

        // 1) Crear 5 productos y agregarlos
        inv.agregarProducto(new Producto("P001", "Leche entera 1L", 1200, 30, CategoriaProducto.ALIMENTOS));
        inv.agregarProducto(new Producto("P002", "Auriculares BT", 8500, 12, CategoriaProducto.ELECTRONICA));
        inv.agregarProducto(new Producto("P003", "Remera básica", 3500, 40, CategoriaProducto.ROPA));
        inv.agregarProducto(new Producto("P004", "Tostadora", 22000, 8, CategoriaProducto.HOGAR));
        inv.agregarProducto(new Producto("P005", "Yerba 1Kg", 2800, 25, CategoriaProducto.ALIMENTOS));

        // 2) Listar todos los productos
        System.out.println("\n-- 2) Listado de productos --");
        inv.listarProductos();

        // 3) Buscar un producto por ID
        System.out.println("\n-- 3) Buscar producto por ID 'P003' --");
        Producto buscado = inv.buscarProductoPorId("P003");
        System.out.println(buscado ≠ null ? buscado : "No encontrado");

        // 4) Filtrar por categoría
        System.out.println("\n-- 4) Filtrar por categoría ALIMENTOS --");
        inv.filtrarPorCategoria(CategoriaProducto.ALIMENTOS).forEach(System.out::println);

        // 5) Eliminar un producto por ID y listar restantes
        System.out.println("\n-- 5) Eliminar producto ID 'P002' --");
        boolean eliminado = inv.eliminarProducto("P002");
        System.out.println(eliminado ? "Eliminado OK" : "No se encontró el producto");
        System.out.println("Inventario tras eliminación:");
        inv.listarProductos();

        // 6) Actualizar stock de un producto existente
        System.out.println("\n-- 6) Actualizar stock de 'P001' a 50 --");
        boolean actualizado = inv.actualizarStock("P001", 50);
        System.out.println(actualizado ? "Stock actualizado" : "Producto no encontrado");
        System.out.println(inv.buscarProductoPorId("P001"));
        System.out.println(inv.buscarProductoPorId("P001"));

        // 7) Mostrar total de stock disponible
        System.out.println("\n-- 7) Total de stock disponible --");
        System.out.println(inv.obtenerTotalStock());

        // 8) Producto con mayor stock
        System.out.println("\n-- 8) Producto con mayor stock --");
        Producto mayor = inv.obtenerProductoConMayorStock();
        System.out.println(mayor ≠ null ? mayor : "Inventario vacío");

        // 9) Filtrar productos por precio entre $1000 y $3000
        System.out.println("\n-- 9) Productos con precio entre $1000 y $3000 --");
        inv.filtrarProductosPorPrecio(1000, 3000).forEach(System.out::println);

        // 10) Mostrar categorías disponibles con descripciones
        System.out.println("\n-- 10) Categorías disponibles --");
        inv.mostrarCategoriasDisponibles();
    }
}
```

Propuesto 2: Biblioteca y Libros

```java
package Biblioteca;

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(this);
    }

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

```java
package Biblioteca;

import java.util.Objects;

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(this);
    }

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


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Autor)) return false;
        Autor autor = (Autor) o;
        return Objects.equals(id, autor.id);
    }
    @Override
    public int hashCode() {
        return Objects.hash(id);
    }
}
```

```java
package Biblioteca;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

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

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

    // COMPOSICIÓN: la biblioteca crea el Libro y lo guarda internamente
    public void agregarLibro(String isbn, String titulo, int anioPublicacion, Autor autor) {
        libros.add(new Libro(isbn, titulo, anioPublicacion, autor));
    }

    public void listarLibros() {
        if (libros.isEmpty()) {
            System.out.println("(No hay libros en la biblioteca)");
            return;
        }
        libros.forEach(Libro::mostrarInfo);
    }

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

    public boolean eliminarLibro(String isbn) {
        return libros.removeIf(l -> l.getIsbn().equalsIgnoreCase(isbn));
    }

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

    public List<Libro> filtrarLibrosPorAnio(int anio) {
        return libros.stream()
                .filter(l -> l.getAnioPublicacion() == anio)
                .collect(Collectors.toList());
    }

    public void mostrarAutoresDisponibles() {
        if (libros.isEmpty()) {
            System.out.println("(No hay autores: la biblioteca no tiene libros)");
            return;
        }
        // Set para evitar autores repetidos manteniendo orden de aparición
        Set<Autor> autores = new LinkedHashSet<>();
        for (Libro l : libros) autores.add(l.getAutor());
        autores.forEach(Autor::mostrarInfo);
    }

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

```java
package Biblioteca;

import java.util.List;

public class Main {
    public static void main(String[] args) {
        // 1) Creamos una biblioteca
        Biblioteca biblio = new Biblioteca("Biblioteca Central");

        // 2) Crear al menos tres autores
        Autor a1 = new Autor("A001", "Julio Cortázar", "Argentina");
        Autor a2 = new Autor("A002", "Ursula K. Le Guin", "Estados Unidos");
        Autor a3 = new Autor("A003", "Gabriel García Márquez", "Colombia");

