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

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.

```
11  public class Main {
12      public static void main(String[] args) {
13          Estudiante e = new Estudiante("Juan", "Perez", "Programacion II", 6.0);
14
15          e.mostrarInfo();
16          e.subirCalificacion(2.5);
17          e.mostrarInfo();
18          e.bajarCalificacion(5.0);
19          e.mostrarInfo();
20      }
21  }
```

Ejercicio1.Main > main >

Output - TrabajoPractico3 (run)

```
run:
Perez, Juan - Curso: Programacion II - Calificación: 6.0
Perez, Juan - Curso: Programacion II - Calificación: 8.5
Perez, Juan - Curso: Programacion II - Calificación: 3.5
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.

```
11
12  public class Main {
13      public static void main(String[] args) {
14          Mascota m = new Mascota("Luna", "Perro", 2);
15
16          m.mostrarInfo();
17
18          m.cumplirAnios();
19          m.cumplirAnios();
20
21          m.mostrarInfo();
22      }
23  }
24
```

Ejercicio2.Main > main >

```
Output - TrabajoPractico3 (run)
run:
Mascota: Luna | Especie: Perro | Edad: 2 año(s)
Mascota: Luna | Especie: Perro | Edad: 4 año(s)
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.

```
public class Main {
    public static void main(String[] args) {
        Libro l1 = new Libro("Un libro", "Autor anonimo", 1943);
        l1.mostrarInfo();

        Libro l2 = new Libro("Libro Futuro", "Autor X", -200);
        l2.mostrarInfo();

        l1.setTitulo("Un libro (Edición especial)");
        l1.mostrarInfo();
    }
}
```

Ejercicio3.Main > main > l1 >

utput - TrabajoPractico3 (run) ×

```
run:
Libro: "Un libro" de Autor anonimo (1943)
Libro: "Libro Futuro" de Autor X (0)  --- Año de publicación ingresado inválido ---
Libro: "Un libro (Edición especial)" de Autor anonimo (1943)
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.

```
11  public class Main {
12
13      public static void main(String[] args) {
14
15          Gallina g1 = new Gallina(1, 1);
16          Gallina g2 = new Gallina(2, 2);
17
18          g1.mostrarEstado();
19          g2.mostrarEstado();
20
21          g1.envejecer();
22          g1.ponerHuevo();
23          g1.ponerHuevo();
24
25          g2.ponerHuevo();
26          g2.envejecer();
27          g2.ponerHuevo();
28          g2.ponerHuevo();
29
30          g1.mostrarEstado();
31          g2.mostrarEstado();
32      }
33  }
```

Ejercicio4.Main > main > g2 >

```
Output - TrabajoPractico3 (run)
run:
Gallina #1 | Edad: 1 año(s) | Huevos puestos: 0
Gallina #2 | Edad: 2 año(s) | Huevos puestos: 0
Gallina #1 | Edad: 2 año(s) | Huevos puestos: 2
Gallina #2 | Edad: 3 año(s) | Huevos puestos: 3
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.

```
public class Main {

    public static void main(String[] args) {
        NaveEspacial nave = new NaveEspacial(50);
        nave.despegar();
        nave.mostrarEstado();

        nave.avanzar(60);
        nave.mostrarEstado();

        nave.recargar(20);
        nave.mostrarEstado();

        nave.avanzar(30);
        nave.mostrarEstado();

        nave.recargar(200);
        nave.recargar(-10);
    }
}
```

```
Ejercicio5.Main > main >
tput - TrabajoPractico3 (run) ×
run:
Despegando!!
Combustible actual: 50 / 100 L
Combustible insuficiente para avanzar 60 km.
Combustible actual: 50 / 100 L
Recargando combustible.
Combustible actual: 70 / 100 L
La nave avanzó 30 km.
Combustible actual: 40 / 100 L
Recargando combustible.
Se superó la capacidad máxima de carga. Topeado a 100 L.
La cantidad a recargar debe ser positiva.
BUILD SUCCESSFUL (total time: 0 seconds)
```