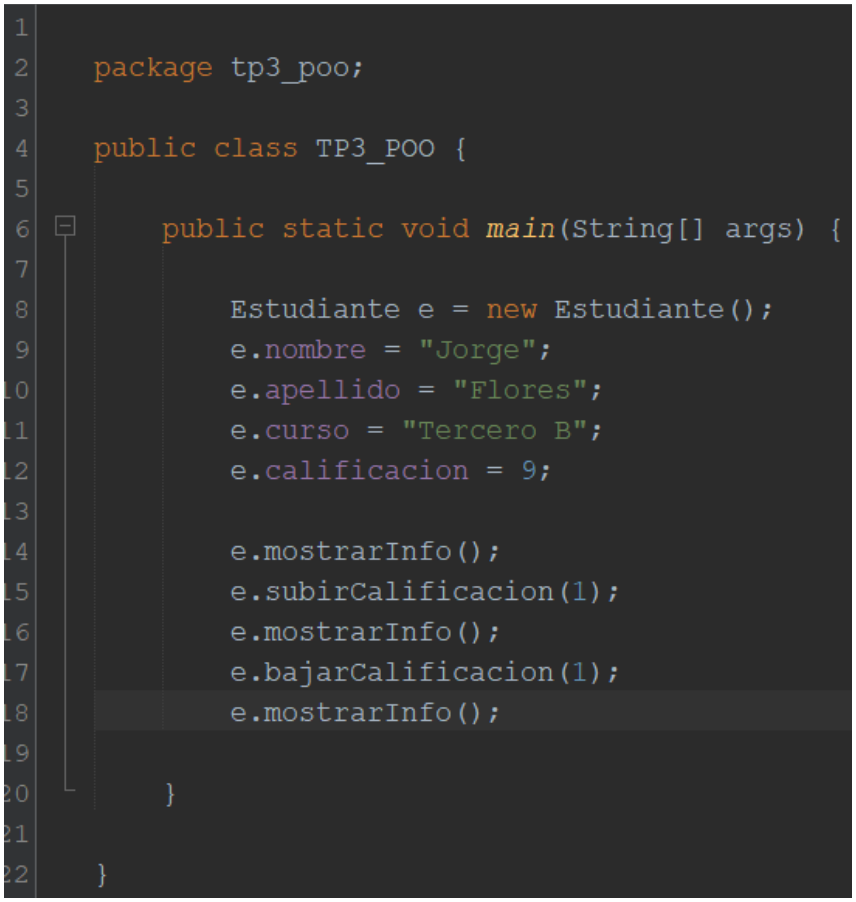
