PROGRAMACIÓN II
Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Estudiante: Roqué, Gabriel Osvaldo

Matricula: 101636

Link a GitHub: https://github.com/Ozzetas/Programacion2.git

OBJETIVO GENERAL

Comprender los fundamentos de la Programación Orientada a Objetos, incluyendo clases, objetos, atributos y métodos, para estructurar programas de manera modular y reutilizable en Java.

MARCO TEÓRICO

Concepto	Aplicación en el proyecto
Clases y Objetos	Modelado de entidades como Estudiante, Mascota, Libro, Gallina y NaveEspacial
Atributos y Métodos	Definición de propiedades y comportamientos para cada clase
Estado e Identidad	Cada objeto conserva su propio estado (edad, calificación, combustible, etc.)
Encapsulamiento	Uso de modificadores de acceso y getters/setters para proteger datos
Modificadores de acceso	Uso de private, public y protected para controlar visibilidad
Getters y Setters	Acceso controlado a atributos privados mediante métodos
Reutilización de código	Definición de clases reutilizables en múltiples contextos

Caso Práctico

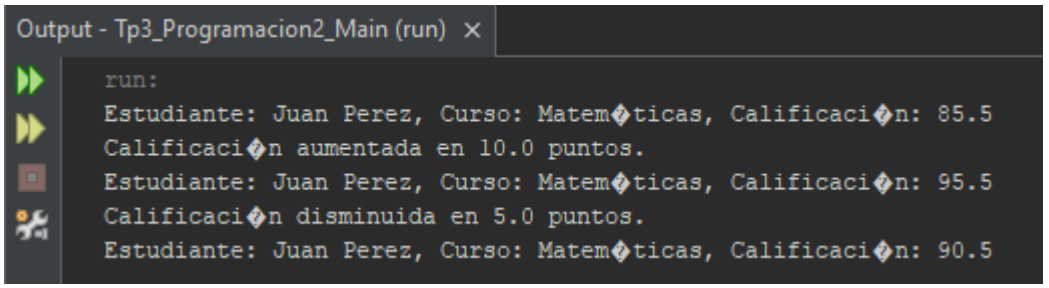
Desarrollar en Java los siguientes ejercicios aplicando los conceptos de programación orientada a objetos:

1. **Registro de Estudiantes**
   a. Crear una clase Estudiante con los atributos: nombre, apellido, curso, calificación.

   Métodos requeridos: mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos).

   Tarea: Instanciar a un estudiante, mostrar su información, aumentar y disminuir calificaciones.

   Salida:

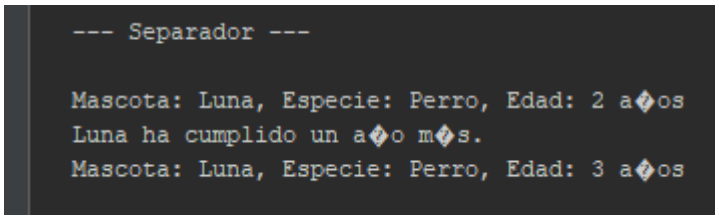


2. Registro de Mascotas
   a. Crear una clase Mascota con los atributos: nombre, especie, edad.

   Métodos requeridos: mostrarInfo(), cumplirAnios().

   Tarea: Crear una mascota, mostrar su información, simular el paso del tiempo y verificar los cambios.

   Salida:



3. Encapsulamiento con la Clase Libro
   a. Crear una clase Libro con atributos privados: titulo, autor, añoPublicacion.

   Métodos requeridos: Getters para todos los atributos. Setter con validación para añoPublicacion.

   Tarea: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.

Salida:

```
--- Separador ---

Libro: Cien a�os de soledad, Autor: Gabriel Garc�a M�rquez, A�o: 1967
A�o inv�lido: -100
Libro: Cien a�os de soledad, Autor: Gabriel Garc�a M�rquez, A�o: 1967
A�o de publicaci�n actualizado a: 1970
Libro: Cien a�os de soledad, Autor: Gabriel Garc�a M�rquez, A�o: 1970

--- Separador ---
```

4. Gestión de Gallinas en Granja Digital
    a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.

Métodos requeridos: ponerHuevo(), envejecer(), mostrarEstado().

Tarea: Crear dos gallinas, simular sus acciones (envejecer y poner huevos), y mostrar su estado.

Salida:

```
--- Separador ---

Gallina G001 ha puesto un huevo.
Gallina G001 ha envejecido un a�o.
Gallina: G001, Edad: 2, Huevos puestos: 1
Gallina G002 ha puesto un huevo.
Gallina G002 ha envejecido un a�o.
Gallina: G002, Edad: 3, Huevos puestos: 6

--- Separador ---
```

5. Simulación de Nave Espacial

   Crear una clase NaveEspacial con los atributos: nombre, combustible.

   Métodos requeridos: despegar(), avanzar(distancia), recargarCombustible(cantidad), mostrarEstado().

   Reglas: Validar que haya suficiente combustible antes de avanzar y evitar que se supere el límite al recargar.

   Tarea: Crear una nave con 50 unidades de combustible, intentar avanzar sin recargar, luego recargar y avanzar correctamente. Mostrar el estado al final.

   Salida:

```
--- Separador ---

Apolo ha avanzado 60 km.
Nave: Apolo, Combustible: 20 unidades
Se han recargado 30 unidades de combustible.
Apolo ha avanzado 20 km.
Nave: Apolo, Combustible: 40 unidades
BUILD SUCCESSFUL (total time: 0 seconds)
```

CONCLUSIONES ESPERADAS

- Comprender la diferencia entre clases y objetos.
- Aplicar principios de encapsulamiento para proteger los datos.
- Usar getters y setters para gestionar atributos privados.
- Implementar métodos que definen comportamientos de los objetos.
- Manejar el estado y la identidad de los objetos correctamente.
- Aplicar buenas prácticas en la estructuración del código orientado a objetos.
- Reforzar el pensamiento modular y la reutilización del código en Java.