

Trabajo Practico 7 — Programación 2

Alumnos: Alliot Fermin

Carrera: Tecnicatura Universitaria en Programación

Materia: Programación 2

GitHub: <https://github.com/Falliot00/UTN-TUPaD-P2/tree/main/6.%20Colecciones>

Fecha: 20/11/2025

Introducción

En este trabajo práctico implementé el uso de colecciones en Java, principalmente `ArrayList`, para modelar y gestionar relaciones entre objetos en tres escenarios: inventario de tienda, gestión de biblioteca y sistema universitario. El objetivo fue aplicar relaciones 1:N, N:1 y N:M bidireccionales, así como enums, encapsulamiento y métodos de búsqueda y filtrado.

Ejercicio 1 – Sistema de Inventario

Desarrollé un sistema de inventario compuesto por `Producto`, un enum `CategoriaProducto`, y la clase `Inventario`, que administra una colección de productos. Implementé funcionalidades como agregar, eliminar, filtrar por categoría, actualizar stock, buscar por ID, obtener el producto con mayor stock y calcular el total disponible.

El diseño aplica encapsulamiento, uso correcto de ArrayList y filtros mediante recorridos lineales.

Ejercicio 2 – Sistema de Biblioteca

Implementé un sistema de biblioteca con `Autor`, `Libro` y `Biblioteca`. Se utilizó una relación N:1 entre `Libro` y `Autor`, y la clase `Biblioteca` gestiona un conjunto de libros. Las funcionalidades cubren el alta y baja de libros, búsqueda por ISBN, filtrado por año de publicación y listado de autores sin duplicados.

El sistema garantiza integridad básica y presenta correctamente la información combinada libro-autor.

Ejercicio 3 – Sistema Universitario

Desarrollé un sistema universitario con relación N:M bidireccional entre `Profesor` y `Curso`. Este fue el ejercicio más complejo, ya que requería mantener sincronización entre ambos lados de la relación al asignar, cambiar o eliminar profesores y cursos.

Implementé métodos internos y flags de sincronización para evitar recursión infinita y garantizar que cada cambio se refleje correctamente en las dos entidades.

Conceptos Aplicados

- Uso de `ArrayList<T>` para administrar colecciones dinámicas.

- Aplicación de relaciones entre clases (1:N, N:1 y N:M bidireccional).
- Implementación de enumeraciones con atributos descriptivos.
- Encapsulamiento, separación de responsabilidades y código limpio.
- Búsquedas lineales, filtrados y operaciones de reporte.
- Prevención de errores comunes: null, duplicados, modificaciones concurrentes.

Conclusión

El trabajo práctico permitió integrar de forma sólida las colecciones en Java con modelado orientado a objetos. Los tres ejercicios simulan escenarios reales de administración de datos y afianzan el manejo de relaciones, filtrados y sincronización bidireccional.

El resultado final es un conjunto de sistemas funcionales, claros y correctamente encapsulados, que cumplen en totalidad los requisitos del TP y reflejan buenas prácticas de diseño.