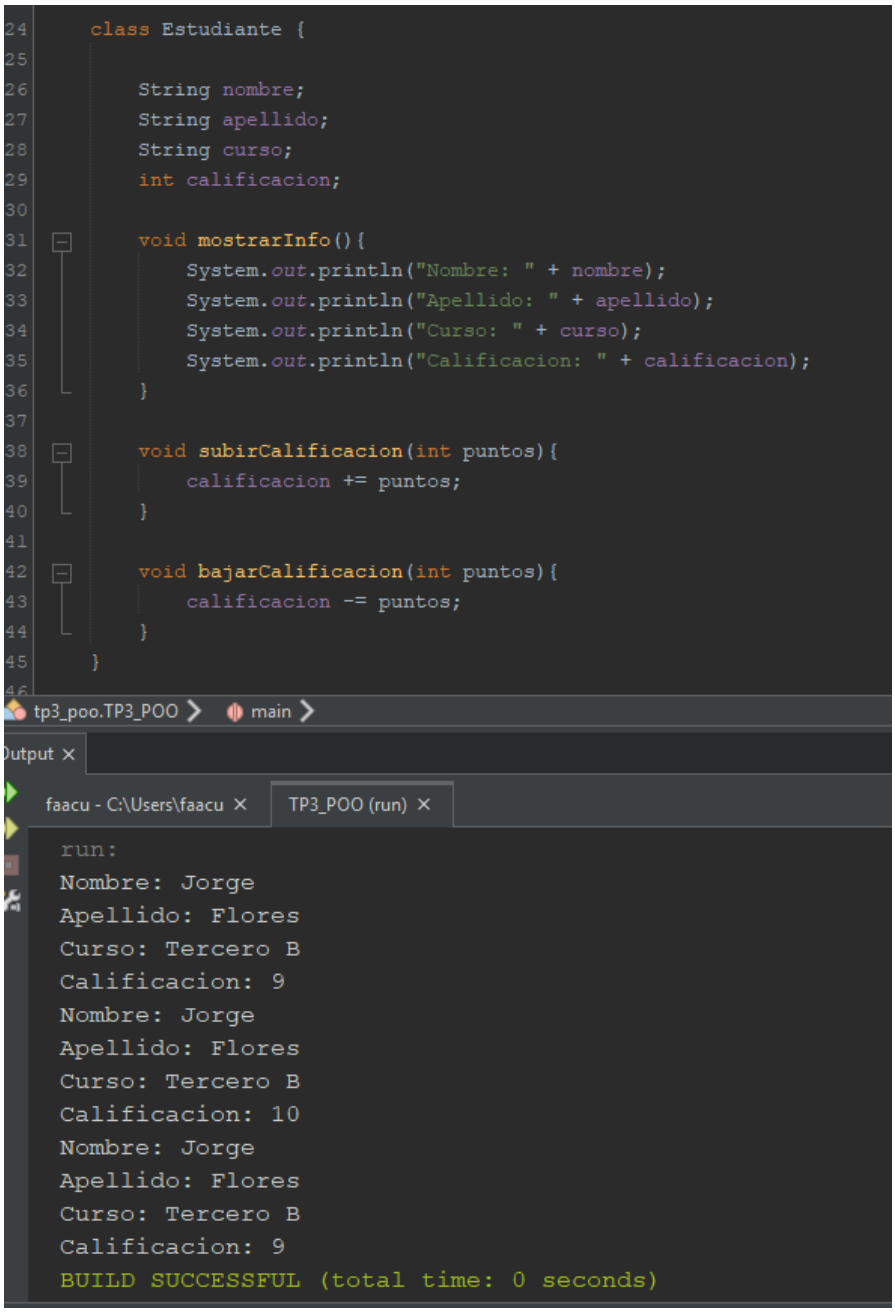
PROGRAMACIÓN II Trabajo Práctico 3:

