Programación II - TP3: Intro a POO

Alumno: *Romero, Abel Tomás (Comisión 5)*

1. Registro de Estudiantes.

- 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
package Ejercicio1;

public class Estudiante {

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

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

```java
    // Método que muestra información del estudiante
    public void mostrarInfo() {
        System.out.println("- Nombre completo: " + apellido + ", " + nombre);
        System.out.println("- Curso: " + curso);
        System.out.println("- Calificación: " + calificacion + "\n");
    }

    // Método para subir la calificación
    public void subirCalificacion(double puntos) {
        if (calificacion + puntos <= 10) {
            calificacion += puntos;
            System.out.println("La calificación ha subido a: " + calificacion);
        } else {
            System.out.println("ERROR: La calificación no debe superar el máximo de 10.");
        }
    }

    // Método para bajar la calificación
    public void bajarCalificacion(double puntos) {
        if (calificacion - puntos >= 0) {
            calificacion -= puntos;
            calificacion = Math.round(calificacion * 10.0) / 10.0; // Redondear a 1 decimal
            System.out.println("La calificación ha bajado a: " + calificacion);
        } else {
            System.out.println("ERROR: La calificación no debe ser menos que el mínimo de 0.");
        }
    }
```

```java
    // Getters y Setters
    public String getNombre() {
        return nombre;
    }

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

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }

    public double getCalificacion() {
        return calificacion;
    }

    private void setCalificacion(double calificacion) {
        if (calificacion < 0) {
            this.calificacion = 0;  // Si se inserta una calificación negativa, pasa a 0
        } else if (calificacion > 10) {
            this.calificacion = 10; // Si se inserta una calificación arriba del máximo, pasa a 10
        } else {
            this.calificacion = calificacion;
        }
```

```java
public class Mainl {

    public static void main(String[] args) {

        // Instancia de la clase Estudiante
        Estudiante estudiantel = new Estudiante("Abel Tomás", "Romero", "Programación II", 7.2);

        // Mostrar información inicial
        System.out.println("\nInformación inicial del estudiante:");
        estudiantel.mostrarInfo();

        // ERRORES
        estudiantel.subirCalificacion(3.3);  // ERROR: Pasa de 10
        estudiantel.bajarCalificacion(12.2); // ERROR: Pasa a ser negativa

        // Subir  la calificación
        estudiantel.subirCalificacion(2.5); // Sube correctamente sin pasar arriba de 10

        // Bajar la calificación
        estudiantel.bajarCalificacion(0.8); // Baja correctamente sin pasar debajo de 0

        // Mostrar información final
        System.out.println("\nInformación final del estudiante:");
        estudiantel.mostrarInfo();
```

```
Informaci□n inicial del estudiante:
- Nombre completo: Romero, Abel Tom□s
- Curso: Programaci□n II
- Calificaci□n: 7.2

ERROR: La calificaci□n no debe superar el m□ximo de 10.
ERROR: La calificaci□n no debe ser menos que el m□nimo de 0.
La calificaci□n ha subido a: 9.7
La calificaci□n ha bajado a: 8.9

Informaci□n final del estudiante:
- Nombre completo: Romero, Abel Tom□s
- Curso: Programaci□n II
- Calificaci□n: 8.9
```

2. Registro de Mascotas.

- 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
package Ejercicio2;

public class Mascota {

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

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

```java
    // Método que muestra la información
    public void mostrarInfo() {
        System.out.println("- Nombre: " + nombre);
        System.out.println("- Especie: " + especie);
        System.out.println("- Edad: " + edad + " años\n");
    }

    // Método que permite cumplir años
    public void cumplirAnios(int anios) {
        if (anios >= 1) {
            edad += anios;
            System.out.println("\nSimulando paso del tiempo de " + anios + " años.");
            System.out.println("Ahora " + nombre + " ha cumplido " + edad + " años.");
        } else {
            System.out.println("ERROR: Los años a cumplir debe ser mayor o igual a 1.");
        }
```

```java
    // Getters y Setters
    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;
    }

    private void setEdad(int edad) {
        if (edad < 0) {
            System.out.println("ADVERTENCIA: Edad negativa (" + edad + ") corregida a 1 año.\n");
            this.edad = 1;
        } else if (edad == 0) {
            this.edad = 1;
            System.out.println("ADVERTENCIA: Edad 0 corregida a 1 año.\n");
        } else {
            this.edad = edad;
        }
```

```java
package Ejercicio2;

public class Main2 {

    public static void main(String[] args) {

        // Instancia de la clase Mascota
        Mascota mascota1 = new Mascota("Frida", "Perro", 7);

        // Mostrar información inicial
        System.out.println("Información inicial de la mascota:");
        mascota1.mostrarInfo();

        // ERRORES
        mascota1.cumplirAnios(0);  // ERROR: No puede cumplir 0 años

        // Paso del tiempo
        mascota1.cumplirAnios(3);

        // Mostrar información final con los cambios
        System.out.println("\nInformación final de la mascota:");
        mascota1.mostrarInfo();

