PROGRAMACIÓN II

Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Alumno: Angelelli, Rodrigo Martin

Comisión N.7

Link al repositorio: https://github.com/roangelelli/Programacion2_TUPAD_Angelelli_Rodrigo

Caso Práctico

Desarrollar en Java los siguientes ejercicios aplicando los conceptos de programación orientada a objetos:

1. **Registro de Estudiantes**: Crear una clase Estudiante con los atributos: nombre, apellido, curso, calificación.
    a. **Métodos requeridos:** mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos).
    b. **Tarea**: Instanciar a un estudiante, mostrar su información, aumentar y disminuir calificaciones.

```java
package tp.introduccion.a.poo.angelelli.rodrigo;
public class Estudiantes {
    public String nombre;
    public String apellido;
    public String curso;
    public double calificacion;

    public void mostrarInfo(){
        System.out.println(
            "Datos del estudiante: \n"
            + "Nombre: " + nombre + "\n"
            + "Apellido: " + apellido + "\n"
            + "Curso: " + curso + "\n"
            + "Calificación: " + calificacion);
    }

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

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

```java
// EJERCICIO 1 - ESTUDIANTES
System.out.println("============== EJERCICIO 1 ==============");

Estudiantes estudiante1 = new Estudiantes();
estudiante1.nombre = "Lautaro";
estudiante1.apellido = "Perez";
estudiante1.curso = "7 - A";
estudiante1.calificacion = 8;

estudiante1.mostrarInfo();

System.out.println("=".repeat(40));

estudiante1.subirCalificacion(2);
estudiante1.mostrarInfo();

System.out.println("=".repeat(40));

estudiante1.bajarCalificacion(3.5);
estudiante1.mostrarInfo();
```

```
run:
============== EJERCICIO 1 ==============
Datos del estudiante:
Nombre: Lautaro
Apellido: Perez
Curso: 7 - A
Calificación: 8.0
========================================
Datos del estudiante:
Nombre: Lautaro
Apellido: Perez
Curso: 7 - A
Calificación: 10.0
========================================
Datos del estudiante:
Nombre: Lautaro
Apellido: Perez
Curso: 7 - A
Calificación: 6.5
```

2. **Registro de Mascotas:** Crear una clase Mascota con los atributos: nombre, especie, edad.
    a. **Métodos** requeridos: mostrarInfo(), cumplirAnios().
    b. **Tarea**: Crear una mascota, mostrar su información, simular el paso del tiempo y verificar los cambios.

```java
package tp.introduccion.a.poo.angelelli.rodrigo;
public class Mascotas {
    public String nombre;
    public String especie;
    public int edad;

    public void mostrarInfo(){
        System.out.println(
            "Datos de la mascota: \n"
            + "Nombre: " + nombre + "\n"
            + "Especie: " + especie + "\n"
            + "Edad: " + edad);
    }

    void cumplirAnios (int anios){
    edad = edad + anios;
        System.out.println(nombre + " ha cumplido años. Ahora tiene " +edad + " años.");
    }

}
```

```java
// EJERCICIO 2 - MASCOTAS
System.out.println("============== EJERCICIO 2 ==============");

Mascotas mascota1 = new Mascotas();
mascota1.nombre = "Pancho";
mascota1.especie = "Dachshund";
mascota1.edad = 1;
mascota1.mostrarInfo();

System.out.println("=".repeat(40));

mascota1.cumplirAnios(4);
mascota1.mostrarInfo();
```

```
============== EJERCICIO 2 ==============
Datos de la mascota:
Nombre: Pancho
Especie: Dachshund
Edad: 1
========================================
Pancho ha cumplido años. Ahora tiene 5 años.
Datos de la mascota:
Nombre: Pancho
Especie: Dachshund
Edad: 5
```

3. **Encapsulamiento con la Clase Libro:** Crear una clase Libro con atributos privados: titulo, autor, añoPublicacion.
    a. **Métodos requeridos**: Getters para todos los atributos. Setter con validación para añoPublicacion.
    b. **Tarea**: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.

