

Trabajo Práctico 6: Colecciones

Nombre: Alfredo de Inocenti

Materia: Programación II

Carrera: Tecnicatura Universitaria en Programación

Repositorio:

OBJETIVO GENERAL

Desarrollar 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.

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.

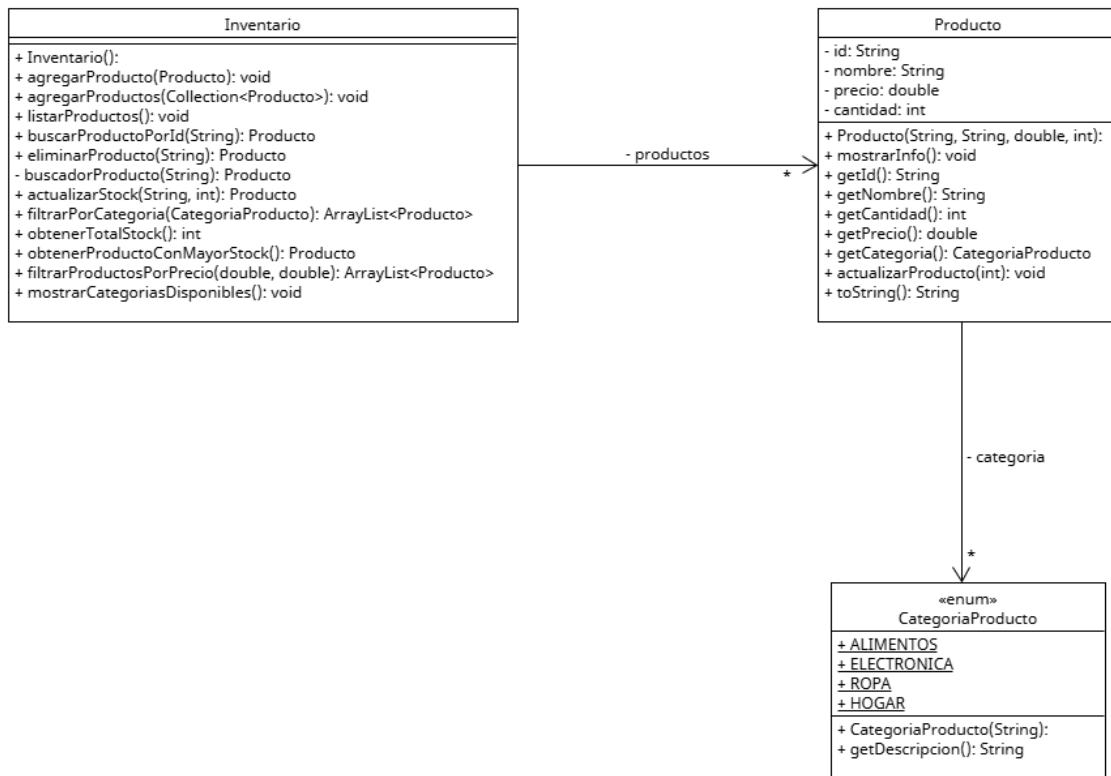
2. Clases a implementar

- Clase Producto
- Enum CategoriaProducto
- Clase Inventario

3. Tareas a realizar

- Crear al menos cinco productos con diferentes categorías y agregarlos al inventario.
- Listar todos los productos mostrando su información y categoría.
- Buscar un producto por ID y mostrar su información.
- Filtrar y mostrar productos que pertenezcan a una categoría específica.
- Eliminar un producto por su ID y listar los productos restantes.
- Actualizar el stock de un producto existente.
- Mostrar el total de stock disponible.
- Obtener y mostrar el producto con mayor stock.
- Filtrar productos con precios entre \$1000 y \$3000.
- Mostrar las categorías disponibles con sus descripciones.

