# Trabajo Práctico 3: Introducción a la POO

## Nombre y apellido: Alfredo de Inocenti

## Repositorio GitHub: https://github.com/fonsecaFuentes/tps-Programacion_II/tree/main/Trabajo_Practico_3

## Alumno: Alfredo de Inocenti

---

*Caso Práctico*

---

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

1. **Registro de Estudiantes**
   - 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.
   - Fragmento de código:

```
public static void main(String[] args) {
    Estudiante nuevoEstudiante = new Estudiante();

    nuevoEstudiante.setNombre(nuevoNombre:"Wilson");
    nuevoEstudiante.setApellido(nuevoApellido:"Fonseca");
    nuevoEstudiante.setCurso(nuevoCurso:"9C");
    nuevoEstudiante.setCalificacion(nuevaCalificacion:10);

    nuevoEstudiante.mostrarInfo();
    nuevoEstudiante.subirCalificacion(puntos:10);
    nuevoEstudiante.mostrarInfo();
    Estudiante nuevoEstudiante - RegistroEstudiantes.Principal.main(String[])
    nuevoEstudiante.bajarCalificacion(puntos:11);
    nuevoEstudiante.mostrarInfo();
}
```

## 2. Registro de Mascotas

- 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.
- Fragmento de código:

```
Run | Debug
public static void main(String[] args) {
    Mascota miMascota = new Mascota();

    miMascota.setNombre(nuevoNombre:"Apolo");
    miMascota.setEspecie(nuevaEspecie:"perro");
    miMascota.setEdad(nuevaEdad:13);

    miMascota.mostrarInfo();
    miMascota.cumplirAnios();
    miMascota.mostrarInfo();
}
```

## 3. Encapsulamiento con la Clase Libro

- 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.
- Fragmento de código:

```
Run | Debug
public static void main(String[] args) {
    Libro libro1 = new Libro();

    libro1.setTitulo(nuevoTitulo:"Marte Rojo");
    libro1.setAutor(nombre:"Kim Stanley Robinson");
    libro1.setAnioPublicacion(-155);
    libro1.setAnioPublicacion(anio:2030);
    libro1.setAnioPublicacion(anio:1992);

    // Uso de getters
    System.out.println(x:"\nInformacion del libro(Uso de getters):");
    System.out.println("Titulo: " + libro1.getTitulo());
    System.out.println("Autor: " + libro1.getAutor());
    System.out.println("Anio de publicacion: " + libro1.getAnioPublicacion());

    // Uso de mostrarInfo()
    System.out.println(x:"\nInformacion del libro(Uso de mostrarInfo()):");
    libro1.mostrarInfo();
}
```

## 4. Gestión de Gallinas en Granja Digital

- 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.
- Fragmento de código:

```
Run | Debug
public static void main(String[] args) {
    System.out.println(x:" -------- Gallina 1 -------- ");
    // Gallina 1
    Gallina gallina1 = new Gallina();
    gallina1.setIdGallina(id:1);
    gallina1.setEdad(nuevaEdad:1);
    // Métodos
    gallina1.mostrarEstado();
    System.out.println(x:" -------- Acciones -------- ");
    gallina1.ponerHuevo();
    gallina1.envejecer();
    gallina1.mostrarEstado();

    System.out.println(x:" -------- Gallina 2 -------- ");
    // Gallina 2
    Gallina gallina2 = new Gallina();
    gallina2.setIdGallina(id:2);
    gallina2.setEdad(nuevaEdad:2);
    // Métodos
    gallina2.mostrarEstado();
    System.out.println(x:" -------- Acciones -------- ");
    gallina2.ponerHuevo();
    gallina2.ponerHuevo();
    gallina2.envejecer();
    gallina2.mostrarEstado();
}
```

## 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.
- Fragmento de código:

```
Run | Debug
public static void main(String[] args) {
    NaveEspacial enterprise = new NaveEspacial();

    enterprise.setNombre(nuevoNombre:"USS Enterprise");
    enterprise.setCombustible(cantidad:50);
    enterprise.mostrarEstado();

    System.out.println(x:"\n--- Intento 1: avanzar sin despegar (y sin recargar) ---");
    enterprise.avanzar(nuevaDistancia:60);

    System.out.println(x:"\n--- Despegue ---");
    enterprise.despegar();

    System.out.println(x:"\n--- Intento 2: avanzar 60 con solo 50 de combustible ---");
    enterprise.avanzar(nuevaDistancia:60);

    System.out.println(x:"\n--- Recarga ---");
    enterprise.recargarCombustible(cantidad:100);

    System.out.println(x:"\n--- Intento 3: avanzar correctamente ---");
    enterprise.avanzar(nuevaDistancia:60);

    System.out.println(x:"\n--- Estado final ---");
    enterprise.mostrarEstado();
}
```