TRABAJO PRÁCTICO N° 3

Introducción a la Programación Orientada a Objetos.

Alumno: Ignacio Figueroa – 45.406.120

Tecnicatura Universitaria en Programación – UTN

Materia: Programación II

Comisión: 7

Link al código en GitHub: tup-utn/2do-cuatrimestre/programacion-II/introduccion-a-poo/apuntes/tp-introduccion-a-poo/tp-3-introduccion-a-poo at main · figueroaignacio/tup-utn

OBJETIVO GENERAL

Comprender los fundamentos de la Programación Orientada a Objetos, incluyendo clases, objetos, atributos y métodos, para estructurar programas de manera modular y reutilizable en Java.

1. **Registro de Estuadiantes**.
   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.

   Clase Estudiante:

```java
*/
public class Estudiante {
    private String nombre;
    private String apellido;
    private String curso;
    private double calificacion;

    public Estudiante(String nombre, String apellido, String curso, double calificacion) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.curso = curso;
        this.calificacion = calificacion;
    }

    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificación: " + calificacion);
    }

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

    public void bajarCalificacion(double puntos) {
        calificacion -= puntos;
        if (calificacion < 0) {
            calificacion = 0;
        }

        System.out.println("La calificación disminuyó. Nueva calificación: " + calificacion);
    }
}
```

   Instancia de la clase Estudiante:

```
Estudiante estudiante = new Estudiante("Ignacio", "Figueroa", "Programacion II", 8);

estudiante.mostrarInfo();

estudiante.subirCalificacion(1.5);
estudiante.mostrarInfo();

estudiante.bajarCalificacion(1);
estudiante.mostrarInfo();
```

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.

Clase Mascota:

```
 */
public class Mascota {
    private String nombre;
    private String especie;
    private int edad;


    public Mascota(String nombre, String especie, int edad) {
        this.edad = edad;
        this.nombre = nombre;
        this.especie = especie;
    }

    public void mostrarInfo() {
        System.out.println("Nombre: " + nombre);
        System.out.println("Especie: " + especie);
        System.out.println("Edad: " + edad);
    }

    public void cumplirAnios(int anios) {
        edad += anios;

        System.out.println(nombre + " cumplió " + edad);
    }
}

```

Instancia de la clase Mascota:

```
Mascota mascota = new Mascota("Ody", "Caniche", 8);

mascota.mostrarInfo();
mascota.cumplirAnios(1);
```

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 Libro:

```java
*/
public class Libro {

    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);
    }

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public void setAnioPublicacion(int anioPublicacion) {
        if (anioPublicacion > 0 && anioPublicacion ⩽ 2025) {
            this.anioPublicacion = anioPublicacion;
        } else {
            System.out.println("❌ Año inválido: " + anioPublicacion);
        }
    }

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

Instancia de la clase Libro

```java
Libro libro1 = new Libro("La Metamorfosis", "Franz Kafka", 0);
libro1.setAnioPublicacion(1915);
libro1.mostrarInfo();
```

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 Gallina

```
 */
public class Gallina {
    int idGallina;
    int edad;
    int huevosPuestos;

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

    public void ponerHuevo() {
        huevosPuestos++;
        System.out.println("La gallina: " + idGallina + " puso un huevo. Total: " + huevosPuestos);
    }

    public void envejecer() {
        edad++;
        System.out.println("La gallina: " + idGallina + " ahora tiene " + edad + " años");
    }

    public void mostrarEstado() {
        System.out.println("ID: " + idGallina);
        System.out.println("Edad: " + edad + " años");
        System.out.println("Huevos puestos: " + huevosPuestos);
        System.out.println("-----------------------");
    }
}
```

Instancia de la clase gallina

```
Gallina g1 = new Gallina(1, 2, 0);
Gallina g2 = new Gallina(2, 1, 0);

g1.ponerHuevo();
g1.ponerHuevo();
g1.envejecer();

g2.envejecer();
g2.ponerHuevo();

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

5. **Simulación de Nave Espacial**
   a. 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

```java
public class NaveEspacial {

    private String nombre;
    private double combustible;
    private final double CAPACIDAD_MAX = 100;

    public NaveEspacial(String nombre, double combustible) {
        this.nombre = nombre;
        this.combustible = combustible;
    }

    public void despegar() {
        if (combustible ≥ 10) {
            combustible -= 10;
            System.out.println("La nave " + nombre + " despegó. Combustible restante: " + combustible);
        } else {
            System.out.println("No hay suficiente combustible para despegar.");
        }
    }

    public void avanzar(double distancia) {
        double consumo = distancia;

        if (combustible ≥ consumo) {
            combustible -= consumo;
            System.out.println("La nave avanzó " + distancia + " km. Combustible restante: " + combustible);
        } else {
            System.out.println("No hay suficiente combustible para avanzar " + distancia + " km.");
        }
    }

    public void recargarCombustible(double cantidad) {
        if (combustible + cantidad > CAPACIDAD_MAX) {
            combustible = CAPACIDAD_MAX;
            System.out.println("⛽ Tanque lleno. Capacidad máxima alcanzada: " + CAPACIDAD_MAX);
        } else {
            combustible += cantidad;
            System.out.println("⛽ Se recargaron " + cantidad + " unidades. Combustible actual: " + combustible);
        }
    }

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

Instancia de la clase NaveEspacial

```
    NaveEspacial nave1 = new NaveEspacial("Apollo-X", 50);
    nave1.avanzar(60);
    
    nave1.despegar();

    nave1.avanzar(30);

    nave1.recargarCombustible(40);

    nave1.avanzar(20);

    nave1.mostrarEstado();
}
```