

## 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 }
```

```
24     class Estudiante {
25
26         String nombre;
27         String apellido;
28         String curso;
29         int calificacion;
30
31         void mostrarInfo() {
32             System.out.println("Nombre: " + nombre);
33             System.out.println("Apellido: " + apellido);
34             System.out.println("Curso: " + curso);
35             System.out.println("Calificacion: " + calificacion);
36         }
37
38         void subirCalificacion(int puntos) {
39             calificacion += puntos;
40         }
41
42         void bajarCalificacion(int puntos) {
43             calificacion -= puntos;
44         }
45     }
46
47 tp3_poo.TP3_POO > main >
48
49 Output x
50
51 faacu - C:\Users\faacu x TP3_POO (run) x
52
53 run:
54 Nombre: Jorge
55 Apellido: Flores
56 Curso: Tercero B
57 Calificacion: 9
58 Nombre: Jorge
59 Apellido: Flores
60 Curso: Tercero B
61 Calificacion: 10
62 Nombre: Jorge
63 Apellido: Flores
64 Curso: Tercero B
65 Calificacion: 9
66 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.

```

1  package ejercicio2;
2
3  public class Ejercicio2 {
4
5      public static void main(String[] args) {
6
7          Mascotas m = new Mascotas();
8
9          m.nombre = "Pancho";
10         m.especie = "Perro";
11         m.edad = 2;
12
13         m.mostrarInfo();
14         m.cumplirAnios(5);
15         m.mostrarInfo();
16     }
17
18 }

```

```

18 }
19
20 class Mascotas{
21
22     String nombre;
23     String especie;
24     int edad;
25
26     void mostrarInfo(){
27         System.out.println("Nombre: " + nombre);
28         System.out.println("Especie: " + especie);
29         System.out.println("Edad: " + edad);
30     }
31
32     void cumplirAnios(int anios){
33         edad += anios;
34     }
35 }

```

ejercicio2.Mascotas > cumplirAnios >

Output x

```

faacu - C:\Users\faacu X  Ejercicio2 (run) X
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.

```
1
2 package ejercicio3;
3
4 public class Ejercicio3 {
5
6     public static void main(String[] args) {
7         Libro li = new Libro();
8         li.setTitulo("La Felicidad");
9         li.setAutor("Gabriel Rolon");
10
11         li.setAnioPublicacion(2999);
12         li.setAnioPublicacion(2023);
13
14         li.mostrarInfo();
15     }
16 }
17
18
19 }
```

```
20
21 class Libro{
22     private String titulo;
23     private String autor;
24     private int anioPublicacion;
25
26     public String getTitulo(){
27         return titulo;
28     }
29     public String getAutor(){
30         return autor;
31     }
32
33     public int getAnioPublicacion(){
34         return anioPublicacion;
35     }
36
37     public void setTitulo(String titulo){
38         this.titulo = titulo;
39     }
40
41     public void setAutor(String autor){
42         this.autor = autor;
43     }
44 }
```

```
40
41 public void setAutor(String autor) {
42     this.autor = autor;
43 }
44
45 public void setAnioPublicacion(int anio) {
46     int anioActual = 2025;
47     if (anio > 0 && anio <= anioActual) {
48         this.anioPublicacion = anio;
49     } else {
50
51     }
52 }
53
54 public void mostrarInfo() {
55     System.out.println("Titulo: " + titulo);
56     System.out.println("Autor: " + autor);
57     System.out.println("Anio de publicacion: " + anioPublicacion);
58 }
59 }
60
61
62
63
```

ejercicio3.Libro > mostrarInfo >

Output x

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

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 >

Output X

faacu - C:\Users\faacu X Ejercicio4 (run) X

