https://github.com/Coman-Che/Programacion-II/tree/main/M%C3%B3dulo%203

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.

Rta:

```
package Ejercicio_1;


public class Estudiante {

    private String nombre;
private String apellido;
private String curso;
private double 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 double getCalificacion() {
return calificacion;
    }

    public void setCalificacion(double calificacion) {
                if (calificacion < 0){
this.calificacion = 0; //Evita valores negativos
        }else if (calificacion > 10){            this.calificacion =
10; //Evita valores mayores a 10
        }else{        this.calificacion =
calificacion;
        }
    }

    //Método para mostrar información del estudiante
    //mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos)

    public void mostrarInfo(){
        System.out.println("Estudiante: " + apellido + ", " + nombre);
        System.out.println("Curso: " + curso);
        System.out.println("Calificación: " + calificacion);
    }

    //Verificamos si al sumar los puntos de calificación no supera el máxima
apermitido (10)
    public void subirCalificacion(double puntos){
if(calificacion + puntos <= 10){
calificacion += puntos; //acumulador
            System.out.println("La calificación  ha aumentado a: " + calificacion);
}else{
            System.out.println("Error, no se puede superar la calificación máxima
(10)");
        }
    }

    public void bajarCalificacion(double puntos){        if(calificacion -
puntos >= 0){            calificacion -= puntos; //asi no es menor que
0, disminuimos
califaicación
            System.out.println("La calificación  ha disminuido a: " + calificacion);
}else{
            //si <0
```

```
            System.out.println("Error, no se puede tener una calificación menor a
0");
        }
    }

 }
package Ejercicio_1;

public class Main_Ejercicio_1 {


    public static void main(String[] args) {

        Estudiante estudiante1 = new Estudiante();

        estudiante1.setNombre("Laura");
estudiante1.setApellido("Gonzalez");
estudiante1.setCurso("Lengua");
estudiante1.setCalificacion(7.5);

        System.out.println("Información Inicial:");
estudiante1.mostrarInfo();

        System.out.println("");

        System.out.println("Intentando subir calificación 1.5 puntos");
estudiante1.subirCalificacion(1.5);

        System.out.println("");

        System.out.println("Intentando subir calificación 3 puntos");
estudiante1.subirCalificacion(3);

        System.out.println("");

        System.out.println("Intentando bajar calificación 5 puntos");
estudiante1.bajarCalificacion(5);

        System.out.println("");

        System.out.println("Intentando bajar calificación 6 puntos");
estudiante1.bajarCalificacion(6);

        System.out.println("");
```

```
        System.out.println("Información Final:");
estudiante1.mostrarInfo();
    }

}
```

```
Informaci□n Inicial:
Estudiante: Gonzalez, Laura
Curso: Lengua
Calificaci□n: 7.5

Intentando subir calificaci□n 1.5 puntos
La calificaci□n  ha aumentado a: 9.0

Intentando subir calificaci□n 3 puntos
Error, no se puede superar la calificaci□n m□xima (10)

Intentando bajar calificaci□n 5 puntos
La calificaci□n  ha disminuido a: 4.0

Intentando bajar calificaci□n 6 puntos
Error, no se puede tener una calificaci□n menor a 0

Informaci□n Final:
Estudiante: Gonzalez, Laura
Curso: Lengua
Calificaci□n: 4.0
BUILD SUCCESSFUL (total time: 0 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.

Rta:
```
package
Ejercicio_2;


public class Mascota {

    private String nombre;
private String 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) {
if (edad >= 0) {            this.edad
= edad;
        } else {
            System.out.println("La edad no puede ser negativa.");
        }
    }

    // Método para simular el paso del tiempo
public void cumplirAnios(int cantidadAnios) {
if (cantidadAnios > 0) {            this.edad +=
cantidadAnios;
            String mensaje = (cantidadAnios == 1) ? "Después de " + cantidadAnios
+ " año:" : "Después de " + cantidadAnios + " años:";
            System.out.println(mensaje);
        } else {
            System.out.println("La cantidad de años debe ser positiva.");
        }
    }

    // Mostrar información
public void mostrarInfo() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad + " años");
    }
}
```

```
package Ejercicio_2;

public class Main_Ejercicio_2 {


    public static void main(String[] args) {

        Mascota mascota1 = new Mascota();

        System.out.println("Información Inicial:");

        mascota1.setNombre("Luna");
mascota1.setEspecie("Perro");
mascota1.setEdad(3);

        mascota1.mostrarInfo();
        System.out.println("");

        // Simular paso del tiempo
mascota1.cumplirAnios(1);
mascota1.mostrarInfo();
        System.out.println("");

        mascota1.cumplirAnios(3);
mascota1.mostrarInfo();
    }
}
```

```
run:
Informaci□n Inicial:
Nombre: Luna
Especie: Perro
Edad: 3 a□os

Despu□s de 1 a□o:
Nombre: Luna
Especie: Perro
Edad: 4 a□os

Despu□s de 3 a□os:
Nombre: Luna
Especie: Perro
Edad: 7 a□os
BUILD SUCCESSFUL (total time: 0 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.

