TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN A DISTANCIA

PROGRAMACIÓN II

Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Alumno:
Matias Carro - matiasmanuelcarro@gmail.com
DNI: 37362003

Materia: Programación 2

Repositorio con los ejercicios:
https://github.com/MatiasManuelCarro/UTN-TUPaD-P2/tree/main/Matias_Carro_TP_3

Caso Práctico

1. Registro de Estudiantes

a. Crear una clase Estudiante con los atributos: nombre, apellido, curso, calificación.
Métodos requeridos: mostrarInfo(), subirCalificacion(puntos), bajarCalificacion(puntos).

Código del programa:

Main():

```
public class Matias_Carro_TP3_Ej1 {

 /**
  * @param args the command line arguments
  */
 public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    String nombre, apellido, curso;
    double calificacion;

    //Creamos un estudiante
    System.out.println("Ingrese el nombre del estudiante:");
    nombre = input.nextLine();

    System.out.println("Ingrese el Apellido del estudiante:");
    apellido = input.nextLine();

    System.out.println("Ingrese la materia:");
    curso = input.nextLine();

    System.out.println("Ingrese la calificacion");
    calificacion = Integer.parseInt(input.nextLine());
    Estudiante estudiante1  = new Estudiante(nombre, apellido, curso, calificacion);

    System.out.println("\n Datos del estudiante:");
    estudiante1.mostrarInfo();

   //Se sube la calificacion 1 punto
```

```
        estudiante1.subirCalificacion(1.0);

        //Se baja la calificacion 0.5 puntos
        estudiante1.bajarCalificacion(0.5);

        //mostramos las nuevas calificaciones modificadas
        System.out.println("\nCalificacion final:");
         estudiante1.mostrarInfo();
    }

}
```

Clase Estudiante:

```
public class Estudiante {
    String nombre;
    String apellido;
    String curso;
    double calificacion;


    //Constructor
    public Estudiante(String nombre, String apellido, String curso, double calificacion) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.curso = curso;
        this.calificacion = calificacion;
    }
    //Metodo para mostrar la informacion
    public void mostrarInfo() {
        System.out.println("Nombre: "+ nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificacion: " + calificacion);
    }

        // Método para subir la calificación
    public void subirCalificacion(double nota) {
        calificacion += nota;
        System.out.println("Nueva calificación: " + calificacion);
    }
```

```java
    // Método para bajar la calificación
    public void bajarCalificacion(double nota) {
        calificacion -= nota;
        System.out.println("Nueva calificación: " + calificacion);
    }

}
```

Tarea: Instanciar a un estudiante, mostrar su información, aumentar y disminuir calificaciones.

```
Ingrese el nombre del estudiante:
Matias
Ingrese el Apellido del estudiante:
Carro
Ingrese la materia:
Programacion
Ingrese la calificacion
8

 Datos del estudiante:
Nombre: Matias Carro
Curso: Programacion
Calificacion: 8.0
Nueva calificación: 9.0
Nueva calificación: 8.5

Calificacion final:
Nombre: Matias Carro
Curso: Programacion
Calificacion: 8.5
BUILD SUCCESSFUL (total time: 15 seconds)
```

2. Registro de Mascotas

a. Crear una clase Mascota con los atributos: nombre, especie, edad.
Métodos requeridos: mostrarInfo(), cumplirAnios().

Código:

Main()

```
package matias_carro_tp3_ej2;

import java.util.Scanner;

/**
*
* @author Matias
*/
public class Matias_Carro_TP3_Ej2 {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    //creamos una mascota
    String nombre, especie;
    int edad;

    System.out.println("Ingrese el nombre de la mascota:");
      nombre = input.nextLine();

      System.out.println("Ingrese la especie:");
      especie = input.nextLine();

      System.out.println("Ingrese la edad");
      edad = Integer.parseInt(input.nextLine());

    Mascota mascota1 = new Mascota(nombre, especie, edad);
```

```
    mascota1.mostrarInfo();

    mascota1.cumplirAnios();

    mascota1.mostrarInfo();
  }

}
```

Clase Mascota:

```
package matias_carro_tp3_ej2;

/**
*
* @author Matias
*/
public class Mascota{
  String nombre;
  String especie;
  int edad;

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

public void mostrarInfo(){
  System.out.println("\nDatos de la mascota:");
  System.out.println(nombre);
  System.out.println(especie);
  System.out.println(edad);
}

public void cumplirAnios(){
  System.out.println("Feliz cumpleaños " + nombre + "!");
  edad ++;
}
```

```
}
```

Tarea: Crear una mascota, mostrar su información, simular el paso del tiempo y verificar los cambios.

```
Ingrese el nombre de la mascota:
Stitch
Ingrese la especie:
Bulldog Frances
Ingrese la edad
5

Datos de la mascota:
Stitch
Bulldog Frances
5
Feliz cumpleaños Stitch!

Datos de la mascota:
Stitch
Bulldog Frances
6
BUILD SUCCESSFUL (total time: 13 seconds)
```

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.

Codigo:

Main()

```
package matias_carro_tp3_ej3;

import java.util.Scanner;

/**
*
* @author matias.carro
*/
```

```java
public class Matias_Carro_TP3_Ej3 {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {
      int año;
      Scanner input = new Scanner(System.in);

      //creamos un nuevo libro
      Libro libro1 = new Libro("El cuervo","Edgar Allan Poe",1840);

      //mostramos el nuevo libro con los Getter
      System.out.println("Titulo del libro: " + libro1.getTitulo() + "\nAutor: " + libro1.getAutor()
+ " \nAño: " +  libro1.getAñoPublicacion());

      //se pide año nuevo para el libro
      System.out.println("Ingrese nuevo año para el libro");
      año = Integer.parseInt(input.nextLine());
      libro1.setAñoPublicacion(año);

      System.out.println("Titulo del libro: " + libro1.getTitulo() + "\nAutor: " + libro1.getAutor()
+ " \nAño: " +  libro1.getAñoPublicacion());

  }

}
```

Clase Libro:

```java
package matias_carro_tp3_ej3;

/**
*
* @author matias.carro
*/
public class Libro {
  private String titulo, autor;
  private int añoPublicacion;
```

```java
public Libro(String titulo, String autor, int añoPublicacion) {
    this.titulo = titulo;
    this.autor = autor;
    this.añoPublicacion = añoPublicacion;
}

//Getter titulo
public String getTitulo(){
    return titulo;
}

//Getter autor
public String getAutor(){
    return autor;
}

 //Getter añoPublicacion
public int getAñoPublicacion(){
    return añoPublicacion;
}

public void setAñoPublicacion(int año){
    if (año > 0 && año < 2025){
    this.añoPublicacion = año;
        System.out.println("Año modificado correctamente");
    } else {
        System.out.println("Ingrese un año correcto.");
    }

}

}
```

Tarea: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.

Valor de año invalido:

```
run:
Picked up JAVA_TOOL_OPTIONS: -Dfile.encodi
Titulo del libro: El cuervo
Autor: Edgar Allan Poe
Año: 1840
Ingrese nuevo año para el libro
-560
Ingrese un año correcto.
Titulo del libro: El cuervo
Autor: Edgar Allan Poe
Año: 1840
BUILD SUCCESSFUL (total time: 17 seconds)
```

Valor de Año válido:

```
Titulo del libro: El cuervo
Autor: Edgar Allan Poe
Año: 1840
Ingrese nuevo año para el libro
1845
Año modificado correctamente
Titulo del libro: El cuervo
Autor: Edgar Allan Poe
Año: 1845
BUILD SUCCESSFUL (total time: 2 seconds)
```

4. Gestión de Gallinas en Granja Digital

a. Crear una clase Gallina con los atributos: idGallina, edad, huevosPuestos.
Métodos requeridos: ponerHuevo(), envejecer(), mostrarEstado().

Código:

Main()

```java
package matias_carro_tp3_ej4;

/**
*
* @author Matias
*/
public class Matias_Carro_TP3_Ej4 {
```

```
 /**
  * @param args the command line arguments
  */
 public static void main(String[] args) {
 int idGallina;
 int huevosPuestos;
 int edad;

 //se crean las gallinas
 Gallina gallina1 = new Gallina(1, 0, 0);
 Gallina gallina2 = new Gallina(2, 0, 0);

 //simulamos un año de huevos puesto por la gallina 1
 for (int i = 0; i <  200; i++){
 gallina1.ponerHuevo();
 }
 //gallina 1 envejece 1 año
 gallina1.envejecer();

 gallina1.mostrarEstado();

 //simulamos dos años de huevos puesto por la gallina 2
 for (int i = 0; i <  400; i++){
 gallina2.ponerHuevo();
 }
 //gallina 2 envejece 2 años
   for (int i = 0; i <  2; i++){
 gallina2.envejecer();
 }

gallina2.mostrarEstado();
 }

}
```

