PROGRAMACIÓN II

Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Alumno: Mauro Gaspar
Comisión: 4

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.

```java
public class ejerciciol {

    // Atributos privados
    private String nombre;
    private String apellido;
    private String curso;
    private double calificacion;

    public ejerciciol(String nombre, String apellido, String curso,double calificacion) {

        this.nombre = nombre;
        this.apellido = apellido;
        this.curso=curso;
        this.calificacion = calificacion;

    }

    // Método para mostrar la infomcación del estudiante
    public void mostrarInfo() {

        System.out.println("** Información del Estudiante **");
        System.out.println("Nombre: " + nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificación: " + calificacion);
        System.out.println("---------------");
    }

    // Método para subir calificación
    public void subirCalificacion(double puntos) {
        this.calificacion += puntos;
        System.out.println("Calificación aumentada en " + puntos + " puntos.");
    }

    // Método para ahar calificación
    public void bajarCalificacion(double puntos) {
        this.calificacion -= puntos;
        if (this.calificacion < 0) {
            this.calificacion = 0; // Evita calificación negativa
        }
        System.out.println("Calificación disminuida en " + puntos + " puntos.");
    }
```

```
TP3 (run) ×    NetBeansProjects - D:\Mauro\Documents\NetBeansProjects ×
run:
Ingrese el nombre del estudiante: Antonio
Ingrese el apellido del estudiante: Mendez
Ingrese el curso: ´Matematica
Ingrese la calificación inicial: 5
** Información del Estudiante **
Nombre: Antonio Mendez
Curso: □Matematica
Calificación: 5.0
--------------
¿Cuántos puntos desea aumentar a la calificación? 3
Calificación aumentada en 3.0 puntos.
** Información del Estudiante **
Nombre: Antonio Mendez
Curso: □Matematica
Calificación: 8.0
--------------
¿Cuántos puntos desea disminuir a la calificación? 2
Calificación disminuida en 2.0 puntos.
** Información del Estudiante **
Nombre: Antonio Mendez
Curso: □Matematica
Calificación: 6.0
--------------
BUILD SUCCESSFUL (total time: 48 seconds)
```