Rta:

```
package
Ejercicio_3;

import java.time.Year;

public class Libro {

    private String titulo;
private String 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;
    }

    //Modificar el año de publicación con validación     public void
setAnioPublicacion(int nuevoAnio) {         int anioActual =
Year.now().getValue(); //Obtener el año actual         if (nuevoAnio
>= 1900 && nuevoAnio <= anioActual) {
this.anioPublicacion = nuevoAnio;
            System.out.println("Año de publicación actualizado correctamente");
        } else {
            System.out.println("Error: el año debe estar entre 1900 y " +
anioActual);
```

```
        }
    }

    public void mostrarInfo() {
        System.out.println("Titulo: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Año de publicación: " + anioPublicacion);
    }
 }
package Ejercicio_3;

public class Main_Ejercicio_3 {

    public static void main(String[] args) {

        Libro miLibro = new Libro();

        miLibro.setTitulo("Cómo programar en Java");
miLibro.setAutor("Deitel");
miLibro.setAnioPublicacion(2017);
        System.out.println("Mostrar información inicial:");

        miLibro.mostrarInfo();
System.out.println("");

        //Modificar el año de publicación con un valor válido

        System.out.println("Intentando cambiar el año");
miLibro.setAnioPublicacion(2000);
miLibro.mostrarInfo();        System.out.println("");


        //Modificar el año de publicación con un valor pasado inválido
System.out.println("Intentando cambiar el año");
miLibro.setAnioPublicacion(1800);        miLibro.mostrarInfo();
System.out.println("");

        //Modificar el año de publicación con un valor futuro inválido
        System.out.println("Intentando cambiar el año");
miLibro.setAnioPublicacion(2030);        miLibro.mostrarInfo();
System.out.println("");

    }

}
```

```
run:
A�o de publicaci�n actualizado correctamente
Mostrar informaci�n inicial:
Titulo: C�mo programar en Java
Autor: Deitel
A�o de publicaci�n: 2017

Intentando cambiar el a�o
A�o de publicaci�n actualizado correctamente
Titulo: C�mo programar en Java
Autor: Deitel
A�o de publicaci�n: 2000

Intentando cambiar el a�o
Error: el a�o debe estar entre 1900 y 2025
Titulo: C�mo programar en Java
Autor: Deitel
A�o de publicaci�n: 2000

Intentando cambiar el a�o
Error: el a�o debe estar entre 1900 y 2025
Titulo: C�mo programar en Java
Autor: Deitel
A�o de publicaci�n: 2000

BUILD SUCCESSFUL (total time: 0 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.

Rta:

```
package Ejercicio_4;

public class Gallina {

    private int idGallina;
    private int edad;     private int huevosPuestos;

    public int getIdGallina() {
return idGallina;
    }
```

```
    public void setIdGallina(int idGallina) {
if (idGallina > 0) {            this.idGallina =
idGallina;
        } else {
            System.out.println("El ID de la gallina debe ser positivo.");
        }
    }

    public int getEdad() {
return edad;
    }

    public void setEdad(int edad) {
if (edad >= 0) {            this.edad
= edad;
        } else {
            System.out.println("La edad no puede ser negativa.");
        }
    }

    public int getHuevosPuestos() {
return huevosPuestos;
    }

    public void setHuevosPuestos(int huevosPuestos) {
if (huevosPuestos >= 0) {
            this.huevosPuestos = huevosPuestos;
        } else {
            System.out.println("La cantidad de huevos no puede ser negativa.");
        }
    }

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

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

    public void mostrarEstado() {
        System.out.println("Gallina #" + idGallina);
        System.out.println("Edad: " + edad + " años");
        System.out.println("Huevos puestos: " + huevosPuestos);
    }
```

```
}
package Ejercicio_4;

public class Main_Ejercicio_4 {

    public static void main(String[] args) {

        Gallina gallina1 = new Gallina();
gallina1.setIdGallina(1);
gallina1.setEdad(2);
gallina1.setHuevosPuestos(0);

        // Crear segunda gallina
Gallina gallina2 = new Gallina();
gallina2.setIdGallina(2);
gallina2.setEdad(1);
gallina2.setHuevosPuestos(1);

        System.out.println("Estado inicial");
gallina1.mostrarEstado();
System.out.println("");
gallina2.mostrarEstado();
        System.out.println("");

        // Simular acciones
gallina1.envejecer();
gallina1.ponerHuevo();

        gallina2.envejecer();
gallina2.ponerHuevo();

        // Mostrar estado de ambas gallinas
System.out.println("Estado actualizado");
gallina1.mostrarEstado();
System.out.println("");
gallina2.mostrarEstado();
    }

}
```

```
run:
Estado inicial
Gallina #1
Edad: 2 a�os
Huevos puestos: 0

Gallina #2
Edad: 1 a�os
Huevos puestos: 1

Estado actualizado
Gallina #1
Edad: 3 a�os
Huevos puestos: 1

Gallina #2
Edad: 2 a�os
Huevos puestos: 2
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.