```

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.

```

1  package ejercicio5;
2
3  public class Ejercicio5 {
4      public static void main(String[] args) {
5          NaveEspacial nave = new NaveEspacial("Java", 50);
6
7          System.out.println("Estado inicial de la nave");
8          nave.mostrarEstado();
9
10         nave.avanzar(10);
11         nave.despegar();
12         nave.avanzar(30);
13         nave.recargarCombustible(5);
14         nave.recargarCombustible(100);
15
16         System.out.println("Estado final de la nave");
17         nave.mostrarEstado();
18     }
19 }

```

```

21  class NaveEspacial {
22      private String nombre;
23      private int combustible;
24
25
26      private static final int COMBUSTIBLE_MAXIMO = 50;
27      private static final int COSTO_DESPEGUE = 10;
28      private static final int CONSUMO_POR_DISTANCIA = 1;
29
30
31      public NaveEspacial(String nombre, int combustibleInicial) {
32          this.nombre = nombre;
33
34          if (combustibleInicial < 0) {
35              this.combustible = 0;
36          } else if (combustibleInicial > COMBUSTIBLE_MAXIMO) {
37              this.combustible = COMBUSTIBLE_MAXIMO;
38          } else {
39              this.combustible = combustibleInicial;
40          }
41      }

```

```

43 public String getNombre() {
44     return nombre;
45 }
46
47 public int getCombustible() {
48     return combustible;
49 }
50
51
52 public void despegar() {
53     if (combustible < COSTO_DESPEGUE) {
54         System.out.println("No hay combustible suficiente para despegar. Se necesitan " + COSTO_DESPEGUE);
55         return;
56     }
57     combustible = combustible - COSTO_DESPEGUE;
58     System.out.println("Despegando... Combustible restante: " + combustible);
59 }
60
61 public void avanzar(int distancia) {
62     if (distancia <= 0) {
63         System.out.println("La distancia debe ser mayor que 0.");

```

```

1 public void avanzar(int distancia) {
2     if (distancia <= 0) {
3         System.out.println("La distancia debe ser mayor que 0.");
4         return;
5     }
6
7     int consumo = distancia * CONSUMO_POR_DISTANCIA;
8
9     if (consumo > combustible) {
10        System.out.println("No alcanza el combustible para avanzar " + distancia + ". Disponible: " + combustible);
11        return;
12    }
13    combustible = combustible - consumo;
14    System.out.println("Avanzando " + distancia + " unidades. Combustible: " + combustible);
15 }

```

```

1 public void recargarCombustible(int cantidad) {
2     if (cantidad <= 0) {
3         System.out.println("La cantidad a recargar debe ser mayor que 0.");
4         return;
5     }
6
7     combustible = combustible + cantidad;
8
9     if (combustible > COMBUSTIBLE_MAXIMO) {
10        combustible = COMBUSTIBLE_MAXIMO;
11        System.out.println("Tanque lleno: " + COMBUSTIBLE_MAXIMO);
12    } else {
13        System.out.println("Se cargaron " + cantidad + " unidades. Combustible: " + combustible);
14    }
15 }

```



```
public void mostrarEstado() {
    System.out.println("Nave: " + nombre);
    System.out.println("Combustible: " + combustible);
    System.out.println("Combustible maximo: " + COMBUSTIBLE_MAXIMO);
}
```

Ejercicio5.Ejercicio5 > main >

faacu - C:\Users\faacu x Ejercicio5 (run) x

Estado inicial de la nave  
Nave: Java  
Combustible: 50  
Combustible maximo: 50  
Avanzando 10 unidades. Combustible: 40  
Despegando... Combustible restante: 30  
Avanzando 30 unidades. Combustible: 0  
Se cargaron 5 unidades. Combustible: 5  
Tanque lleno: 50  
Estado final de la nave  
Nave: Java  
Combustible: 50  
Combustible maximo: 50  
BUILD SUCCESSFUL (total time: 0 seconds)

Link repo github: [https://github.com/FacuAuciello/UTN\\_JAVA\\_2do\\_Cuatri](https://github.com/FacuAuciello/UTN_JAVA_2do_Cuatri)