        // 3) Agregar 5 libros asociados a alguno de los Autores a la biblioteca
        biblio.agregarLibro("ISBN-001", "Rayuela", 1963, a1);
        biblio.agregarLibro("ISBN-002", "Bestiario", 1951, a1);
        biblio.agregarLibro("ISBN-003", "Los desposeídos", 1974, a2);
        biblio.agregarLibro("ISBN-004", "La mano izquierda de la oscuridad", 1969, a2);
        biblio.agregarLibro("ISBN-005", "Cien años de soledad", 1967, a3);

        // 4) Listar todos los libros con su información y la del autor
        System.out.println("\n-- 4) Listado de libros --");
        biblio.listarLibros();

        // 5) Buscar un libro por su ISBN y mostrar su información
        System.out.println("\n-- 5) Buscar por ISBN 'ISBN-003' --");
        Libro buscado = biblio.buscarLibroPorIsbn("ISBN-003");
        System.out.println(buscado ≠ null ? buscado : "No encontrado");

        // 6) Filtrar y mostrar los libros publicados en un año específico
        System.out.println("\n-- 6) Libros de 1969 --");
        List<Libro> de1969 = biblio.filtrarLibrosPorAnio(1969);
        if (de1969.isEmpty()) System.out.println("(Sin resultados)");
        else de1969.forEach(System.out::println);

        // 7) Eliminar un libro por su ISBN y listar los libros restantes
        System.out.println("\n-- 7) Eliminar ISBN 'ISBN-002' --");
        boolean eliminado = biblio.eliminarLibro("ISBN-002");
        System.out.println(eliminado ? "Eliminado OK" : "No se encontró el libro");
        System.out.println("Libros restantes:");
        biblio.listarLibros();

        // 8) Mostrar la cantidad total de libros en la biblioteca
        System.out.println("\n-- 8) Cantidad total de libros --");
        System.out.println(biblio.obtenerCantidadLibros());

        // 9) Listar todos los autores de los libros disponibles en la biblioteca
        System.out.println("\n-- 9) Autores disponibles --");
        biblio.mostrarAutoresDisponibles();
    }
}
```

Propuesto 2: Universidad

```java
package Universidad;

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

public class Profesor {
    private final String id;
    private String nombre;
    private String especialidad;
    private final 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 String getEspecialidad() { return especialidad; }
    public List<Curso> getCursos() { return Collections.unmodifiableList(cursos); }

    // ── API pública (sincroniza ambos lados) ──
    public void agregarCurso(Curso c) {
        if (c == null) return;
        if (!cursos.contains(c)) {
            cursos.add(c);
        }
        if (c.getProfesor() ≠ this) {
            c.setProfesor(this); // sincroniza lado Curso (quita de profesor anterior si aplica)
        }
    }

    public void eliminarCurso(Curso c) {
        if (c == null) return;
        if (cursos.remove(c)) {
            if (c.getProfesor() == this) {
                c.setProfesor(null); // sincroniza lado Curso
            }
        }
    }

    public void listarCursos() {
        if (cursos.isEmpty()) {
            System.out.println("  (sin cursos asignados)");
            return;
        }
        for (Curso c : cursos) {
            System.out.println("  - " + c.getCodigo() + " | " + c.getNombre());
        }
    }

    public void mostrarInfo() {
        System.out.println("Profesor{id='" + id + "', nombre='" + nombre + "', especialidad='" + especialidad
                + "', cursos=" + cursos.size() + "}");
    }

    // ── Helpers internos para evitar recursión ──
    void addCursoInternal(Curso c) {
        if (c ≠ null && !cursos.contains(c)) cursos.add(c);
    }
    void removeCursoInternal(Curso c) {
        cursos.remove(c);
    }
}
```

```java
package Universidad;

public class Curso {
    private final String codigo;
    private String nombre;
    private Profesor profesor; // 1 profesor responsable

    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; }

    // Asigna/cambia profesor sincronizando ambos lados
    public void setProfesor(Profesor p) {
        if (this.profesor == p) return; // nada que hacer

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

        if (anterior != null) {
            anterior.removeCursoInternal(this); // quitar de la lista del profesor anterior (sin recursión)
        }
        if (p != null) {
            p.addCursoInternal(this); // agregar a la lista del nuevo profesor (sin recursión)
        }
    }

    public void mostrarInfo() {
        String prof = (profesor == null) ? "(sin profesor)" : profesor.getNombre();
        System.out.println("Curso{codigo='" + codigo + "', nombre='" + nombre + "', profesor=" + prof + "}");
    }
}
```

```java
package Universidad;