```

```
run:
Informaci□n inicial de la mascota:
- Nombre: Frida
- Especie: Perro
- Edad: 7 a□os

ERROR: Los a□os a cumplir debe ser mayor o igual a 1.

Simulando paso del tiempo de 3 a□os.
Ahora Frida ha cumplido 10 a□os.

Informaci□n final de la mascota:
- Nombre: Frida
- Especie: Perro
- Edad: 10 a□os
```

3. Encapsulamiento con la Clase Libro

- 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
package Ejercicio3;

public class Libro {

    // Atributos
    private final String titulo;
    private final String autor;
    private int anioPublicacion;

    // Constructor
    public Libro(String titulo, String autor, int anioPublicacion) {
        this.titulo = titulo;
        this.autor = autor;
        this.anioPublicacion = anioPublicacion;
    }
```

```java
    // Getters
    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnioPublicacion() {
        return anioPublicacion;
    }

    // Setters
    public void setAnioPublicacion(int nuevoAnio) {
        int anioActual = java.time.Year.now().getValue();
        if (nuevoAnio >= 1700 && nuevoAnio <= anioActual) {
            System.out.println("Año actualizado correctamente a: " + nuevoAnio);
            this.anioPublicacion = nuevoAnio;
        } else {
            System.out.println("\nERROR: El año debe estar entre 1700 y " + anioActual);
        }
```

```java
package Ejercicio3;

public class Main3 {

    public static void main(String[] args) {

        // Instancia de la clase Libro
        Libro libro1 = new Libro("12 Reglas para vivir", "Jordan Peterson", 2012);

        // Mostrar información inicial
        System.out.println("\nInformación inicial del libro:");
        libro1.mostrarInfoLibro();

        // ERRORES
        libro1.setAnioPublicacion(-1212);

        // Actualizar el año correctamente
        libro1.setAnioPublicacion(2018);

        // Mostrar información actualizada
        System.out.println("\nInformación actualizada:");
        libro1.mostrarInfoLibro();
```

```
Informaci□n inicial del libro:
- T□tulo: 12 Reglas para vivir
- Autor: Jordan Peterson
- A□o de Publicaci□n: 2012

ERROR: El a□o debe estar entre 1700 y 2025
A□o actualizado correctamente a: 2018

Informaci□n actualizada:
- T□tulo: 12 Reglas para vivir
- Autor: Jordan Peterson
- A□o de Publicaci□n: 2018
```

4. Gestión de Gallinas en Granja Digital

- Crea una clase Gallina con los atributos: idGallina, edad, huevoPuestos.
- **Métodos requeridos:** `ponerHuevo()`, `envejecer()`, `mostrarEstado()`.
- **Tarea:** Crear dos gallinas, simular sus acciones (envejecer y poner huevos), y mostrar su estado.

```java
package Ejercicio4;

public class Gallina {

    // Atributos
    private static int contadorId = 1; // Contador para el autoincremento del id
    private final int idGallina;
    private int edad;
    private int huevosPuestos = 0; // Empieza con cero huevos puestos

    // Constructor
    public Gallina(int edad) {
        this.idGallina = contadorId++;
        setEdad(edad);
    }
```

```java
    // Método que hace poner un huevo a la gallina
    public void ponerHuevo() {
        huevosPuestos++;
        System.out.println("La gallina " + idGallina + " ha puesto un huevo.");
    }

    // Método que hace envejecer a la gallina
    public void envejecer() {
        edad++;
        System.out.println("La gallina " + idGallina + " ha envejecido un año.");
    }

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

    // Setter para la edad
    private void setEdad(int edad) {
        if (edad <= 0) {
            System.out.println("ADVERTENCIA: Edad inválida (" + edad + ") corregida a 1 año.");
            this.edad = 1;
        } else {
            this.edad = edad;
        }
```

```java
package Ejercicio4;

public class Main4 {

    public static void main(String[] args) {

        // Instancias de la clase Gallina
        Gallina gallina1 = new Gallina(2);
        Gallina gallina2 = new Gallina(3);

        // Mostrar estados iniciales
        System.out.println("\nEstado inicial de las gallinas:");
        gallina1.mostrarEstado();
        gallina2.mostrarEstado();

        // Simular acciones de la primer gallina
        System.out.println("\nSimulando acciones de la primer gallina:");
        gallina1.ponerHuevo();
        gallina1.ponerHuevo();
        gallina1.envejecer();

        // Simular acciones de la segunda gallina
        System.out.println("\nSimulando acciones de la segunda gallina:");
        gallina2.ponerHuevo();
        gallina2.ponerHuevo();
        gallina2.ponerHuevo();
        gallina2.envejecer();

        // Mostrar estados finales
        System.out.println("\nEstado final de las gallinas:");
        gallina1.mostrarEstado();
        gallina2.mostrarEstado();
```

```
Estado inicial de las gallinas:
- ID Gallina: 1
- Edad: 2 a�os
- Huevos puestos: 0
----------------------
- ID Gallina: 2
- Edad: 3 a�os
- Huevos puestos: 0
----------------------

Simulando acciones de la primer gallina:
La gallina 1 ha puesto un huevo.
La gallina 1 ha puesto un huevo.
La gallina 1 ha envejecido un a�o.

