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

ALUMNO:

Franco Sarrú

Link público de GitHub:

https://github.com/fsarru/Programacion2.git

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.

```java
package sarru_tp3;

public class Estudiante {
    String nombre;
    String apellido;
    String curso;
    double calificacion;
    private double puntos;


    public void mostrarInfo() {
        System.out.println("El estudiante es: " + nombre + " " + apellido);
        System.out.println("Está cursando: " + curso);
        System.out.println("Su calificación actual es: " + calificacion);
    }

    public void subirCalificacion(double puntos) {
        calificacion += puntos;
        System.out.println("Nueva calificación: " + calificacion);
    }

    public void bajarCalificacion(double puntos) {
    this.calificacion -= puntos;
    System.out.println("Nueva calificación: " + calificacion);
    }

}
```

```java
package sarru_tp3;

public class Sarru_TP3 {


    public static void main(String[] args) {

        Estudiante a = new Estudiante();
        a.nombre = "Franco";
        a.apellido = "Sarrú";
        a.curso = "Programación II";
        a.calificacion = 8.5;


      a.mostrarInfo();
      a.subirCalificacion(1.5);
      a.bajarCalificacion(3);


    }

}
```

sarru_tp3.Sarru_TP3 > main >

Output ×

franc - C:\Users\franc × | Sarru_TP3 (run) ×

```
run:
El estudiante es: Franco Sarr
Est cursando: Programaci n II
Su calificaci n actual es: 8.5
Nueva calificaci n: 10.0
Nueva calificaci n: 7.0
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.

```java
/*EJERCICIO2:
a.      Crear una clase Mascota con los atributos: nombre, especie, edad.
Métodos requeridos: mostrarInfo(), cumplirAnios().

 */
package sarru_tp3_ej2;

public class Mascota {

    String nombre;
    String especie;
    int edad;

    public void mostrarInfo() {
        System.out.println("Las mascota se llama " + nombre + ", es un " + especie + " y tiene " + edad + " años.");
    }

    public void cumplirAnios() {
        edad = edad + 1;
        System.out.println("¡Es el cumpleaños de " + nombre + "! Felicidades!! Ahora tiene " + edad + " años.");



    }

}
```

```java
/*EJERCICIO 2:
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.

 */
package sarru_tp3_ej2;

public class Sarru_tp3_ej2 {

    public static void main(String[] args) {
        Mascota a = new Mascota();
        a.nombre = "Pipo";
        a.especie = "perro";
        a.edad = 1;


        a.mostrarInfo();
        a.cumplirAnios();
        a.cumplirAnios();
        a.cumplirAnios();


    }

}
```

```
sarru_tp3_ej2.Sarru_tp3_ej2 > main >
Output ×
franc - C:\Users\franc ×   sarru_tp3_ej2 (run) ×
run:
Las mascota se llama Pipo, es un perro y tiene 1 a♦os.
♦Es el cumplea♦os de Pipo! Felicidades!! Ahora tiene 2 a♦os.
♦Es el cumplea♦os de Pipo! Felicidades!! Ahora tiene 3 a♦os.
♦Es el cumplea♦os de Pipo! Felicidades!! Ahora tiene 4 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.

```java
package sarru_tp3_ej3;

public class Libro {

    private String titulo;
    private String autor;
    private int anioPublicacion;


    public String getTitulo() {
        return this.titulo;
    }

    public String getAutor() {
        return this.autor;
    }

    public int getAnioPublicacion() {
        return this.anioPublicacion;
    }


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

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

    public void setanioPublicacion(int anioPublicacion) {
        this.anioPublicacion = anioPublicacion;
    }
```

```java
    // Setter con validación para el año
    public void setcambioAnioPublicacion(int anio) {
        if (anio > 0 && anio <= 2025) {
            this.anioPublicacion = anio;
            System.out.println("El año de publicacion se actualizo correctamente a " + anio + ".");
        } else {
            System.out.println("Error: El año de publicacion " + anio + " no es valido.");
        }
    }

    public void mostrarInfo() {
        System.out.println("El nombre del libro es: " + titulo);
        System.out.println("El autor es: " + autor);
        System.out.println("El año de publicacion fue: " + anioPublicacion);
    }
}
```

```java
package sarru_tp3_ej3;

public class Sarru_TP3_ej3 {

    public static void main(String[] args) {

        Libro a = new Libro();
        a.setTitulo("Fundamentos de la Programacion");
        a.setAutor("Jose Martin Fierro");
        a.setanioPublicacion(2025);

         a.mostrarInfo();


         // Cambio de año incorrecto
        a.setcambioAnioPublicacion(2026);

         // Cambio de año correcto
        a.setcambioAnioPublicacion(2020);
        a.mostrarInfo();
    }

}
```

```
sarru_tp3_ej3.Sarru_TP3_ej3 > main >
Output ×
franc - C:\Users\franc ×   sarru_TP3_ej3 (run) ×
run:
El nombre del libro es: Fundamentos de la Programacion
El autor es: Jose Martin Fierro
El a□o de publicacion fue: 2025
Error: El a□o de publicacion 2026 no es valido.
El a□o de publicacion se actualizo correctamente a 2020.
El nombre del libro es: Fundamentos de la Programacion
El autor es: Jose Martin Fierro
El a□o de publicacion fue: 2020
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.