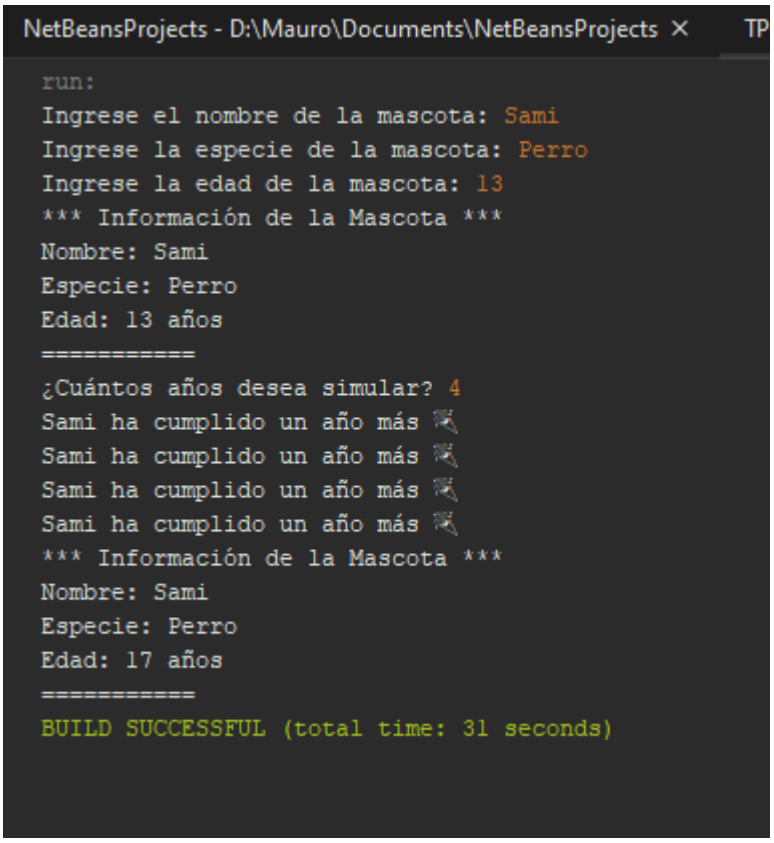
2. Registro de Mascotas

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.

```java
public class Mascota {

    // Atributos privados
    private String nombre;
    private String especie;
    private int edad;

    // Constructor recibe los valores cuando se crea un objeto
    public Mascota(String nombre, String especie, int edad) {
        this.nombre = nombre;
        this.especie = especie;
        this.edad = edad;
    }

    // Método para mostrar la información de la mascota
    public void mostrarInfo() {
        System.out.println("*** Información de la Mascota ***");
        System.out.println("Nombre: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad + " años");
        System.out.println("============");
    }

    // Método para cumplir años, aumenta la edad en 1
    public void cumplirAnios() {
        this.edad++;
        System.out.println(nombre + " ha cumplido un año más 🎉");
    }
}
```



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.

```java
public class Libro {

    // Atributos privados, que se accederán a ellos mediante getters y setters.
    private String titulo;
    private String autor;
    private int anioPublicacion;

    public Libro(String titulo, String autor, int anioPublicacion) {
        this.titulo = titulo;
        this.autor = autor;
        setAnioPublicacion(anioPublicacion); // usamos el setter para validar
    }

    // Getters para todos los atributos (accesi de lectura)
    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    // Setter con validación para el año de publicación (acceso de escritura seguro)
    public void setAnioPublicacion(int anioPublicacion) {
        if (anioPublicacion > 0 && anioPublicacion <= 2025) {
            this.anioPublicacion = anioPublicacion;
        } else {
            System.out.println("Año de publicación inválido: " + anioPublicacion);
        }
    }

    // Método para mostrar la información
    public void mostrarInfo() {
        System.out.println("*** Información del Libro ***");
        System.out.println("Título: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Año de publicación: " + anioPublicacion);
        System.out.println("==============");
    }
```

```java
public class MainLibro {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Pedir datos del libro
        System.out.print("Ingrese el título del libro: ");
        String titulo = sc.nextLine();

        System.out.print("Ingrese el autor del libro: ");
        String autor = sc.nextLine();

        System.out.print("Ingrese el año de publicación: ");
        int anio = sc.nextInt();

        // Crear objeto Libro con los datos ingresados
        Libro libro1 = new Libro(titulo, autor, anio);

        // Mostrar información inicial
        libro1.mostrarInfo();

        // Asignar un año inválido
        System.out.print("Ingrese un nuevo año de publicación (inválido para probar): ");
        int anioInvalido = sc.nextInt();
        libro1.setAnioPublicacion(anioInvalido);
        libro1.mostrarInfo();

        // Asignar un año válido
        System.out.print("Ingrese un nuevo año de publicación (válido): ");
        int anioValido = sc.nextInt();
        libro1.setAnioPublicacion(anioValido);
        libro1.mostrarInfo();

        sc.close();
    }
}
```

```
run:
Ingrese el título del libro: La cura en un minuto
Ingrese el autor del libro: Madison Cavanaugh
Ingrese el año de publicación: 2010
*** Información del Libro ***
Título: La cura en un minuto
Autor: Madison Cavanaugh
Año de publicación: 2010
==============
Ingrese un nuevo año de publicación (inválido para probar): 1700
Año de publicación inválido: 1700
*** Información del Libro ***
Título: La cura en un minuto
Autor: Madison Cavanaugh
Año de publicación: 2010
==============
Ingrese un nuevo año de publicación (válido): 2020
*** Información del Libro ***
Título: La cura en un minuto
Autor: Madison Cavanaugh
Año de publicación: 2020
==============
BUILD SUCCESSFUL (total time: 3 minutes 35 seconds)
```

