Ejercicio 1

```
/*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.

*/

package tp3_introduccionpoo;

// Se creea Clase Estudiante con los atributos: nombre, apellido, curso,
//calificación y 3 metodos

class Estudiante {
    String nombre;
    String apellido;
    String curso;
    int calificacion; // 0 a 10

// Muestra por pantalla los datos del estudiante
    void mostrarInfo() {
        System.out.println("Estudiante: " + nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificacion: " + calificacion);
    }
// Suma puntos a la calificacion
    void subirCalificacion(int puntos) {
        calificacion += puntos;
    }
// Resta puntos a la calificacion
    void bajarCalificacion(int puntos) {
        calificacion -= puntos;
    }
}

//////////////////////Comienza main para ejecutar ////////////////////////

public class E1_RegistroDeEstudiantes {

    public static void main(String[] args) {
     Estudiante e = new Estudiante();//Crear estudiante en memoria
     // Cargar valores en los atributos del objeto 'e'
        e.nombre = "Lionel";
        e.apellido = "Messi";
        e.curso = "Programacion 2";
        e.calificacion = 7;
//Mostrar estado inicial
        System.out.println(" Estado inicial ");
        e.mostrarInfo();
//Subir la calificacion en +2 y volver a mostrar
        System.out.println("\n Subir calificacion (+2) ");
```

```
        e.subirCalificacion(2);
        e.mostrarInfo();
//Bajar la calificacion en -3 y volver a mostrar
        System.out.println("\n Bajar calificacion (-3) ");
        e.bajarCalificacion(3);
        e.mostrarInfo();
    }
}
```

Ejercicio 2

```
/*
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.
 */
package tp3_introduccionpoo;

// Se crear una clase Mascota con los atributos: nombre, especie, edad, y 2 metodos.
class Mascota {
    // Atributos
    String nombre;
    String especie;
    int    edad;

    // Muestra por pantalla los datos de la mascota
    void mostrarInfo() {
        System.out.println("Mascota: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad + " anio ");
    }

    // Suma 1 anio a la edad de la mascota
    void cumplirAnios() {
        edad = edad + 1;
        System.out.println(nombre + " cumplio anios. Nueva edad: " + edad);
    }

    // Setear edad con validacion
    void setEdad(int nuevaEdad) {
        if (nuevaEdad < 0) {
            System.out.println("La edad no puede ser negativa. Se deja en 0.");
            edad = 0;
        } else {
            edad = nuevaEdad;
        }
    }
}

//////////////////////Comienza main para ejecutar ////////////////////////
public class E2_RegistroDeMascotas {

    public static void main(String[] args) {
```

```
        // Crear mascota en memoria
        Mascota m = new Mascota();

        //Cargar valores en los atributos en el objeto m
        m.nombre  = "Barti";
        m.especie = "Perro";
        m.setEdad(5);     // uso el setter con validacion

        //Mostrar estado inicial
        System.out.println(" Estado inicial ");
        m.mostrarInfo();

        // Cumple años
        System.out.println("\n Cumple 1 anio ");
        m.cumplirAnios();

        // Mostrar estado final
        System.out.println("\n Estado final ");
        m.mostrarInfo();


    }
}
```

Ejercicio 3

```
package tp3_introduccionpoo;

/*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.*/


//Clase encapsula su estado y expone metodos de acceso
class Libro {
//Atributo privados
private String titulo;
private String autor;
private int anioPublicacion;

//------------------Metodos----------------------------------------

// Setter para anioPublicacion, con validacion de año
public void setAnioPublicacion(int anio) {
    int MIN = 1700;
    int MAX = 2025;

    if (anio < MIN || anio > MAX) {
        System.out.println("Anio invalido: " + anio +
```

```
            ". Debe estar entre " + MIN + " y " + MAX);
        return; // salimos sin cambiar el valor
    }
    this.anioPublicacion = anio; // valido: se guarda
}
//---------------------------------------------------------------------
// Getters lectura de atributos
public String getTitulo() {
    return titulo;
}

public String getAutor() {
    return autor;
}

public int getAnioPublicacion() {
    return anioPublicacion;
}

//---------------------------------------------------------------------
//Inicializa el objeto al momento de crearlo con new
public Libro(String titulo, String autor, int anioPublicacion) {
        this.titulo = titulo; // "this" se refiere al atributo de la clase
        this.autor = autor;
        this.anioPublicacion = anioPublicacion;
}

//---------------------------------------------------------------------
// Metodo mostrarInfo() para ver el estado actual
public void mostrarInfo() {
        System.out.println("Titulo: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Anio Publicacion: " + anioPublicacion + " anio");

    }

}

////////////////////////// Comienza "main"" para ejecutar//////////////////////////

//---------------------------------------------------------------------------
public class E3_ClaseLibro {

public static void main(String[] args){

//Crear el libro con datos iniciales (año valido)
Libro t= new Libro("Gladiador","Artur", 2010);
System.out.println( "Estado inicial ");
t.mostrarInfo();

//---------------------------------------------------------------------
// Intento de año Invalido
System.out.println("\nIntento con anio INVALIDO ");
t.setAnioPublicacion(1200);   // fuera de rango no debe cambiar

//Intento de año Valido
```

```
System.out.println("\nIntento con anio VALIDO ");
t.setAnioPublicacion(2015);   // dentro de rango debe cambiar

//Se muestra el estado final
System.out.println("\nEstado final ");
t.mostrarInfo();

//-------------------------------------------------------------
//Lectura con getters para verificar encapsulado
System.out.println("\n Lectura con getters ");
System.out.println("Titulo  " + t.getTitulo());
System.out.println("Autor   " + t.getAutor());
System.out.println("Anio    " + t.getAnioPublicacion());
}


}
```