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

public class Universidad {
        private final String nombre;
    private final List<Profesor> profesores = new ArrayList<>();
    private final 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().equalsIgnoreCase(id)) return p;
        return null;
    }
    public Curso buscarCursoPorCodigo(String codigo) {
        for (Curso c : cursos) if (c.getCodigo().equalsIgnoreCase(codigo)) return c;
        return null;
    }

    public void asignarProfesorACurso(String codigoCurso, String idProfesor) {
        Curso c = buscarCursoPorCodigo(codigoCurso);
        Profesor p = buscarProfesorPorId(idProfesor);
        if (c == null || p == null) {
            System.out.println("No se pudo asignar. Curso o Profesor inexistente.");
            return;
        }
        c.setProfesor(p); // usa la lógica de sincronización de Curso
    }

    public void eliminarCurso(String codigo) {
        Curso c = buscarCursoPorCodigo(codigo);
        if (c == null) return;
        // Romper relación con profesor si existe
        if (c.getProfesor() != null) {
            c.setProfesor(null);
        }
        cursos.remove(c);
    }
```

```java
    }

    public void eliminarProfesor(String id) {
        Profesor p = buscarProfesorPorId(id);
        if (p == null) return;
        // Dejar profesor = null en todos sus cursos
        // Copiamos la lista para evitar ConcurrentModification
        List<Curso> copia = new ArrayList<>(p.getCursos());
        for (Curso c : copia) {
            c.setProfesor(null);
        }
        profesores.remove(p);
    }

    public void listarProfesores() {
        System.out.println("=== Profesores ===");
        if (profesores.isEmpty()) { System.out.println("(sin profesores)"); return; }
        for (Profesor p : profesores) {
            p.mostrarInfo();
            p.listarCursos();
        }
    }

    public void listarCursos() {
        System.out.println("=== Cursos ===");
        if (cursos.isEmpty()) { System.out.println("(sin cursos)"); return; }
        for (Curso c : cursos) c.mostrarInfo();
    }

    public void reporteCantidadCursosPorProfesor() {
        System.out.println("=== Reporte: cantidad de cursos por profesor ===");
        if (profesores.isEmpty()) { System.out.println("(sin profesores)"); return; }
        for (Profesor p : profesores) {
            System.out.println(p.getNombre() + " → " + p.getCursos().size() + " curso(s)");
        }
    }
}
```

```java

package Universidad;

public class Main {
    public static void main(String[] args) {
        // 1) Crear al menos 3 profesores y 5 cursos.
        Profesor p1 = new Profesor("P001", "Ana Torres", "Matemática");
        Profesor p2 = new Profesor("P002", "Luis Pérez", "Programación");
        Profesor p3 = new Profesor("P003", "Carla Gómez", "Física");

        Curso c1 = new Curso("MATE1", "Álgebra I");
        Curso c2 = new Curso("PROG1", "Programación I");
        Curso c3 = new Curso("FIS1",  "Física I");
        Curso c4 = new Curso("PROG2", "Programación II");
        Curso c5 = new Curso("MATE2", "Cálculo I");

        // 2) Agregar profesores y cursos a la universidad.
        Universidad uni = new Universidad("UNLaR");
        uni.agregarProfesor(p1);
        uni.agregarProfesor(p2);
        uni.agregarProfesor(p3);

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

        // 3) Asignar profesores a cursos usando asignarProfesorACurso(...).
        uni.asignarProfesorACurso("MATE1", "P001"); // Álgebra I → Ana
        uni.asignarProfesorACurso("MATE2", "P001"); // Cálculo I → Ana
        uni.asignarProfesorACurso("PROG1", "P002"); // Programación I → Luis
        uni.asignarProfesorACurso("PROG2", "P002"); // Programación II → Luis
        uni.asignarProfesorACurso("FIS1",  "P003"); // Física I → Carla

        // 4) Listar cursos con su profesor y profesores con sus cursos.
        System.out.println();
        uni.listarCursos();
        System.out.println();
        uni.listarProfesores();
```

```
40              uni.listarProfesores();
41
42              // 5) Cambiar el profesor de un curso y verificar sincronización.
43              System.out.println("\n-- Cambio de profesor: PROG2 pasa de Luis (P002) a Carla (P003) --");
44              uni.asignarProfesorACurso("PROG2", "P003");
45              uni.listarCursos();
46              System.out.println();
47              uni.listarProfesores();
48
49              // 6) Remover un curso y confirmar que ya no aparece en la lista del profesor.
50              System.out.println("\n-- Eliminar curso MATE1 --");
51              uni.eliminarCurso("MATE1");
52              uni.listarCursos();
53              System.out.println();
54              uni.listarProfesores();
55
56              // 7) Remover un profesor y dejar profesor = null en sus cursos.
57              System.out.println("\n-- Eliminar profesor P002 (Luis) --");
58              uni.eliminarProfesor("P002");
59              uni.listarCursos(); // cursos PROG1 (de Luis) debe quedar con profesor = null
60              System.out.println();
61              uni.listarProfesores();
62
63              // 8) Reporte: cantidad de cursos por profesor.
64              System.out.println();
65              uni.reporteCantidadCursosPorProfesor();
66          }
67      }
68
```