PROGRAMACION II

TRABAJO PRÁCTICO 6: COLECCIONES

RESUMEN

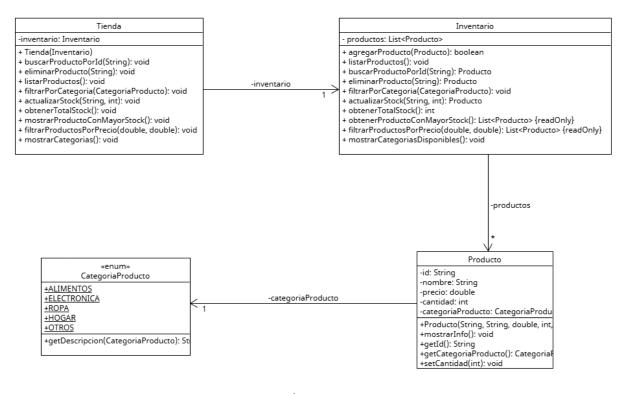
Estructuras de datos dinámicas en Java mediante el uso de colecciones (ArrayList) y enumeraciones (enum), implementando un sistema de stock con funcionalidades progresivas que refuerzan conceptos clave de la programación orientada a objetos

Nicolas Olima
nicolima200@gmail.com
REPO GITHUB

Caso Práctico 1

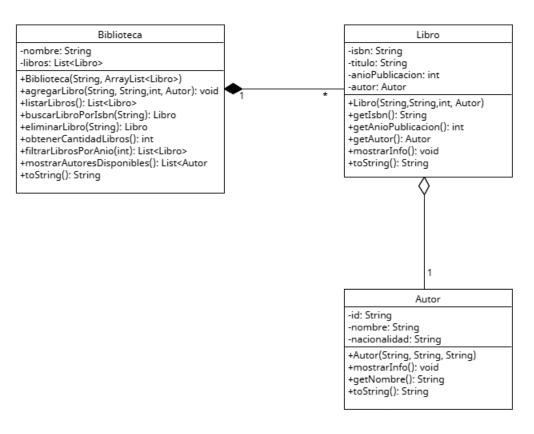
Se debe desarrollar un sistema de stock que permita gestionar productos en una tienda, controlando su disponibilidad, precios y categorías. La información se modelará utilizando clases, colecciones dinámicas y enumeraciones en Java.

TIENDA-PRODUCTO-INVENTARIO



Caso práctico 2:

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.



Caso práctico 3:

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).

UNIVERSIDAD-PROFESOR-CURSO

Universidad Curso -nombre: String -codigo: String -profesores: List<Profesor> -nombre: String -creacion -profesor: Profesor -cursos: List<Curso> +Universidad(String, List<Profesor> +Curso(String, String) +agregarProfesor(Profesor): void +setProfesor(Profesor): void +getProfesor(): Profesor +agregarCurso(Curso): void +asignarProfesorACurso(String, Stri +getCodigo(): String +listarProfesores(): void +mostrarInfo(): void +listarCursos(): void +toString(): String +buscarProfesorPorId(String): Profe +buscarCursoPorCodigo(String): Cur +eliminarCurso(String): void +eliminarProfesor(String): void +cantCursosPorProfesorId(String): i +toString(): String Profesor -id: String -nombre: String -especialidad: String -cursos: List<Curso> +Profesor(String, String, String) -creacion +agregarCurso(Curso): void +eliminarCurso(Curso): void +listarCursos(): void +getId(): String

> Repo

+getCursos(): List<Curso> +getNombre(): String +cantCursos(): int +mostrarInfo(): void +toString(): String