PROGRAMACIÓN II

Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

01/09/2025

Alumno:

- Federico Garcia Bengolea - feddericogarciaa@gmail.com
- Comisión: M2025-14
- Link GitHub: https://github.com/FeddericoGarcia/utn-tupad-p2

Profesores:

- **Profesor**: Alberto Cortez
- **Tutor**: Ramiro Hualpa

Caso Práctico:

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.

```java
package javaapp.TP3;

public class Estudiante {

    private String nombre, apellido, curso;
    private float calificacion;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }

    public float getCalificacion() {
        return calificacion;
    }

    public void setCalificacion(float calificacion) {
        this.calificacion = calificacion >= 0 ? calificacion : 0;
    }

    public void subirCalificacion(float puntos){
        this.calificacion += puntos;
    }

    public void bajarCalificacion(float puntos){
        this.calificacion -= puntos;
    }

    public String mostrarInfo(){
        return "Nombre:" + this.nombre +
                "\nApellido: "+ this.apellido +
                "\nCurso: "+ this.curso +
                "\nCalificación: "+ this.calificacion;
    }
}
```

```
Ingresa los datos del estudiante
Nombre:
Federico
Apellido:
Garcia
Curso:
Programacion-2
Calificación:
10
Nombre:Federico
Apellido: Garcia
Curso: Programacion-2
Calificación: 10.0
bajarCalificacion(5)
subirCalificacion(3)
Nombre:Federico
Apellido: Garcia
Curso: Programacion-2
Calificación: 8.0
BUILD SUCCESSFUL (total time: 8 seconds)
```

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

```java
package javaapp.TP3;

public class Mascota {

    private String nombre, especie;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEspecie() {
        return especie;
    }

    public void setEspecie(String especie) {
        this.especie = especie;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad >= 0 ? edad : 0;
    }

    public String mostrarInfo(){
        return "Nombre: " + this.nombre +
                "\nEspecie: "+ this.especie +
                "\nEdad: "+ this.edad;
    }

    public void cumplirAnios(){
        this.edad += 1;
    }

}
```

```
Ingresa los datos de la mascota
Nombre:
Gandalf
Especie:
Gato
Edad:
3
Nombre: Gandalf
Especie: Gato
Edad: 3
... Despúes de 2 año ...
Nombre: Gandalf
Especie: Gato
Edad: 5
BUILD SUCCESSFUL (total time: 4 seconds)
```

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.

```java
package javaapp.ej5;

public class Libro {
    private String titulo, autor;
    private int anioPublicacion;

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    public void setAnioPublicacion(int anioPublicacion) {
        int anioActual = 2025;
        if(anioPublicacion <= anioActual){
            this.anioPublicacion = anioPublicacion;
        }
    }
}
```

```
run:
Ingresa los datos del libro
Titulo:
Nombre de la Rosa
Autor:
Umberto Eco
Año de publicación:
1980
Intento de modificación inválido (mayor al año actual)
2030
Valor actual de Año Publicado: 1980
Setteo Año = 2022
2022
Titulo: Nombre de la Rosa
Autor: Umberto Eco
Año de Publicación: 2022
BUILD SUCCESSFUL (total time: 28 seconds)
```

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.

```java
package javaapp.TP3;

public class Gallina {
    private int idGallina, edad, huevosPuestos;

    public int getIdGallina() {
        return idGallina;
    }

    public void setIdGallina(int idGallina) {
        this.idGallina = idGallina;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public int getHuevosPuestos() {
        return huevosPuestos;
    }

    public void setHuevosPuestos(int huevosPuestos) {
        this.huevosPuestos = huevosPuestos;
    }

    public void ponerHuevo(){
        this.huevosPuestos += 1;
    }

    public void envejecer(){
        this.edad += 1;
    }

    public void mostrarEstado(){
        System.out.println("Datos de gallina\nID: "+ this.idGallina +
                "\nEdad: "+ this.edad + "\nHuevos Puestos: "+ this.huevosPuestos);
    }
}
```


```
run:
Ejercicio #4 - Gestión de Gallinas en Granja Digital
Simulación de gallinas
Gallina1:
ID=1,
Edad=3
HuevosPuestos=500
Gallina2:
ID=2,
Edad=1
HuevosPuestos=120
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.

```java
package javaapp.TP3;

public class NaveEspacial {

    private String nombre;
    private int combustible;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getCombustible() {
        return combustible;
    }

    public void setCombustible(int combustible) {
        this.combustible = combustible;
    }

    public void despegar(){
        if (this.combustible >= 50){
            System.out.println("¡DESPEGANDO!");
        } else {
            System.out.println("No es posible despegar...");
        }
    }

    public void avanzar(int distancia){
        if(this.combustible > 0){
            System.out.println("AVANZANDO La nave va hacia la pista de lanzamiento");
        } else {
            System.out.println("Antes de avanzar necesitas recargar combustible!");
        }
    }

    public void recargarCombustible(int cantidad){
        setCombustible(cantidad);
        System.out.println("Recargando combustible: "+ cantidad + " litros");
    }

    public void mostrarEstado(){
        System.out.println("Nombre: "+ this.nombre + "\nCombustible: "+
                this.combustible+ " litros");
    }
}
```

```
run:
Ejercicio #5 - Simulación de Nave Espacial
Nombre: Nave 1
Combustible: 0 litros
Antes de avanzar necesitas recargar combustible!
No es posible despegar...
Recargando combustible: 100 litros
AVANZANDO La nave va hacia la pista de lanzamiento
¡DESPEGANDO!
Nombre: Nave 1
Combustible: 100 litros
BUILD SUCCESSFUL (total time: 0 seconds)
```