Informe de aplicación de POO en Java

1. Aplicación del Principio de Encapsulamiento (Ocultación de Información)

El principio de encapsulamiento se ha aplicado rigurosamente para resguardar la consistencia y el estado interno de cada objeto. Su propósito central es proteger los datos y asegurar que toda interacción o manipulación se realice únicamente a través de una interfaz controlada y bien definida.

Variables de Instancia Privadas (private): Todas las variables de instancia (atributos) de las clases han sido declaradas con el modificador de acceso private. Este nivel de restricción es fundamental para prevenir la modificación directa, no autorizada o accidental, de los datos internos del objeto por cualquier elemento externo del código.

Métodos de Acceso Públicos (Getters y Setters - public): Se han diseñado métodos públicos (public), conocidos como getters y setters, que actúan como el único canal permitido para leer (obtener) o modificar (establecer) los valores de los atributos. Esto centraliza el control de acceso a los datos.

2. Implementación de Relaciones Estructurales entre Clases

La arquitectura de la solución se basa en una jerarquía académica estructurada mediante relaciones de Composición/Agregación. Esta técnica permite construir objetos complejos a partir de otros más simples, modelando relaciones de "tiene un/a" (has-a).

Para la administración eficiente de colecciones de objetos relacionados (como una lista de estudiantes en un curso), se adoptó la siguiente lógica de gestión:

Control de Ocupación (int contador): Se incluyó un atributo numérico, contador, cuyo rol es rastrear la cantidad exacta de elementos que han sido efectivamente añadidos a un array interno. Esto facilita la iteración precisa de la colección, evitando el procesamiento de posiciones vacías o nulas dentro del arreglo.

Mecanismos de Inserción (Agregar...): Se implementaron métodos de gestión públicos (ej: AgregarEstudiante) que aceptan una instancia del objeto a relacionar. Este método se encarga de insertar el nuevo objeto en la siguiente posición disponible del array interno, e inmediatamente incrementa el valor del contador.

3. Integración de Validaciones en los Métodos de Modificación (Setter)

Se han incorporado reglas de validación dentro de cada método setter con el fin primordial de garantizar la consistencia e integridad de la información almacenada en el objeto.

Mecanismo de Verificación: Antes de asignar un nuevo valor a un atributo, el setter evalúa si el dato propuesto cumple con los criterios de validación predefinidos (ej: un rango de edad válido, un formato de cadena específico, etc.).

Manejo de Errores:

Éxito: Si la validación es exitosa, el setter procede a asignar el valor al atributo privado.

Fallo: Si la validación falla, el setter omite la asignación del valor incorrecto. En lugar de ello, genera y muestra un mensaje de error en el stream de salida de errores (System.err.println). Esta práctica asegura que los objetos nunca almacenen datos no válidos, ya sea durante su creación o en posteriores modificaciones.