Ejercicio 4

```
package tp3_introduccionpoo;

/**
 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.
 */

//Clase estado y metodos
class Gallina{
//Atributos
int idGallina;
int edad;
int huevosPuestos;
//-------------------------------------------------------------
// Metodo ponerHuevos Suma 1 huevo a la gallina
void ponerHuevos (){

  huevosPuestos +=1;
}
//-------------------------------------------------------------
// Metodo envejecer Suma 1 año a la edad de la gallina
void envejecer (){

    edad +=1;
}
//-----------------------------------------------------------------
// Metodo mostrarEstado() para ver el estado actual
 void mostrarEstado() {
        System.out.println("\nId Gallina: " + idGallina);
        System.out.println("Edad Gallina: " + edad);
        System.out.println("Cant Huevos Puestos: " + huevosPuestos);
    }
}
```

```
////////////////////////// Comienza "main"" para ejecutar//////////////////////////
public class E4_GranjaDigital {

public static void main(String[] args){

// Se crean dos gallinas y se asignan valores iniciales
        Gallina g1 = new Gallina();
        g1.idGallina = 1;
        g1.edad = 1;
        g1.huevosPuestos = 0;

        Gallina g2 = new Gallina();
        g2.idGallina = 2;
        g2.edad = 2;
        g2.huevosPuestos = 3;

//-------------------------------------------------------------
//  Estado inicial
        System.out.println("\nEstado inicial ");
        g1.mostrarEstado();
        g2.mostrarEstado();

//-----------------------------------------------------
    // Simulacion envejecer y poner huevos
        System.out.println("\nSimulacion ");
        g1.envejecer();
        g1.ponerHuevos();
        g1.ponerHuevos();          // g1 pone 2 huevos

        g2.envejecer();
        g2.ponerHuevos();          // g2 pone 1 huevo

//-----------------------------------------------------------
    // Estado final
        System.out.println("\nEstado final ");
        g1.mostrarEstado();
        g2.mostrarEstado();
  }
}
```

Ejercicio 5

```
package tp3_introduccionpoo;

/*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.*/

class NaveEspacial{
```

```
//Atributos
    private String nombre;
    private int combustible;



// Se toma los parámetros y se asignan a los atributos de la clase
    public NaveEspacial(String nombre, int combustible) {
        this.nombre = nombre;
        this.combustible = combustible;

    }

//Metodos:  despegar(), avanzar(distancia), recargarCombustible(cantidad),
//mostrarEstado().
//-------------------------------------------

public void despegar (){
    if (combustible > 0) {
        System.out.println("Despegando la nave " + nombre);
    }
    else {
        System.out.println("No hay suficiente combustible para despegar.");
    }
}
//-----------------------------------------------------
public void avanzar (int distancia){

     int consumo = distancia / 10;  // Definimos que por cada 10 de distancia se gasta 1 de combustible

    if (combustible >= consumo) {
        combustible -= consumo;  // Descontamos el combustible
        System.out.println("La nave avanzo " + distancia + " unidades.");
    }
    else {
        System.out.println("No hay suficiente combustible para avanzar esa distancia.");
    }
 }
//-------------------------------------------------------
// Metodo recargarCombustible ()
public void recargarCombustible (int cantidad){
     if (cantidad > 0 && combustible + cantidad <= 100) { // Si es una cantidad válida y no supera el límite
        combustible += cantidad;
        System.out.println("Recargando " + cantidad + " unidades de combustible.");
       }
    else {
        System.out.println("Cantidad inválida o exceso de combustible.");
    }
  }

//-------------------------------------------------
// Metodo mostrarEstado() para ver el estado
 void mostrarEstado() {
       // System.out.println("\ndespegar: " + );
        System.out.println("Nombre: " + nombre);
```

```
        System.out.println("Cant de combustible: " + combustible);
    }
}

////////////////////// Main ejecucion del programa//////////////////

public class E5_NaveEspacial {

    public static void main(String[] args) {

        NaveEspacial nave = new NaveEspacial ("Apolo 11", 50);

 //-----------Estado inicial----------------
    //Estado inicial
        System.out.println("\nEstado inicial ");
        nave.mostrarEstado();

    // Despegar
        System.out.println("\nEstado ok despegar ");
        nave.despegar();


    // Intentar avanzar sin recargar
        System.out.println("\nIntentando avanzar sin recargar");
        nave.avanzar(200); // No debería poder avanzar con 50 unidades

    // Recargar combustible
        System.out.println("\nRecargando combustible");
        nave.recargarCombustible(30);  // Recargamos 30 unidades
        nave.avanzar(200);             // Ahora debería poder avanzar

    // Mostrar estado final
        System.out.println("\nEstado final");
        nave.mostrarEstado();

  }
}
```