Diagrama UML:



Caso Práctico 2

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. Clases a implementar

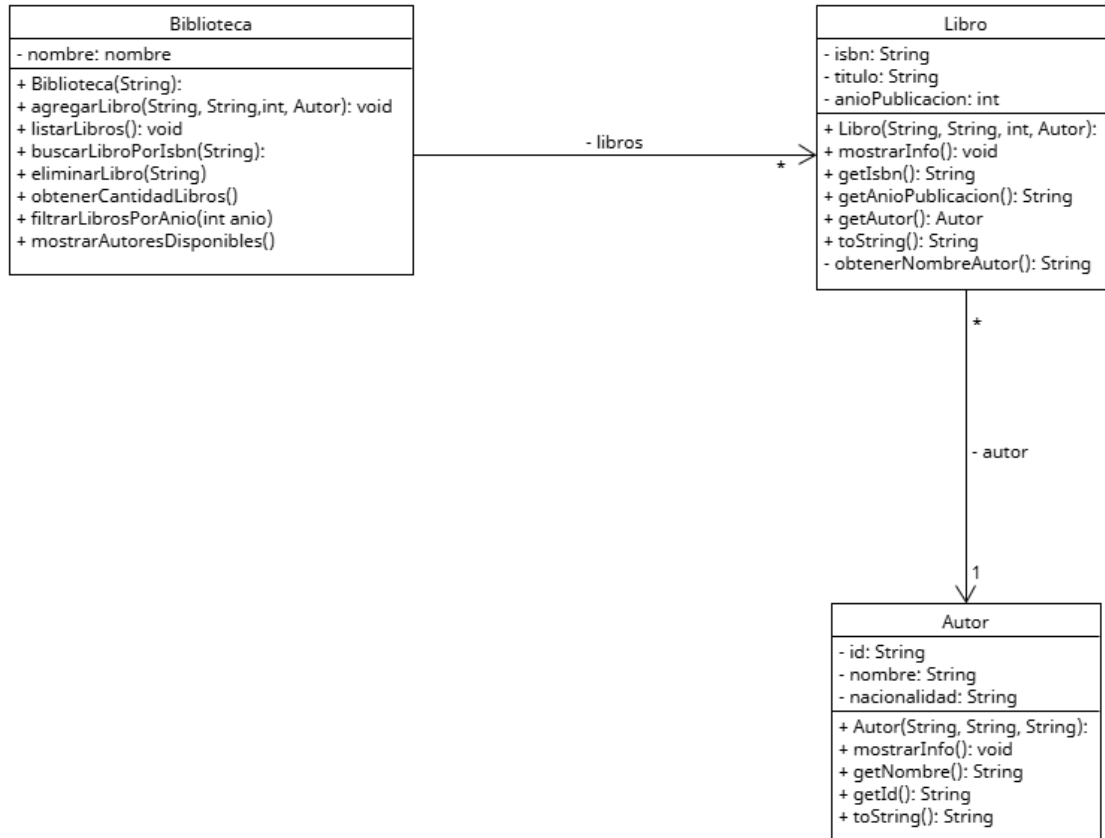
- Clase Autor
- Clase Libro
- Biblioteca

3. Tareas a realizar

- Creamos una biblioteca.
- Crear al menos tres autores
- Agregar 5 libros asociados a alguno de los Autores a la biblioteca.
- Listar todos los libros con su información y la del autor.
- Buscar un libro por su ISBN y mostrar su información.
- Filtrar y mostrar los libros publicados en un año específico.

- Eliminar un libro por su ISBN y listar los libros restantes.
- Mostrar la cantidad total de libros en la biblioteca.
- Listar todos los autores de los libros disponibles en la biblioteca.

Diagrama UML:



Caso Práctico 3

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 \leftrightarrow 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).

2. Clases a implementar

- Clase Profesor
- Clase Curso
- Clase Universidad

3. Tareas a realizar

- Crear al menos 3 profesores y 5 cursos.
- Agregar profesores y cursos a la universidad.
- Asignar profesores a cursos usando `asignarProfesorACurso(...)`.
- Listar cursos con su profesor y profesores con sus cursos.
- Cambiar el profesor de un curso y verificar que ambos lados quedan sincronizados.
- Remover un curso y confirmar que ya no aparece en la lista del profesor.
- Remover un profesor y dejar profesor = null.
- Mostrar un reporte: cantidad de cursos por profesor.

Diagrama UML:

