PROGRAMACIÓN II

Trabajo Práctico 3: Introducción Programación Orientada a Objetos

Alumno: Mauro Zavatti

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.

```
public class RegistroEstudianteMain {


  public static void main(String[] args) {
    RegistroEstudiantes alum = new RegistroEstudiantes();


    alum.setNombre("Mauro");
    alum.setApellido("Zavatti");
    alum.setCurso("Programacion");
    alum.setCalificacion(7.5);


    alum.mostrarInfo();


    alum.subirCalificacion(1.5);


    alum.bajarCalificacion(3);


    System.out.println("\nInformacion final del estudiante:");
    alum.mostrarInfo();
  }
  }
```

```
public class RegistroEstudiantes {
    private String nombre;
    private String apellido;
    private String curso;
    private double calificacion;


   // Getter y Setter para nombre
   public String getNombre() {
      return nombre;
   }
   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   // Getter y Setter para apellido
   public String getApellido() {
      return apellido;
   }
   public void setApellido(String apellido) {
      this.apellido = apellido;
   }

   // Getter y Setter para curso
   public String getCurso() {
      return curso;
   }
   public void setCurso(String curso) {
      this.curso = curso;
   }
```

```
// Getter y Setter para calificación
public double getCalificacion() {
   return calificacion;
}
public void setCalificacion(double calificacion) {
   this.calificacion = calificacion;
}
public void mostrarInfo() {
   System.out.println("Nombre: " + nombre +
         "\nApellido: " + apellido +
         "\nCurso: " + curso +
         "\nCalificacion: " + calificacion);
}

public void subirCalificacion(double puntos){
   calificacion += puntos;
   if(calificacion == 10){
   calificacion = 10;
   }
   System.out.println("La nueva calificacion es: " + calificacion);
}

public void bajarCalificacion(double puntos) {
   calificacion -= puntos;
   if (calificacion < 0) {
      calificacion = 0;
   }
   System.out.println("La calificacion ha bajado. Nueva calificacion: " + calificacion);
}
```

```
}
```

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.

```
import java.util.Scanner;

public class RegistroMascotasMain {
  public static void main(String[] args) {
    RegistroMascotas mascota = new RegistroMascotas();

    mascota.setNombre("Polo");
    mascota.setEspecie("Perro");
    mascota.setEdad(1);

    mascota.mostrarInfo();

    int tiempo;
    Scanner input = new Scanner(System.in);
    System.out.print("Cuanto tiempo paso?(anios): ");
    tiempo = Integer.parseInt(input.nextLine());

    mascota.cumplirAnios(tiempo);

    System.out.println("\nInformación actualizada:");
    mascota.mostrarInfo();
  }
```

```
}
public class RegistroMascotas {
    private String nombre;
    private String especie;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEspecie() {
        return especie;
    }

    public void setEspecie(String especie) {
        this.especie = especie;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
```

```
    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre +
                "\nEspecie: " + especie +
                "\nEdad: " + edad
                );
    }

    public void cumplirAnios(int tiempo){
        edad += tiempo;
        System.out.println(nombre + " cumplió " + tiempo + " años.");
    }
}
```

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

        libro.setTitulo("El Quijote");
        libro.setAutor("Miguel de Cervantes");
        libro.setAnioPublicacion(1605);

        libro.mostrarInfo();
```

```
        // le paso un año invalido
        libro.setAnioPublicacion(1200);

        //le paso un año valido
        libro.setAnioPublicacion(2005);

        System.out.println("\nInformación final del libro:");
        libro.mostrarInfo();
    }
}
public class Libro {
    private String titulo;
    private String autor;
    private int anioPublicacion;

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    public void setTitulo(String titulo) {
```

```
        this.titulo = titulo;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }
    public void setAnioPublicacion(int anioPublicacion) {
        if (anioPublicacion >= 1500 && anioPublicacion <= 2025) {
        this.anioPublicacion = anioPublicacion;
    } else {
            System.out.println("Año inválido: " + anioPublicacion + ". No se actualizó.");
        }
    }

    public void mostrarInfo() {
        System.out.println("Título: " + titulo);
        System.out.println("Autor: " + autor);
        System.out.println("Año de publicación: " + anioPublicacion);
    }
}
```

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.

