Práctica 8. Relaciones entre clases

Objetivo

- Aplicar los conceptos de POO en la manipulación de arreglos de objetos en Java.
- Implementa operaciones básicas como creación, búsqueda, modificación y eliminación de objetos dentro de un arreglo.
- Desarrollar habilidades en el uso de ciclos y estructuras de control para gestionar colecciones de datos (solo arreglos

Competencias para desarrollar:

- Diseño e implementación de clases en Java
- Manejo de arreglos de objetos
- Aplicación de principios de encapsulamiento y reutilización de código.
- Implementación de métodos para manipulación de datos.

Requisitos previos:

- Conocimientos básicos de Java y POO (clases, objetos, métodos y atributos)
- Uso de arreglos en Java
- Conocimientos de estructuras de control como ciclos y condicionales.

🛠 **Equipo y material de apoyo:**

- Computadora Personal (PC)
- Notepad o TextEdit
- JDK Java Development Kit

DESARROLLO DE LA PRACTICA:

Programa 1:

Una universidad necesita desarrollar un sistema para gestionar su comunidad académica. El sistema debe representar a **alumnos** y **profesores**, quienes comparten características comunes como nombre completo, dirección y fecha de nacimiento. Para modelar esto de forma eficiente, se utilizará una **clase base Persona**, de la cual heredarán las clases Alumno y Profesor.

Cada **alumno** puede estar inscrito en varias **materias**, y cada materia puede tener múltiples alumnos inscritos, por lo que se requiere una **asociación bidireccional** entre Alumno y Materia.

Además, se necesita representar una **lista de materias** en la que está inscrito cada alumno. Esta lista es **única y específica de cada alumno**, pero puede existir y ser modificada durante el tiempo de vida del alumno, por lo que se modelará como una **composición débil (agregación)** entre Alumno y ListaMaterias.

Por otra parte, cada Profesor imparte una o varias materias, y **es obligatorio** que cada materia esté asociada a **exactamente un profesor**, sin el cual la materia no tiene sentido en el sistema. Esta relación se modelará como una **composición fuerte** entre Profesor y Materia.

Finalmente, tanto Alumno como Profesor tienen un **nombre completo**, que se descompone en una clase Nombre (nombre, apellido paterno y materno), ya que esta estructura se reutiliza en varias entidades. Esta clase será **componente obligatorio**, pero puede modelarse con **composición débil**, dado que puede modificarse o intercambiarse durante la vida del objeto.

Hacer el diagram UML

Análisis de Resultados

Conclusiones

Referencias

Criterios para evaluar:

Código (70 puntos)

Programa 1

Criterio	Puntos
Implementación de la clase Libro con atributos, getters, setters y constructores	20
Implementación de la clase Biblioteca con atributos y métodos requeridos	30
Correcta funcionalidad de búsqueda y ordenación de libros	30
Uso adecuado de buenas prácticas de POO	10
Claridad en la estructura y documentación del código	10
Total	100

Programa 2

Criterio	Puntos
Implementación correcta de la clase Estudiante	20
Implementación correcta de la clase GestorEstudiantes	30

Criterio	Puntos
Funcionalidad de los métodos de manipulación de datos	30
Uso adecuado de POO y buenas prácticas	10
Documentación y claridad en el código	10
Total	**100**

Reporte (30 puntos):

- Portada, competencia u objetivo 2.5 puntos
- Pegar Código del programa (anexar archivos .java) 2.5 puntos
- Explicación completa del código y su funcionamiento 15 puntos
- Resultados (análisis de resultados) 5 puntos
- Conclusiones: 5 puntos

Nota: No se aceptarán trabajos fuera de la fecha de entrega o que no cumplan con las especificaciones de entrega requeridas. Los trabajos que se detecten copiados, serán anulados automáticamente para ambos alumnos.