PROGRAMACIÓN II Trabajo Práctico 3:

1. Registro de Estudiantes

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

```
Public class Estudiante  {
        //atributos
        String = nombre;
        String = apellido;
        String = curso;
        double calificación;
}
Public MostrarInfo  {
        System.out.println(“Nombre: ” + nombre);
        System.out.println(“Apellido: ” + apellido);
        System.out.println(“Curso: ” + curso);
        System.out.println(“Calificacion: ” + calificacion);
}
  public static void main(String[] args) {
      Estudiante estudiante1 = new Estudiante("Ana", "García", "Matemáticas",
8.7);
      estudiante1.mostrarInformacion();
    }
}
  // Método main para probar la clase
    public static void main(String[] args) {
      Estudiante estudiante1 = new Estudiante("Ana", "García", "Matemáticas",
8.7);
      estudiante1.mostrarInformacion();
    }
}
```

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

2. Registro de Mascotas

```
Public class Mascotas  {
          //atributos
          String = nombre;
          String = especie;
          int = edad;
}
public Mascota(String nombre, String especie, int edad) {
        this.nombre = nombre;
        this.especie = especie;
        this.edad = edad;
  }
public void mostrarInfo() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad + " años");
    }

public void cumplirAnios() {
        edad++;
        System.out.println(nombre + " ha cumplido un año más. Ahora tiene " +
edad + " años.");
    }
```

```
    // Método main para probar la clase
    public static void main(String[] args) {
        // Crear una mascota
        Mascota miMascota = new Mascota("Nacha", "Gato", 3);
System.out.println("Información inicial de la mascota:");
        miMascota.mostrarInfo();
   }
}
```

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 Libro {

    // Atributos privados
    private String titulo;
    private String autor;
    private int añoPublicacion;

    public Libro(String titulo, String autor, int añoPublicacion) {
        this.titulo = titulo;
        this.autor = autor;
        setAñoPublicacion(añoPublicacion);
// Usar setter
    }
```

```
    // Getters
    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAñoPublicacion() {
        return añoPublicacion;
    }

    // Setter con validación
    public void setAñoPublicacion(int año) {
        if (año >= 1450 && año <= 2025) {
            this.añoPublicacion = año;
        } else {
            System.out.println("Año inválido: " + año + ". No se modificó el año de
publicación.");
        }
    }

    // Método para mostrar información
    public void mostrarInfo() {
        System.out.println("Título: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Año de publicación: " + añoPublicacion);
```

```
    }

    // Método main para probar
    public static void main(String[] args) {
        Libro libro1 = new Libro("Cien Años de Soledad", "Gabriel García
Márquez", 1967);

        // Intentar modificar con un año inválido
        libro1.setAñoPublicacion(1300);

        // Modificar con un año válido
        libro1.setAñoPublicacion(1985);

        // Mostrar información final
        System.out.println("Información del libro:");
        libro1.mostrarInfo();
    }
}
```

4. Gestión de Gallinas en Granja Digital a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.

```
class Gallina {
    private int idGallina;
    private int edad; // en meses o años según prefieras
    private int huevosPuestos;

    // Constructor
    public Gallina(int idGallina, int edadInicial) {
        this.idGallina = idGallina;
```

```
        this.edad = edadInicial;
        this.huevosPuestos = 0;
    }

    // Método para poner un huevo
    public void ponerHuevo() {
        huevosPuestos++;
        System.out.println("Gallina " + idGallina + " puso un huevo.");
    }

    // Método para envejecer
    public void envejecer() {
        edad++;
        System.out.println("Gallina " + idGallina + " ahora tiene " + edad + "
años.");
    }

    // Mostrar estado de la gallina
    public void mostrarEstado() {
        System.out.println("Gallina " + idGallina + " | Edad: " + edad + " | Huevos
puestos: " + huevosPuestos);
    }
}

// Clase principal
public class GranjaDigital {
    public static void main(String[] args) {
        // Crear dos gallinas
        Gallina gallina1 = new Gallina(1, 2);
        Gallina gallina2 = new Gallina(2, 1);
```

```
        // Simular acciones
        gallina1.envejecer();
        gallina1.ponerHuevo();
        gallina1.ponerHuevo();

        gallina2.envejecer();
        gallina2.ponerHuevo();

        // Mostrar estado final
        System.out.println("\n--- Estado Final de las Gallinas ---");
        gallina1.mostrarEstado();
        gallina2.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.

```
// Clase NaveEspacial
class NaveEspacial {
    private String nombre;
```

```
    private int combustible;
    private final int MAX_COMBUSTIBLE = 100;

    // Constructor
    public NaveEspacial(String nombre, int combustibleInicial) {
        this.nombre = nombre;
        if (combustibleInicial > MAX_COMBUSTIBLE) {
            this.combustible = MAX_COMBUSTIBLE;
        } else {
            this.combustible = combustibleInicial;
        }
    }

    // Método despegar
    public void despegar() {
        if (combustible >= 10) {
            combustible -= 10;
            System.out.println(nombre + " ha despegado. (Consumió 10 de
combustible)");
        } else {
            System.out.println(nombre + " no tiene suficiente combustible para
despegar.");
        }
    }

    // Método avanzar
    public void avanzar(int distancia) {
        int consumo = distancia * 2; // por ejemplo, 2 unidades de combustible por
cada unidad de distancia
        if (combustible >= consumo) {
```

```
            combustible -= consumo;
            System.out.println(nombre + " avanzó " + distancia + " km. (Consumió " + consumo + " de combustible)");
        } else {
            System.out.println(nombre + " no tiene suficiente combustible para avanzar " + distancia + " km.");
        }
    }

    // Método recargar combustible
    public void recargarCombustible(int cantidad) {
        if (combustible + cantidad > MAX_COMBUSTIBLE) {
            combustible = MAX_COMBUSTIBLE;
            System.out.println(nombre + " se recargó al máximo (" + MAX_COMBUSTIBLE + ").");
        } else {
            combustible += cantidad;
            System.out.println(nombre + " recargó " + cantidad + " unidades de combustible.");
        }
    }

    // Mostrar estado
    public void mostrarEstado() {
        System.out.println("Nave: " + nombre + " | Combustible: " + combustible + "/" + MAX_COMBUSTIBLE);
    }
}

// Clase principal
public class SimulacionNave {
```

```
    public static void main(String[] args) {
        // Crear la nave con 50 unidades de combustible
        NaveEspacial nave = new NaveEspacial("Explorer-1", 50);

        // Intentar avanzar sin recargar
        nave.despegar();
        nave.avanzar(30); // debería fallar por falta de combustible

        // Recargar y luego avanzar correctamente
        nave.recargarCombustible(40);
        nave.avanzar(20);

        // Mostrar estado final
        System.out.println("Estado Final de la Nave ");
        nave.mostrarEstado();
    }
}
```