Para este proyecto, me enfoqué en tres conceptos clave: el encapsulamiento, las relaciones entre clases y la validación de datos.

#### 1. Encapsulamiento

El encapsulamiento fue mi herramienta principal para proteger los datos de la clase Estudiante. Hice lo siguiente:

* Convertí todos los atributos del estudiante (nombre, apellido, edad, etc.) a private, lo que impide que cualquier otra parte del programa acceda a ellos directamente.
* Despues, creé métodos getter y setter públicos para cada uno de los atributos. Esto me permite controlar cómo se accede y se modifica la información del estudiante, asegurando que solo se haga de la manera correcta.

#### 2. Relaciones entre Clases

Para modelar la forma en que los estudiantes se relacionan con las materias y las carreras, implementé las siguientes conexiones:

* Estudiante y Materia: Un estudiante puede tener varias materias. Para representar esta relación de uno a muchos, agregué una lista de materias (List<Materia>) a la clase Estudiante. También creé un método llamado agregarMateria() para que sea muy sencillo asignar una materia a un estudiante.
* Carrera y Estudiante: De la misma forma, una carrera agrupa a varios estudiantes. Para modelar esto, la clase Carrera ahora tiene una lista de estudiantes (List<Estudiante>). Creé métodos como agregarEstudiante(), listarEstudiantes() y buscarEstudiante() para manejar esta relación de forma eficiente.

#### 3. Validaciones en los Setters

Para asegurarme de que la información que se ingresa en el sistema es válida, agregué reglas de validación en los setters de la clase Estudiante:

* Edad: Solo se permite una edad mayor a 16 años. Si se intenta ingresar un valor menor, el programa mostrará un error.
* Nombre y Apellido: Hice que los campos de nombre y apellido no puedan estar vacíos. Si el usuario intenta dejar alguno de estos campos en blanco, el sistema lo detectará y mostrará un error.

De esta manera, me aseguré de que el proyecto no solo sea funcional, sino que también maneje la información de forma segura y consistente.