Introducción a la 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.



```
1
2   package tp3_poo;
3
4   public class TP3_POO {
5
6       public static void main(String[] args) {
7
8           Estudiante e = new Estudiante();
9           e.nombre = "Jorge";
10          e.apellido = "Flores";
11          e.curso = "Tercero B";
12          e.calificacion = 9;
13
14          e.mostrarInfo();
15          e.subirCalificacion(1);
16          e.mostrarInfo();
17          e.bajarCalificacion(1);
18          e.mostrarInfo();
19
20      }
21
22  }
```



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 ejercicio2;

public class Ejercicio2 {

    public static void main(String[] args) {

        Mascotas m = new Mascotas();

        m.nombre = "Pancho";
        m.especie = "Perro";
        m.edad = 2;

        m.mostrarInfo();
        m.cumplirAnios(5);
        m.mostrarInfo();
    }

}
```

```java
}
    class Mascotas{

    String nombre;
    String especie;
    int edad;

    void mostrarInfo(){
        System.out.println("Nombre: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad);
    }

    void cumplirAnios(int anios){
        edad += anios;
    }
}
```

ejercicio2.Mascotas > cumplirAnios >

Output ×

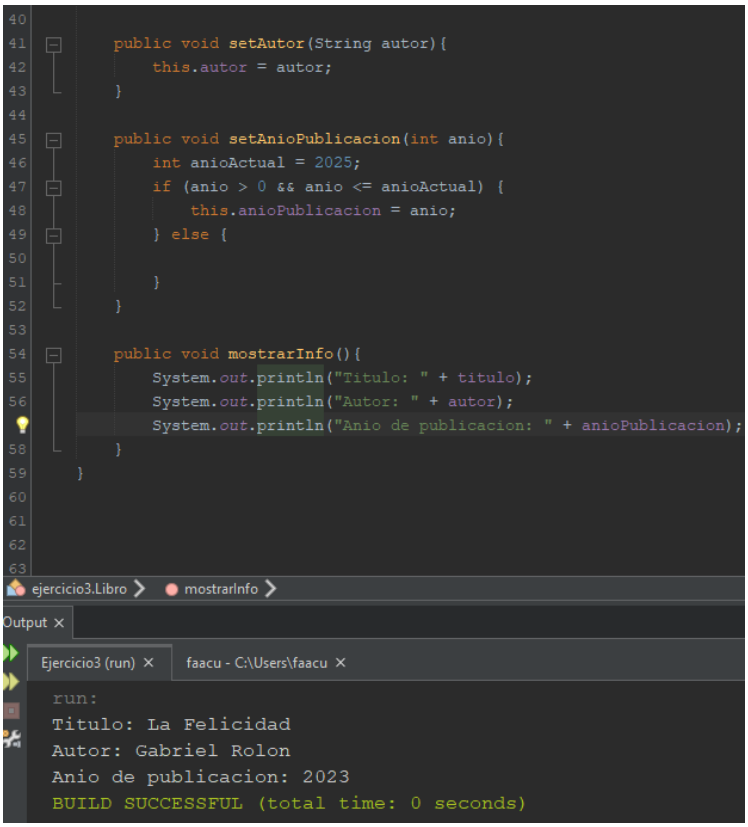
faacu - C:\Users\faacu × | Ejercicio2 (run) ×

```
run:
Nombre: Pancho
Especie: Perro
Edad: 2
Nombre: Pancho
Especie: Perro
Edad: 7
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 ejercicio3;

public class Ejercicio3 {

    public static void main(String[] args) {
        Libro li = new Libro();
         li.setTitulo("La Felicidad");
         li.setAutor("Gabriel Rolon");

         li.setAnioPublicacion(2999);
         li.setAnioPublicacion(2023);

         li.mostrarInfo();

    }


}
```

```java

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

     public String getTitulo(){
         return titulo;
 }
     public String getAutor(){
         return autor;
     }

     public int getAnioPublicacion(){
         return anioPublicacion;
     }

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

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



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

    public void setAnioPublicacion(int anio){
        int anioActual = 2025;
        if (anio > 0 && anio <= anioActual) {
            this.anioPublicacion = anio;
        } else {

        }
    }

    public void mostrarInfo(){
        System.out.println("Titulo: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Anio de publicacion: " + anioPublicacion);
    }
}
```

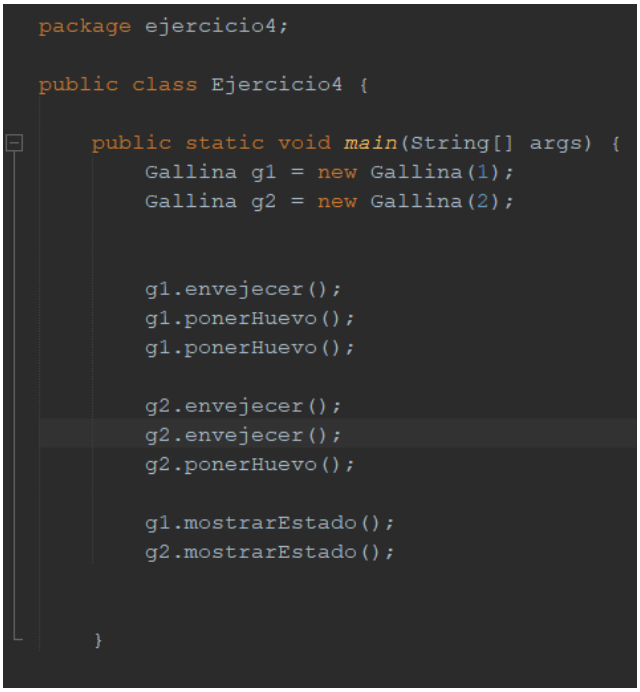
ejercicio3.Libro > mostrarInfo >

Output ×

Ejercicio3 (run) × faacu - C:\Users\faacu ×