4. Gestión de Gallinas en Granja Digital

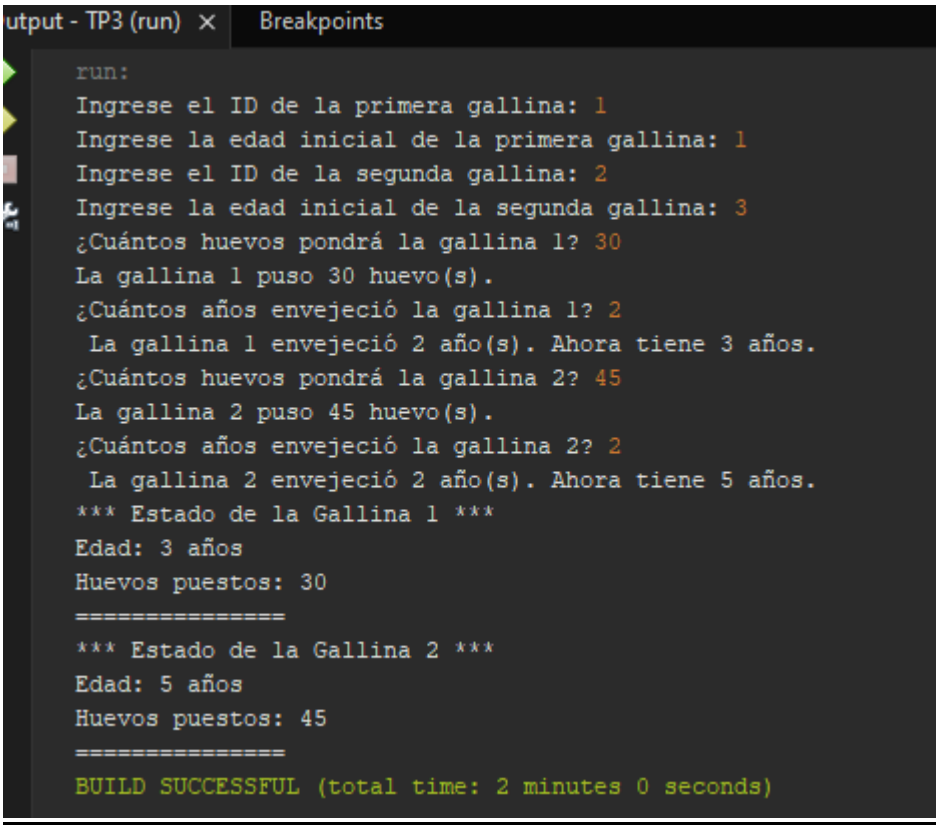
a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.

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

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

```
 * @author Mauro
 */
public class Gallina {
    // Atributos
    private int idGallina;
    private int edad;
    private int huevosPuestos;

    // COnstructor
    public Gallina(int idGallina, int edad) {
        this.idGallina = idGallina;
        this.edad = edad;
        this.huevosPuestos = 0;
    }

    // Método para poner un huevo
    public void ponerHuevo(int cantidad) {
        this.huevosPuestos += cantidad;
        System.out.println("La gallina " + idGallina + " puso " + cantidad + " huevo(s).");
    }

    // Método para envejecer
    public void envejecer(int anios) {
        this.edad += anios;
        System.out.println(" La gallina " + idGallina + " envejeció " + anios + " año(s). Ahora

    }

    // Método para mostrar el estado
    public void mostrarEstado() {
        System.out.println("*** Estado de la Gallina " + idGallina + " ***");
        System.out.println("Edad: " + edad + " años");
        System.out.println("Huevos puestos: " + huevosPuestos);
        System.out.println("==============");
    }
}
```

```java
// Crear primera gallina
System.out.print("Ingrese el ID de la primera gallina: ");
int id1 = sc.nextInt();
System.out.print("Ingrese la edad inicial de la primera gallina: ");
int edad1 = sc.nextInt();
Gallina g1 = new Gallina(id1, edad1);

// Crear segunda gallina
System.out.print("Ingrese el ID de la segunda gallina: ");
int id2 = sc.nextInt();
System.out.print("Ingrese la edad inicial de la segunda gallina: ");
int edad2 = sc.nextInt();
Gallina g2 = new Gallina(id2, edad2);

// Acciones para la primera gallina
System.out.print("¿Cuántos huevos pondrá la gallina " + id1 + "? ");
int huevos1 = sc.nextInt();
g1.ponerHuevo(huevos1);

System.out.print("¿Cuántos años envejeció la gallina " + id1 + "? ");
int anios1 = sc.nextInt();
g1.envejecer(anios1);

// Acciones para la segunda gallina
System.out.print("¿Cuántos huevos pondrá la gallina " + id2 + "? ");
int huevos2 = sc.nextInt();
g2.ponerHuevo(huevos2);

System.out.print("¿Cuántos años envejeció la gallina " + id2 + "? ");
int anios2 = sc.nextInt();
g2.envejecer(anios2);

// Mostrar estado final
g1.mostrarEstado();
g2.mostrarEstado();

sc.close();
```