Rta:

```
package Ejercicio_5;

public class NaveEspacial {

    private String nombre;     private int combustible;     private final int MAX_COMBUSTIBLE = 100;

    public String getNombre() {
return nombre;
    }

    public int getCombustible() {
return combustible;
```

```
    }

    public void setNombre(String nombre) {        if
(nombre != null && !nombre.trim().isEmpty()) {
this.nombre = nombre;
        } else {
            System.out.println("El nombre no puede estar vacío.");
        }
    }

    public void setCombustible(int combustible) {        if (combustible
>= 0 && combustible <= MAX_COMBUSTIBLE) {
this.combustible = combustible;
        } else {
            System.out.println("El combustible debe estar entre 0 y " +
MAX_COMBUSTIBLE + ".");
        }
    }

    public void despegar() {
if (combustible > 0) {
            System.out.println("La nave " + nombre + " ha despegado.");
        } else {
            System.out.println("No hay suficiente combustible para despegar.");
        }
    }

    public void avanzar(int distancia) {
if (distancia <= 0) {
            System.out.println("La distancia debe ser positiva.");
return;
        }
        if (combustible >= distancia) {
combustible -= distancia;
            System.out.println("La nave " + nombre + " avanzó " + distancia + "
unidades.");
        } else {
            System.out.println("No hay suficiente combustible para avanzar " +
distancia + " unidades.");
        }
    }

    public void recargarCombustible(int cantidad) {
        if (cantidad > 0) {            if (combustible + cantidad <=
MAX_COMBUSTIBLE) {                combustible += cantidad;
```

```
            System.out.println("Se recargaron " + cantidad + " unidades de
combustible.");
        } else {
            combustible = MAX_COMBUSTIBLE;
            System.out.println("Se intentó sobrecargar. Combustible ajustado al
máximo (" + MAX_COMBUSTIBLE + ").");
        }
    } else {
        System.out.println("La cantidad de recarga debe ser positiva.");
    }
  }

  public void mostrarEstado() {
    System.out.println("Nave: " + nombre);
    System.out.println("Combustible actual: " + combustible + "/" +
MAX_COMBUSTIBLE);
  }
}
package Ejercicio_5;

public class Main_Ejercicio_5 {

  public static void main(String[] args) {

    NaveEspacial nave = new NaveEspacial();
nave.setNombre("Odisea");
nave.setCombustible(50);

    System.out.println("Estado inicial");
nave.mostrarEstado();
    System.out.println("");

    // Intentar avanzar sin recargar (distancia mayor al combustible disponible)
nave.avanzar(60);
    System.out.println("");

    // Recargar y avanzar correctamente
nave.recargarCombustible(30);
System.out.println("");
nave.avanzar(40);
    System.out.println("");

    // Mostrar estado final
    System.out.println("Estado actualizado");
nave.mostrarEstado();
```

```
        }
    }
```

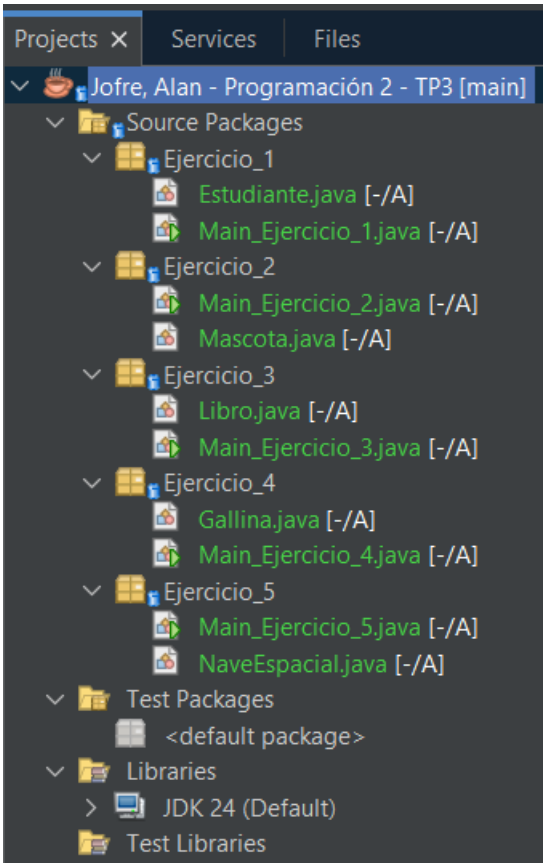
```
run:
Estado inicial
Nave: Odisea
Combustible actual: 50/100

No hay suficiente combustible para avanzar 60 unidades.

Se recargaron 30 unidades de combustible.

La nave Odisea avanz  40 unidades.

Estado actualizado
Nave: Odisea
Combustible actual: 40/100
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.