```java
package tp.introduccion.a.poo.angelelli.rodrigo;
import java.time.Year;

public class Libro {
    private String titulo;
    private String autor;
    private int anioPublicacion;

    public String getTitulo() {...3 lines }

    public void setTitulo(String titulo) {...3 lines }

    public String getAutor() {...3 lines }

    public void setAutor(String autor) {...3 lines }

    public int getAnioPublicacion() {...3 lines }

    public void setAnioPublicacion(int nuevoAnio) {
        int anioActual = Year.now().getValue();
        if (nuevoAnio >= 1450 && nuevoAnio<= anioActual){  //Fecha de origen de la imprenta moderna aprox.
            this.anioPublicacion = nuevoAnio;}
        else {
            System.out.println("El año de publicación debe ser entre 1450 y " +anioActual );}
    }

    public void mostrarInfo() {
        System.out.println("INFORMACIÒN SOBRE SU LIBRO \n"
                           + "Titulo: " + titulo + "\n"
                           + "Autor: " + autor + "\n"
                           + "Año de Publicación: " +anioPublicacion);}
```

```java
// EJERCICIO 3 - LIBRO
System.out.println("============== EJERCICIO 3 ==============");

Libro miLibro = new Libro ();
miLibro.setTitulo("Mi libro favorito");
miLibro.setAutor("Mi autor favorito");
miLibro.setAnioPublicacion(1980);
miLibro.mostrarInfo();

System.out.println("=".repeat(40));
miLibro.setAnioPublicacion(1300); //Valor invalido en el pasado
miLibro.setAnioPublicacion(2027); //Valor invalido en el futuro
miLibro.mostrarInfo();

System.out.println("=".repeat(40));
miLibro.setAnioPublicacion(2000); //Valor valido
miLibro.mostrarInfo();
```

```
============== EJERCICIO 3 ==============
INFORMACIÒN SOBRE SU LIBRO
Titulo: Mi libro favorito
Autor: Mi autor favorito
Año de Publicación: 1980
========================================
El año de publicación debe ser entre 1450 y 2025
El año de publicación debe ser entre 1450 y 2025
INFORMACIÒN SOBRE SU LIBRO
Titulo: Mi libro favorito
Autor: Mi autor favorito
Año de Publicación: 1980
========================================
INFORMACIÒN SOBRE SU LIBRO
Titulo: Mi libro favorito
Autor: Mi autor favorito
Año de Publicación: 2000
```

4. **Gestión de Gallinas en Granja Digital:** Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.
    a. **Métodos requeridos**: ponerHuevo(), envejecer(), mostrarEstado().
    b. **Tarea**: Crear dos gallinas, simular sus acciones (envejecer y poner huevos), y mostrar su estado.

```java
package tp.introduccion.a.poo.angelelli.rodrigo;
public class Gallinas {

    public int idGallina;
    public int edad;
    public int huevosPuestos;

    public void ponerHuevo (int cantidad){
        if (cantidad > 0){
            huevosPuestos = huevosPuestos + cantidad;}
        else { System.out.println("Cantidad invalida");}
        }

    public void envejecer (int cantidad){
        if (cantidad > 0) {
          edad = edad + cantidad;}
        else { System.out.println("Imposible envejecer hacia atras");}
    }

    public void mostrarEstado (){
        System.out.println("Gallina "+idGallina + "\n" +
        "Edad: " +edad + " año(s) \n" +
        "Huevos puestos: " +huevosPuestos);
    }
```

```java
// EJERCICIO 4 - GALLINAS
System.out.println("============== EJERCICIO 4 ==============");
Gallinas g1 = new Gallinas();
g1.idGallina = 123;
g1.edad = 1;

Gallinas g2 = new Gallinas ();
g2.idGallina = 456;
g2.edad = 2;

g1.mostrarEstado();
g2.mostrarEstado();
System.out.println("=".repeat(40));

g1.ponerHuevo(2);
g1.envejecer(1);

g2.ponerHuevo(4);
g2.envejecer(1);

g1.mostrarEstado();
g2.mostrarEstado();
System.out.println("=".repeat(40));

g1.ponerHuevo(-2); // Error "Cantidad Invalida"
g1.envejecer(1);

g2.ponerHuevo(4);
g2.envejecer(-1); // Error "Imposible envejecer hacia atras"