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.

```java
public class NaveEspacial {

    private String nombre;
    private double combustible;
    private final double COMBUSTIBLE_MAXIMO = 100.0;

    // Constructor para inicializar la nave
    public NaveEspacial(String nombre, double combustibleInicial) {
        this.nombre = nombre;
        // Se valida el combustible inicial para que no exceda el máximo
        if (combustibleInicial > COMBUSTIBLE_MAXIMO) {
            this.combustible = COMBUSTIBLE_MAXIMO;
            System.out.println("Atención! El combustible inicial excede el límite. Se
        } else {
            this.combustible = combustibleInicial;
        }
    }

    public double getCombustible() {
    return combustible;
    }

    // Método para simular el despegue
    public void despegar() {
        double consumoDespegue = 10.0;
        if (this.combustible >= consumoDespegue) {
            this.combustible -= consumoDespegue;
            System.out.println(this.nombre + " ha despegado. Combustible restante: " +
        } else {
            System.out.println("Error: No hay suficiente combustible para despegar.");
        }
    }

    // Método para avanzar, consume 1 unidad de combustible por unidad de distancia
    public void avanzar(double distancia) {
        double consumoNecesario = distancia;
        if (this.combustible >= consumoNecesario) {
            this.combustible -= consumoNecesario;
            System.out.println(this.nombre + " avanza " + distancia + " unidades. Comb
        } else {
            System.out.println("Error: Combustible insuficiente para avanzar " + dista
        }
    }
```

```java
package tp3;
import java.util.Scanner;

/**
 *
 * @author Mauro
 */
public class MainNave {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("*** Simulación de Nave Espacial ***");

        // Crear una nave con datos ingresados por el usuario
        System.out.print("Ingreso el nombre de la nave: ");
        String nombreNave = sc.nextLine();

        double combustibleInicial = 50.0;
        NaveEspacial miNave = new NaveEspacial(nombreNave, combustibleInicial);
        miNave.mostrarEstado();

        // Intentar avanzar sin recargar
        System.out.println("\n--- Intentar avanzar sin recargar ---");
        System.out.print("Ingresa la distancia a intentar avanzar (más de " + miNave.
        double distanciaIntentada = sc.nextDouble();
        miNave.avanzar(distanciaIntentada);
        miNave.mostrarEstado();

        System.out.print("Ingrese la distancia a avanzar después de la recarga: ");
        double distanciaFinal = sc.nextDouble();
        miNave.avanzar(distanciaFinal);

        // Mostrar el estado final
        miNave.mostrarEstado();

        sc.close();
    }

}
```

```
run:
*** Simulación de Nave Espacial ***
Ingreso el nombre de la nave: Maurix

*** Estado de la Nave Maurix ***
Combustible: 50.0 / 100.0
================

--- Intentar avanzar sin recargar ---
Ingresa la distancia a intentar avanzar (más de 50.0 unidades): 70
Error: Combustible insuficiente para avanzar 70.0 unidades.

*** Estado de la Nave Maurix ***
Combustible: 50.0 / 100.0
================
Ingrese la distancia a avanzar después de la recarga: 20
Maurix avanza 20.0 unidades. Combustible restante: 30.0

*** Estado de la Nave Maurix ***
Combustible: 30.0 / 100.0
================
BUILD SUCCESSFUL (total time: 22 seconds)
```