```
run:
Titulo: La Felicidad
Autor: Gabriel Rolon
Anio de publicacion: 2023
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.



```
package ejercicio4;

public class Ejercicio4 {

    public static void main(String[] args) {
        Gallina g1 = new Gallina(1);
        Gallina g2 = new Gallina(2);


        g1.envejecer();
        g1.ponerHuevo();
        g1.ponerHuevo();

        g2.envejecer();
        g2.envejecer();
        g2.ponerHuevo();

        g1.mostrarEstado();
        g2.mostrarEstado();


    }
```

```
26      }
27
28      class Gallina{
29
30          private int idGallina;
31          private int edad;
32          private int huevosPuestos;
33
34          public Gallina(int idGallina){
35              this.idGallina = idGallina;
36              this.edad = edad;
37              this.huevosPuestos = huevosPuestos;
38          }
39
40          public int getIdGallina(){
41              return idGallina;
42          }
43
44          public int getEdad(){
45              return edad;
46          }
47
48          public int getHuevosPuestos(){
49              return huevosPuestos;
```

```
51
52          public void envejecer(){
53              edad++;
54          }
55
56          public void ponerHuevo(){
57              huevosPuestos++;
58          }
59
60          public void mostrarEstado(){
61              System.out.println("Gallina: " + idGallina);
62              System.out.println("Edad: " + edad);
63              System.out.println("Huevos puestos: " + huevosPuestos);
64          }
65      }
```

```
ejercicio4.Ejercicio4 > main >
utput ×
faacu - C:\Users\faacu ×   Ejercicio4 (run) ×
run:
Gallina: 1
Edad: 1
Huevos puestos: 2
Gallina: 2
Edad: 2
Huevos puestos: 1
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.

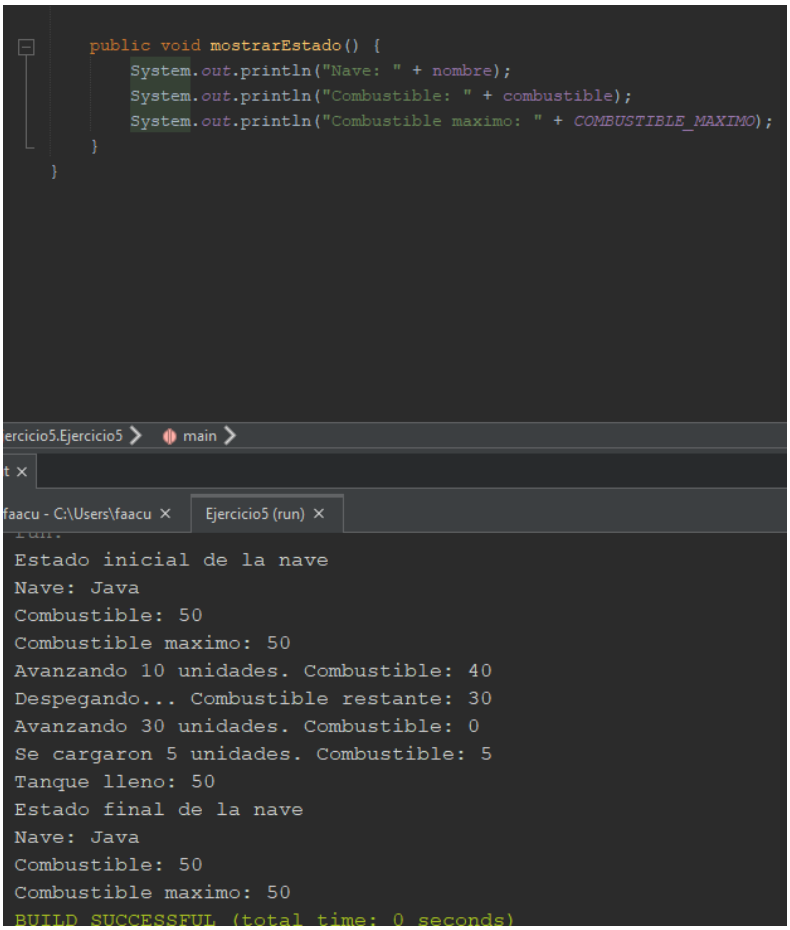
```java
package ejercicio5;

