# PROGRAMACIÓN II INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

Trabajo práctico N°3

#### Resumen

En este trabajo se lleva a la práctica los conceptos de clases, objetos, atributos y métodos. También se desarrollarán el encapsulamiento, getters y setters.

Nicolás Olima

nicolima200@gmail.com <u>Repositorio GitHub</u>



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

## **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

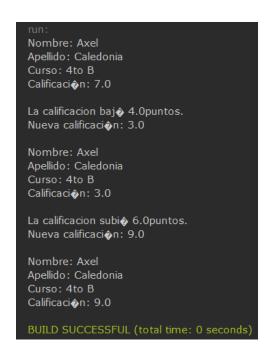
Desarrollar en Java los siguientes ejercicios aplicando los conceptos de programación orientada a objetos:

#### 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.



#### 2. Registro de Mascotas

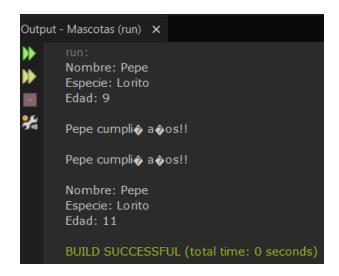
a. Crear una clase Mascota con los atributos: nombre, especie, edad.

Métodos requeridos: mostrarInfo(), cumplirAnios().

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

#### TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

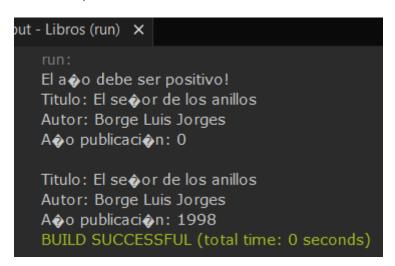




- 3. Encapsulamiento con la Clase Libro
  - a. Crear una clase Libro con atributos privados: titulo, autor, añoPublicacion.

**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.



- 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.

#### TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA



```
run:
Gallina 1 envejeci♦ 4 a os.

Gallina 2 envejeci♦ 3 a os.

Id: 1
Edad: 4
Huevos puestos: 8

Id: 2
Edad: 3
Huevos puestos: 6

BUILD SUCCESSFUL (total time: 0 seconds)
```

#### 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.

#### TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA



//// ESTADO DE LA NAVE //// Nombre de la nave: Nave del olvido Combustible: 0 lts. Intentar despegue... Combustible insuficiente para despegar! Recarga! El combustible no es suficiente para avanzar 1 unidades. Recarga completada. //// ESTADO DE LA NAVE //// Nombre de la nave: Nave del olvido Combustible: 50 lts. Intentar despegue... La nave ha despegado!! El combustible no es suficiente para avanzar 200 unidades. La nave se ha movido 50 unidades. //// ESTADO DE LA NAVE //// Nombre de la nave: Nave del olvido Combustible: 0 lts. 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.