PROGRAMACIÓN II

Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

Caso Práctico

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

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 Estudiante {
    String nombre;
    String apellido;
    String curso;
    int calificacion;

    public void mostrarInfo(){
        System.out.println("Nombre " + nombre);
        System.out.println("Apellido " + apellido);
        System.out.println("Curso " + curso);
        System.out.println("Calificación " + calificacion);
    };
    public void subirCalificacion(int puntos){
        this.calificacion += puntos;
        System.out.println("Nueva calificación " + calificacion);
    };
    public void bajarCalificacion(int puntos){
        this.calificacion -= puntos;
        System.out.println("Nueva calificación " + calificacion);
    };

}
```

```
*/
public class PrimerObjecto {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Estudiante e1 = new Estudiante();
        e1.nombre = "Carlos";
        e1.apellido = "Garcia";
        e1.curso = "A1";
        e1.calificacion = 10;

        e1.mostrarInfo();
        e1.subirCalificacion(2);
        e1.bajarCalificacion(4);
    }

}
```

```
ut ×
UTN-TUPaD-P2 - C:\Users\marus\Documents\TUP\UTN-TUPaD-P2 ×   PrimerObjecto (run) ×
run:
Nombre Carlos
Apellido Garcia
Curso A1
Calificaci□n 10
Nueva calificaci□n 12
Nueva calificaci□n 8
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

```java
public class Mascota {
    String nombre;
    String especie;
    int edad;

    public void mostrarInfo(){
        System.out.println("Nombre " + nombre);
        System.out.println("Especie " + especie);
        System.out.println("Edad " + edad);
    };
    public void cumplirAnios(){
        this.edad += 1;
    };
}
```

```java
//        e1.subirCalificacion(2);
//        e1.bajarCalificacion(4);
        Mascota m1 = new Mascota();
        m1.nombre = "Pelusa";
        m1.especie="gato";
        m1.edad=5;

        m1.mostrarInfo();
        m1.cumplirAnios();
        m1.mostrarInfo();
    }
```

```
primerobjecto.PrimerObjecto > main >
UTN-TUPaD-P2 - C:\Users\marus\Documents\TUP\UTN-TUPaD-P2 ×   PrimerObjecto (
run:
Nombre Pelusa
Especie gato
Edad 5
Nombre Pelusa
Especie gato
Edad 6
BUILD SUCCESSFUL (total time: 0 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.
Tarea: Crear un libro, intentar modificar el año con un valor inválido y luego con uno válido, mostrar la información final.

```java
*/
public class Libro {
    private String titulo;
    private String autor;
    private int anioPublicacion;

    public String getTitulo(){
        return this.titulo;
    }
    public String getAutor(){
        return this.autor;
    }
    public int getAnioPublicacion(){
        return this.anioPublicacion;
    }
    public void setAnoPub(int nuevoAnio){
        if (nuevoAnio > 0 && nuevoAnio <= 2025){
            this.anioPublicacion = nuevoAnio;
        }
```

```java
//          m1.mostrarInfo();

        Libro l1 = new Libro();
        System.out.println(l1.getAnioPublicacion());
        l1.setAnoPub(2027);
        System.out.println(l1.getAnioPublicacion());
        l1.setAnoPub(2024);
        System.out.println(l1.getAnioPublicacion());

    }

}
```

Previous Next Select

primerobjecto.PrimerObjecto > main

tput

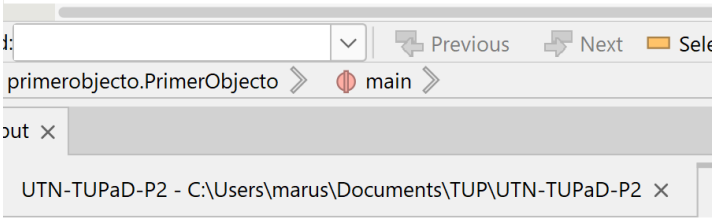
UTN-TUPaD-P2 - C:\Users\marus\Documents\TUP\UTN-TUPaD-P2 | PrimerObjecto (

```
run:
2023
2023
2024
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

```java
*/
public class Gallina {
    private int idGallina;
    public int edad;
    public int huevosPuestos = 0;

    public void ponerhuevo(){
        this.huevosPuestos += 1;
    };
    public void envejecer(){
        this.edad += 1;
    };
    public void mostrarestado(){
        System.out.println("Huevos puestos: " + this.huevosPuestos);
        System.out.println("Edad: " + this.edad);
    };
}
```

```java
        Gallina g1 = new Gallina();
        Gallina g2 = new Gallina();

        g1.envejecer();
        g1.envejecer();
        g1.ponerhuevo();
        g2.envejecer();
        g2.envejecer();
        g2.envejecer();
        g2.ponerhuevo();
        g2.ponerhuevo();
        g1.mostrarestado();
        g2.mostrarestado();

    }

}
```




```
run:
Huevos puestos: 1
Edad: 2
Huevos puestos: 2
Edad: 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
    public void setNuevaNave(String nuevoNombre, int unidades){
        this.nombre = nuevoNombre;
        this.combustible = unidades;
    };

    public void despegar(){
        System.out.println("Despegando");
    };
    public void avanzar(int distancia){
        if(this.combustible <= 50){
            System.out.println("No es posible avanzar. Debe recargar combustible");
        } else {
            System.out.println("Avanzando " + distancia + "Km");
        }
    };
    public void recargarCombustible(int cantidad){
        if((this.combustible + cantidad)> 100){
            System.out.println("Demasiado combustible");
        } else {
            this.combustible += cantidad;
            System.out.println("Combustible cargado");
        }
    };
    public void mostrarEstado(){
        System.out.println("Nave " + this.nombre + " posee " + this.combustible + " unidades de comb
    };
}
```

```java
        NaveEspacial n1 = new NaveEspacial();
        n1.setNuevaNave( "WW1", 50);
        n1.avanzar(60);
        n1.recargarCombustible(20);
        n1.avanzar(60);
        n1.mostrarEstado();

    }
```

Previous Next Select

primerobjecto.PrimerObjecto > main >

ut ×

UTN-TUPaD-P2 - C:\Users\marus\Documents\TUP\UTN-TUPaD-P2 × | PrimerObjecto (run) ×

```
run:
No es posible avanzar. Debe recargar combustible
Combustible cargado
Avanzando 60Km
Nave WW1 posee 70 unidades de combustible
BUILD SUCCESSFUL (total time: 0 seconds)
```