public class Ejercicio5 {
    public static void main(String[] args) {
        NaveEspacial nave = new NaveEspacial("Java", 50);

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

        nave.avanzar(10);
        nave.despegar();
        nave.avanzar(30);
        nave.recargarCombustible(5);
        nave.recargarCombustible(100);

        System.out.println("Estado final de la nave");
        nave.mostrarEstado();
    }
}
```

```java
class NaveEspacial {
    private String nombre;
    private int combustible;


    private static final int COMBUSTIBLE_MAXIMO = 50;
    private static final int COSTO_DESPEGUE = 10;
    private static final int CONSUMO_POR_DISTANCIA = 1;


    public NaveEspacial(String nombre, int combustibleInicial) {
        this.nombre = nombre;

        if (combustibleInicial < 0) {
            this.combustible = 0;
        } else if (combustibleInicial > COMBUSTIBLE_MAXIMO) {
            this.combustible = COMBUSTIBLE_MAXIMO;
        } else {
            this.combustible = combustibleInicial;
        }
    }
```

```java
    public String getNombre() {
        return nombre;
    }

    public int getCombustible() {
        return combustible;
    }


    public void despegar() {
        if (combustible < COSTO_DESPEGUE) {
            System.out.println("No hay combustible suficiente para despegar. Se necesitan " + COSTO_DESPEGUE);
            return;
        }
        combustible = combustible - COSTO_DESPEGUE;
        System.out.println("Despegando... Combustible restante: " + combustible);
    }

    public void avanzar(int distancia) {
        if (distancia <= 0) {
            System.out.println("La distancia debe ser mayor que 0.");
```

```java
    public void avanzar(int distancia) {
        if (distancia <= 0) {
            System.out.println("La distancia debe ser mayor que 0.");
            return;
        }

        int consumo = distancia * CONSUMO_POR_DISTANCIA;

        if (consumo > combustible) {
            System.out.println("No alcanza el combustible para avanzar " + distancia + ". Disponible: " + combustible);
            return;
        }
        combustible = combustible - consumo;
        System.out.println("Avanzando " + distancia + " unidades. Combustible: " + combustible);
    }
```

```java
    public void recargarCombustible(int cantidad) {
        if (cantidad <= 0) {
            System.out.println("La cantidad a recargar debe ser mayor que 0.");
            return;
        }

        combustible = combustible + cantidad;

        if (combustible > COMBUSTIBLE_MAXIMO) {
            combustible = COMBUSTIBLE_MAXIMO;
            System.out.println("Tanque lleno: " + COMBUSTIBLE_MAXIMO);
        } else {
            System.out.println("Se cargaron " + cantidad + " unidades. Combustible: " + combustible);
        }
    }
```



Link repo github: https://github.com/FacuAuciello/UTN_JAVA_2do_Cuatri