

1. Registro de Estudiantes

- Crear una clase Estudiante con los atributos: nombre, apellido, curso, calificación.

Métodos requeridos: mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos).

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package com.myccompany.ejerciciouno;

/**
 *
 * @author lucko
 */
public class Estudiante {
    String nombre;
    String apellido;
    String curso;
    double calificacion;

    void mostrarInfo(){
        System.out.println("nombre: " + nombre);
        System.out.println("apellido: " + apellido);
        System.out.println("curso: " + curso);
        System.out.println("calificacion: " + calificacion);
    }

    void subirCalificacion(double puntos){
        if(puntos > 0){
            calificacion = calificacion + puntos;
        }
    }

    void bajarCalificacion(double puntos){
        if(puntos > 0){
            calificacion = calificacion - puntos;
        }
    }
}
```

Tarea: Instanciar a un estudiante, mostrar su información, aumentar y disminuir calificaciones.

```
package com.mycompany.ejerciciouno;

/**
 *
 * @author luCko
 */
public class EjercicioUno {

    public static void main(String[] args) {
        Estudiante est = new Estudiante();
        est.nombre = "mariano";
        est.apellido = "barrera";
        est.curso = "3º C";
        est.calificacion = 7.5;
        est.mostrarInfo();

        est.subirCalificacion(2.5);
        est.mostrarInfo();

        est.bajarCalificacion(5.5);
        est.mostrarInfo();
    }
}
```

```
--- exec:3.1.0:exec (default-cli) @ EjercicioUno ---
nombre: mariano
apellido: barrera
curso: 3♦ C
calificacion: 7.5
nombre: mariano
apellido: barrera
curso: 3♦ C
calificacion: 10.0
nombre: mariano
apellido: barrera
curso: 3♦ C
calificacion: 4.5

-----
BUILD SUCCESS
-----

Total time: 2.789 s
Finished at: 2025-08-26T15:59:41-03:00
-----
```

2. Registro de Mascotas

- Crear una clase Mascota con los atributos: nombre, especie, edad.

Métodos requeridos: mostrarInfo(), cumplirAnios().

```

    * @author lucko
    */
    public class Mascota {
        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() {
            edad++;
        }
    }

```

Tarea: Crear una mascota, mostrar su información, simular el paso del tiempo y verificar los cambios.

```

    * @author lucko
    */
    public class EjercicioDos {

        public static void main(String[] args) {
            Mascota mascota = new Mascota();
            mascota.nombre = "peluca";
            mascota.especie = "caniche";
            mascota.edad = 7;

            mascota.mostrarInfo();

            mascota.cumplirAnios();
            mascota.mostrarInfo();
            |
            mascota.cumplirAnios();
            mascota.mostrarInfo();
        }
    }

```

```
nombre: peluca  
especie: caniche  
edad: 7  
nombre: peluca  
especie: caniche  
edad: 8  
nombre: peluca  
especie: caniche  
edad: 9
```

```
-----  
BUILD SUCCESS  
-----
```

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.

```

    */
    public class Libro {
        private String titulo;
        private String autor;
        private int añoPublicacion;

        public String getTitulo() {
            return titulo;
        }

        public String getAutor() {
            return autor;
        }

        public int getAñoPublicacion() {
            return añoPublicacion;
        }

        public void setAñoPublicacion(int año) {
            if (año > 0) {
                this.añoPublicacion = año;
            }
        }
    }

```

Tarea: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.

```

    */
    public class EjercicioTres {

        public static void main(String[] args) {
            Libro libro = new Libro();

            libro.setAñoPublicacion(-2);
            System.out.println(libro.getAñoPublicacion());

            libro.setAñoPublicacion(1981);
            System.out.println(libro.getAñoPublicacion());
        }
    }

```

```
--- exec:3.1.0:exec (default-cli) @ ejercicioTres ---
0
1981
-----
BUILD SUCCESS
-----
```

4. Gestión de Gallinas en Granja Digital

- a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.

Métodos requeridos: ponerHuevo(), envejecer(), mostrarEstado().