```
public class GallinaMain {
    public static void main(String[] args) {
        Gallina gallina1 = new Gallina();
        Gallina gallina2 =  new Gallina();
```

```
    //Lomejor aqui seria crear un constructor en la Clase gallina, pero por le momento lo dejo asi
    /*
      public Gallina(int idGallina, int edad, int huevosPuestos) {
          this.idGallina = idGallina;
          this.edad = edad;
          this.huevosPuestos = huevosPuestos;
      }
    */

    gallina1.setIdGallina(1);
    gallina1.setEdad(4);
    gallina1.setHuevosPuestos(20);

    gallina2.setIdGallina(2);
    gallina2.setEdad(5);
    gallina2.setHuevosPuestos(15);


    gallina1.mostrarEstado();
    gallina2.mostrarEstado();

    System.out.println("----- Simulación -----");


    gallina1.envejecer(5);
    gallina1.ponerHuevos(6);

    gallina2.envejecer(7);
```

```
        gallina2.ponerHuevos(3);

        gallina1.mostrarEstado();
        gallina2.mostrarEstado();

    }
}
public class Gallina {
    private int idGallina;
    private int edad;
    private int huevosPuestos;

    public int getIdGallina() {
        return idGallina;
    }

    public void setIdGallina(int idGallina) {
        this.idGallina = idGallina;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }

    public int getHuevosPuestos() {
        return huevosPuestos;
```

```
    }

    public void setHuevosPuestos(int huevosPuestos) {
        this.huevosPuestos = huevosPuestos;
    }


    //Metodos

    public void ponerHuevos(int huevos){
        huevosPuestos += huevos;
    }

    public void envejecer(int tiempo){
        edad += tiempo;
    }

    public void mostrarEstado(){
        System.out.println("La Gallina " + idGallina + " tiene " + edad + " anios y puso " +
huevosPuestos + " huevos");
    }
}
```

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 NaveEspacialMain {
     public static void main(String[] args) {
       NaveEspacial nave = new NaveEspacial();

       nave.setNombre("Endurance");
       nave.setCombustible(50);

       nave.mostrarEstado();

       nave.avanzar(30);
       nave.recargarCombustible(40);
       nave.avanzar(30);
       nave.mostrarEstado();
   }
}

public class NaveEspacial {
   private String nombre;
   private int combustible;
   private final int CAPACIDAD_MAXIMA = 100; // límite máximo de combustible

   /*
   // Constructor
   public NaveEspacial(String nombre, int combustible) {
      this.nombre = nombre;
      this.combustible = combustible;
   }*/

   public String getNombre() {
```

```
    return nombre;
  }

  public void setNombre(String nombre) {
    this.nombre = nombre;
  }

  public int getCombustible() {
    return combustible;
  }

  public void setCombustible(int combustible) {
    this.combustible = combustible;
  }




  // Métodos
  public void despegar() {
    if (combustible >= 10) {
      combustible -= 10;
      System.out.println(nombre + " despegó con éxito. Combustible restante: " +
combustible);
    } else {
      System.out.println("No hay suficiente combustible para despegar.");
    }
  }

  public void avanzar(int distancia) {
```

```
    int consumo = distancia * 2; // ejemplo: 2 unidades de combustible por unidad de distancia
    if (combustible >= consumo) {
      combustible -= consumo;
      System.out.println(nombre + " avanzo " + distancia + " km. Combustible restante: " + combustible);
    } else {
      System.out.println("Combustible insuficiente para avanzar " + distancia + " km.");
    }
  }

  public void recargarCombustible(int cantidad) {
    if (combustible + cantidad > CAPACIDAD_MAXIMA) {
      combustible = CAPACIDAD_MAXIMA;
      System.out.println("Se recargó al máximo. Combustible actual: " + combustible);
    } else {
      combustible += cantidad;
      System.out.println("Se recargaron " + cantidad + " unidades. Combustible actual: " + combustible);
    }
  }

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