Clase Gallina

```
package matias_carro_tp3_ej4;
```

```java
/**
*
* @author Matias
*/
public class Gallina {
int idGallina;
int huevosPuestos;
int edad;

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

public void ponerHuevo(){
huevosPuestos ++;
}

public void envejecer(){
edad++;
}

public void mostrarEstado(){
  System.out.println("\nGallina " + idGallina + "\nEdad: " + edad + "\nHuevos puestos: " +
huevosPuestos);
}

}
```

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

```
Gallina 1
Edad: 1
Huevos puestos: 200

Gallina 2
Edad: 2
Huevos puestos: 400
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.

Codigo:

Main()

```
package matias_carro_tp3_ej5;


/**
*
* @author Matias
*/
public class Matias_Carro_TP3_Ej5 {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {

   //se crea la navecon 50 de combustible
   NaveEspacial nave1 = new NaveEspacial("Millenium Falcon", 50);

   //se despega la nave
     System.out.println("*Despegando la nave*");
   nave1.despegar();
```

```
  //se intenta avanzar sin combustible suficiente
    System.out.println("\n*Intentando avanzar con 50 de combustible:*");
  nave1.avanzar(100);

  //se recarga combustible para el viaje
    System.out.println("\n*Se recarga combustible*");
  nave1.recargarCombustible(125);

  //se logra avanzar
    System.out.println("\n*Intentamos avanzar nuevamente*");
  nave1.avanzar(100);

  //se muestra estado al final
  nave1.mostrarEstado();
 }

}
```

Clase NaveEspacial

```
package matias_carro_tp3_ej5;

/**
*
* @author Matias
*/
public class NaveEspacial {
  String nombre;
  int combustible;
  boolean naveDespegada = false;
  int distanciaViajada;

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


  public void despegar(){
    if (naveDespegada == true){
```

```
        System.out.println("La nave ya se encuentra despegada! ");
    }else{
        naveDespegada = true;
        System.out.println("Despegue en 3... 2... 1...");
        System.out.println("Nave en orbita");
    }
    }


  public void avanzar(int distancia){
    if (combustible <= 50 ){
        System.out.println("Combustible muy bajo para el salto, recargue.");
    }else if (combustible > 50 && naveDespegada == true){
        System.out.println("Comenzando viaje");
        distanciaViajada += distancia;
        combustible -= distancia;
    }
  }

   public void recargarCombustible(int cantidad){
     int cantidadMaximaCombustible = 250;
     if (combustible + cantidad < cantidadMaximaCombustible){
     combustible += cantidad;
     System.out.println("Se recargan "+cantidad+" unidades de combustible\nCombustible
actual: "+combustible);
   }else{
         System.out.println("La carga supera el maximo, ingrese menos de "+
(cantidadMaximaCombustible-combustible));
      }
   }

     public void mostrarEstado(){
     System.out.println("\nEstadisticas del viaje de la nave: "+nombre);
     System.out.println("distancia realizada: "+distanciaViajada);
     System.out.println("Combustible restante: "+combustible);
   }
}
```

Intento de recargar demasiado combustible (más de 250 unidades en total)

```
*Despegando la nave*
Despegue en 3... 2... 1...
Nave en orbita

*Intentando avanzar con 50 de combustible:*
Combustible muy bajo para el salto, recargue.

*Se recarga combustible*
La carga supera el maximo, ingrese menos de 200

*Intentamos avanzar nuevamente*
Combustible muy bajo para el salto, recargue.

Estadisticas del viaje de la nave: Millenium Falcon
distancia realizada: 0
Combustible restante: 50
BUILD SUCCESSFUL (total time: 0 seconds)
```

Cargando la cantidad correcta y viajando:

```
*Despegando la nave*
Despegue en 3... 2... 1...
Nave en orbita

*Intentando avanzar con 50 de combustible:*
Combustible muy bajo para el salto, recargue.

*Se recarga combustible*
Se recargan 125 unidades de combustible
Combustible actual: 175

*Intentamos avanzar nuevamente*
Comenzando viaje

Estadisticas del viaje de la nave: Millenium Falcon
distancia realizada: 100
Combustible restante: 75
BUILD SUCCESSFUL (total time: 0 seconds)
```