Simulando acciones de la segunda gallina:
La gallina 2 ha puesto un huevo.
La gallina 2 ha puesto un huevo.
La gallina 2 ha puesto un huevo.
La gallina 2 ha envejecido un a�o.

Estado final de las gallinas:
- ID Gallina: 1
- Edad: 3 a�os
- Huevos puestos: 2
----------------------
- ID Gallina: 2
- Edad: 4 a�os
- Huevos puestos: 3
----------------------
BUILD SUCCESSFUL (total time: 0 seconds)
```

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
package Ejercicio5;

public class NaveEspacial {

    // Atributos
    private final String nombre;
    private int combustible;
    private boolean haDespegado; // Bandera si la nave despegó o no
    private static final int MAX_COMBUSTIBLE = 100; // Límite maximo de combustible

    // Constructor
    public NaveEspacial(String nombre, int combustibleInicial) {
        this.nombre = nombre;
        this.combustible = Math.min(combustibleInicial, MAX_COMBUSTIBLE);
        this.haDespegado = false;
    }
```

```java
    // Método para despegar la nave
    public void despegar() {
        if (haDespegado) {
            System.out.println("ERROR: La nave ya ha despegado.\n");
        } else if (combustible < 10) {
            System.out.println("ERROR: No puede despegar con menos de 10 unidades de combustible.\n");
        } else {
            combustible -= 10;
            haDespegado = true;
            System.out.println("Comenzando despegue...");
            System.out.println(nombre + " ha despegado. Combustible restante: " + combustible + "\n");
        }
```

```java
    // Método para que la nave avance
    public void avanzar(int distancia) {
        if (!haDespegado) {
            System.out.println("ERROR: La nave debe despegar primero.\n");
        } else if (distancia > combustible) {
            System.out.println("DENEGADO: Sin combustible suficiente para avanzar. Combustible actual: " + combustible + "\n");
        } else if (distancia <= 0) {
            System.out.println("ERROR: Distancia inválida. Ingrese una distancia mayor a 0.\n");
        } else {
            combustible -= distancia;
            System.out.println("Avanzando " + distancia + " unidades de distancia...\n");
        }
```

```java
    // Método para recargar combustible
    public void recargarCombustible(int cantidad) {
        if ((combustible + cantidad) > MAX_COMBUSTIBLE) {
            int restante = 100 - combustible;
            combustible = MAX_COMBUSTIBLE;
            System.out.println("DENEGADO: La cantidad a cargar supera el límite del tanque (" + MAX_COMBUSTIBLE + ").");
            System.out.println("Cargando hasta el límite...");
            System.out.println("Se ha cargado " + restante + " unidades de combustible. Tanque lleno.\n");
        } else if (cantidad <= 0) {
            System.out.println("ERROR: Cantidad de combustible a cargar inválida. Cargue entre 1 y 100 unidades.\n");
        } else {
            combustible += cantidad;
            System.out.println("Se ha cargado " + cantidad + " unidades de combustible. Total actual: " + combustible + "\n");
        }
    }

    // Método para mostrar el estado
    public void mostrarEstado() {
        System.out.println("- Nave: " + nombre);
        System.out.println("- Unidades de combustible: " + combustible);
        if (haDespegado) {
            System.out.println("- " + nombre + " en movimiento...\n");
        } else {
            System.out.println("- " + nombre + " aún en tierra.\n");
        }
    }
```

```java
package Ejercicio5;

public class Main5 {

    public static void main(String[] args) {

        // Instancia de la clase NaveEspacial
        NaveEspacial nave1 = new NaveEspacial("Cowboy Bebop", 50);

        // Mostrar estado inicial de la nave
        nave1.mostrarEstado();

        // Despegar nave
        nave1.despegar();

        // ERROR: Intentar avanzar 60 unidades sin recargar
        nave1.avanzar(60);

        // Recargar 40 unidades de combustible
        nave1.recargarCombustible(40);

        // Intentar avazar 60 unidades nuevamente
        nave1.avanzar(60);

        // Mostrar estado final de la nave
        nave1.mostrarEstado();
```

```
run:
- Nave: Cowboy Bebop
- Unidades de combustible: 50
- Cowboy Bebop a◆n en tierra.

Comenzando despegue...
Cowboy Bebop ha despegado. Combustible restante: 40

DENEGADO: Sin combustible suficiente para avanzar. Combustible actual: 40

Se ha cargado 40 unidades de combustible. Total actual: 80

Avanzando 60 unidades de distancia...

- Nave: Cowboy Bebop
- Unidades de combustible: 20
- Cowboy Bebop en movimiento...

BUILD SUCCESSFUL (total time: 0 seconds)
```

Link del repo de GitHub:
https://github.com/Tomu98/UTN-TUPaD-P2-TPs/tree/main/03%20Introducci%C3%B3n%20a%20POO