```java
package sarru_tp3_ej4;

public class Gallina {


    int idGallina;
    int edad;
    int huevosPuestos;

    public void mostrarEstado() {
        System.out.println("La gallina con ID: " + idGallina + " tiene " + edad + " meses y ha puesto " + huevosPuestos + " huevos.");

    }

    public void envejecer(int meses) {
        edad += meses;
        System.out.println("La gallina " + idGallina + " ahora tiene: " + edad + " meses");
    }

    public void ponerHuevo(int huevos) {
        huevosPuestos += huevos;
        System.out.println("La gallina " + idGallina + " acumula " + huevosPuestos + " huevos puestos.");
    }
}
```

```java
package sarru_tp3_ej4;

public class Sarru_TP3_ej4 {

    public static void main(String[] args) {

        Gallina a = new Gallina();
        a.idGallina = 250501;
        a.edad = 5;
        a.huevosPuestos = 0;

        Gallina b = new Gallina();
        b.idGallina = 91218;
        b.edad = 2;
        b.huevosPuestos = 3;

        a.mostrarEstado();
        b.mostrarEstado();


      a.envejecer(2);
      a.ponerHuevo(5);
      b.envejecer(8);
      b.ponerHuevo(14);

        a.mostrarEstado();
        b.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.

```java
package sarru_tp3_ej5;

public class NaveEspacial {


    private String nombre;
    private double combustible;
    private final double CAPACIDAD_MAXIMA = 100.0;

    public void despegar() {
        System.out.println("La nave " + nombre + " ha despegado!");
    }


    public void avanzar(double distancia) {
        double combustibleNecesario = distancia * 0.5; // Defini un consumo por cantidad de km.
        if (combustible >= combustibleNecesario) {
            combustible -= combustibleNecesario;
            System.out.println("La nave " + nombre + " avanzo " + distancia + " km.");
        } else {
            System.out.println("No hay suficiente combustible para avanzar " + distancia + " km. Necesitas al menos " + combustibleNecesario + " unidades.");
        }
    }
```

```java
    public void recargarCombustible(double cantidad) {
        if (combustible + cantidad > CAPACIDAD_MAXIMA) {
            System.out.println("No se puede recargar. La cantidad supera la capacidad maxima!");
        } else {
            combustible += cantidad;
            System.out.println("Se recargaron " + cantidad + " unidades de combustible.");
        }
    }

    // Método para mostrar el estado actual
    public void mostrarEstado() {
        System.out.println("Combustible actual: " + this.combustible + " / " + CAPACIDAD_MAXIMA + " unidades.");
    }

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

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

```java
package sarru_tp3_ej5;


public class Sarru_TP3_ej5 {


    public static void main(String[] args) {


        NaveEspacial a = new NaveEspacial();


        a.setNombre("SpaceF");
        a.setCombustible(50.0);


        a.mostrarEstado();


        a.despegar();



        a.avanzar(150.0);
        a.mostrarEstado();
```

```
        a.despegar();


        a.avanzar(150.0);
        a.mostrarEstado();


        a.recargarCombustible(60.0);


        a.recargarCombustible(40.0);


        a.avanzar(150.0);


        a.mostrarEstado();
    }
}
```

```
sarru_tp3_ej5.Sarru_TP3_ej5 > main >
Output ×
franc - C:\Users\franc ×   Sarru_TP3_ej5 (run) ×
run:
Combustible actual: 50.0 / 100.0 unidades.
La nave SpaceF ha despegado!
No hay suficiente combustible para avanzar 150.0 km. Necesitas al menos 75.0 unidades.
Combustible actual: 50.0 / 100.0 unidades.
No se puede recargar. La cantidad supera la capacidad maxima!
Se recargaron 40.0 unidades de combustible.
La nave SpaceF avanzo 150.0 km.
Combustible actual: 15.0 / 100.0 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.