```
* @author lucko
*/
public class Gallina {
    private int idGallina;
    private int edad = 0;
    private int huevosPuestos = 0;

    public void setIdGallina(int id){
        if(id > 0){
            this.idGallina = id;
        }
    }

    public void ponerHuevo(){
        huevosPuestos++;
    }

    public void envejecer(){
        edad++;
    }

    public void mostrarEstado(){
        System.out.println("id: " + idGallina);
        System.out.println("edad: " + edad);
        System.out.println("huevos puestos: " + huevosPuestos);
    }
}
```

Tarea: Crear dos gallinas, simular sus acciones (envejecer y poner huevos), y mostrar su estado.

```

    */
    public class EjercicioCuatro {

        public static void main(String[] args) {
            Gallina gallinaUno = new Gallina();
            Gallina gallinaDos = new Gallina();

            gallinaUno.setIdGallina(1);
            gallinaDos.setIdGallina(2);

            gallinaUno.envenjecer();
            gallinaUno.ponerHuevo();
            gallinaUno.ponerHuevo();
            gallinaUno.ponerHuevo();
            gallinaUno.ponerHuevo();

            gallinaDos.envenjecer();
            gallinaDos.envenjecer();
            gallinaDos.envenjecer();
            gallinaDos.ponerHuevo();
            gallinaDos.ponerHuevo();

            gallinaUno.mostrarEstado();
            gallinaDos.mostrarEstado();
        }
    }
}

```

```

--- exec:3.1.0:exec (default-cli) @ ejercicioCuatro ---
id: 1
edad: 1
huevos puestos: 4
id: 2
edad: 3
huevos puestos: 2
-----
BUILD SUCCESS
-----
Total time: 1.503 s
Finished at: 2025-08-26T16:15:33-03:00
-----

```

5. Simulación de Nave Espacial

Crear una clase NaveEspacial con los atributos: nombre, combustible.

Métodos requeridos: despegar(), avanzar(distancia),

recargarCombustible(cantidad), mostrarEstado().

```

    */
    public class NaveEspacial {
        private String nombre;
        private double combustible;

        public void despegar(String nombre, double combustible){
            this.nombre = nombre;
            this.combustible = combustible;
        }

        public void avanzar(double distancia){
            if(distancia <= combustible){
                combustible = combustible - distancia;
                System.out.println("avanzaste sin problema esta distancia " + distancia);
            } else {
                System.out.println("no tienes la cantidad de combustible necesario para avanzar esta distancia " + distancia);
            }
        }

        public void recargarCombustible(double cantidad){
            combustible = combustible + cantidad;
        }

        public void mostrarEstado(){
            System.out.println("nombre: " + nombre);
            System.out.println("combustible: " + combustible);
        }
    }
}

```

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 EjercicioCinco {  
  
    public static void main(String[] args) {  
        NaveEspacial nave = new NaveEspacial();  
  
        nave.despegar("nave 1", 50);  
        nave.mostrarEstado();  
  
        nave.avanzar(30);  
        nave.mostrarEstado();  
  
        nave.avanzar(30);  
  
        nave.recargarCombustible(40);  
        nave.mostrarEstado();  
  
        nave.avanzar(30);  
        nave.mostrarEstado();  
  
        nave.avanzar(30);  
        nave.mostrarEstado();  
    }  
}
```

```
nombre nave 1  
combustible: 50.0  
avanzaste sin problema esta distancia 30.0  
nombre nave 1  
combustible: 20.0  
no tienes la cantidad de combustible necesario para avanzar esta distancia 30.0  
nombre nave 1  
combustible: 60.0  
avanzaste sin problema esta distancia 30.0  
nombre nave 1  
combustible: 30.0  
avanzaste sin problema esta distancia 30.0  
nombre nave 1  
combustible: 0.0  
-----  
BUILD SUCCESS  
-----
```

LINK AL REPOSITORIO:

<https://github.com/BarreraMariano/UTN-TUPaD-P2>