g1.mostrarEstado();
g2.mostrarEstado();
```

```
============== EJERCICIO 4 ==============
Gallina 123
Edad: 1 año(s)
Huevos puestos: 0
Gallina 456
Edad: 2 año(s)
Huevos puestos: 0
========================================
Gallina 123
Edad: 2 año(s)
Huevos puestos: 2
Gallina 456
Edad: 3 año(s)
Huevos puestos: 4
========================================
Cantidad invalida
Imposible envejecer hacia atras
Gallina 123
Edad: 3 año(s)
Huevos puestos: 2
Gallina 456
Edad: 3 año(s)
Huevos puestos: 8
```

5. **Simulación de Nave Espacial:** Crear una clase NaveEspacial con los atributos: nombre, combustible.
    a. **Métodos requeridos:** despegar(), avanzar(distancia), recargarCombustible(cantidad), mostrarEstado().
    b. **Reglas**: Validar que haya suficiente combustible antes de avanzar y evitar que se supere el límite al recargar.
    c. **Tarea**: Crear una nave con 50 unidades de combustible, intentar avanzar sin recargar, luego recargar y avanzar correctamente. Mostrar el estado al final.

```java
package tp.introduccion.a.poo.angelelli.rodrigo;
public class NaveEspacial {
    private String nombre;
    private int combustible;
    private static final int MAX_COMBUSTIBLE = 100;

    public String getNombre() {...3 lines }

    public void setNombre(String nombre) {...3 lines }

    public int getCombustible() {...3 lines }

    public void setCombustible(int combustible) {...3 lines }

    public void mostrarEstado(){
        System.out.println("Nombre de la nave: "+nombre + "\n"
        + "Combustible disponible: " + combustible + " unidades");}

    public void despegar (){
    if (combustible >= 10){
        combustible-=10;
        System.out.println("Nave "+nombre + " ha despegado");
        System.out.println("Combustible restante: " + combustible);
    } else {
        System.out.println("No hay suficiente combustible para despegar. Al menos se necesitan 10 unidades de combustible.");
    }}

    public void avanzar (int distancia){
        if (combustible >= distancia) {
            combustible = combustible - distancia;
            System.out.println("La nave " +nombre+ " ha avanzado " +distancia+ " unidades de distancia.");
            System.out.println("Combustible restante: " +combustible);
        } else {
            System.out.println("Combustible insuficiente para avanzar.");
        }
```

```java
    public void recargarCombustible (int cantidad){
        if (cantidad <= 0){
            System.out.println("No es posible recargar una cantidad negativa de combustible");
            return;}
        if (combustible + cantidad > MAX_COMBUSTIBLE){
            combustible = MAX_COMBUSTIBLE;
            System.out.println("La cantidad ingresada es mayor a la cantidad maxima. Se ha recargado hasta el maximo disponible");
            System.out.println("Cantidad de combustible actual: " +combustible);
        } else {
            combustible = combustible + cantidad;
            System.out.println("Se han recargado " + cantidad + "unidades de combustible");
            System.out.println("Combustible actual luego de la recarga: "+combustible);
        }
    }
```

```java
// EJERCICIO 5 – NAVE ESPACIAL
System.out.println("============== EJERCICIO 5 ==============");

NaveEspacial nave1 = new NaveEspacial ();
nave1.setNombre("Apolo 23");
nave1.setCombustible(50);

System.out.println("Estado inicial de la nave ");
nave1.mostrarEstado();

System.out.println("=".repeat(40));

nave1.despegar();
nave1.avanzar(50);

System.out.println("=".repeat(40));
nave1.recargarCombustible(100);

System.out.println("=".repeat(40));

nave1.avanzar(200);
nave1.avanzar(80);
System.out.println("=".repeat(40));

nave1.avanzar(40);
nave1.recargarCombustible(10);
nave1.mostrarEstado();
```

```
============== EJERCICIO 5 ==============
Estado inicial de la nave
Nombre de la nave: Apolo 23
Combustible disponible: 50 unidades
========================================
Nave Apolo 23 ha despegado
Combustible restante: 40
Combustible insuficiente para avanzar.
========================================
La cantidad ingresada es mayor a la cantidad maxima. S
Cantidad de combustible actual: 100
========================================
Combustible insuficiente para avanzar.
La nave Apolo 23 ha avanzado 80 unidades de distancia.
Combustible restante: 20
========================================
Combustible insuficiente para avanzar.
Se han recargado 10unidades de combustible
Combustible actual luego de la recarga: 30
Nombre de la nave: Apolo 23
Combustible disponible: 30 unidades
BUILD SUCCESSFUL (total time: 0 seconds)
```