PROGRAMACIÓN II
Trabajo Práctico 3: Introducción a la Programación Orientada a Objetos

1)

```java
import java.util.Scanner;

public class Introduccion_POO {
    public static void main(String[]arg){
        Scanner input = new Scanner(System.in);
/*
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.
 */

        Estudiantes alumno = new Estudiantes();

        // Asignar valores con setters
        alumno.setNombre("Fabian Ignacio");
        alumno.setApellido("Cardozo");
        alumno.setCurso("5to F");
        alumno.setCalificacion(8);

        // Mostrar información inicial
        alumno.mostrarInfo();

        // Subir calificación
        alumno.subirCalificacion( puntos: 3);
        System.out.println("Calificación después de subir: " + alumno.getCalificacion());

        // Bajar calificación
        alumno.bajarCalificacion( puntos: 2);
        System.out.println("Calificación después de bajar: " + alumno.getCalificacion());
```

```java
public class Estudiantes {  2 usages
    private String nombre;  2 usages
    private String apellido;  2 usages
    private String curso;  2 usages
    private double calificacion;  7 usages

    public void setNombre(String nom) {  1 usage
        nombre = nom;
    }
    public void setApellido(String ape) {  1 usage
        apellido = ape;
    }
    public void setCurso(String cur) {  1 usage
        curso = cur;
    }
    public void setCalificacion(double calif) {  1 usage
        calificacion = calif;
    }
    public void mostrarInfo() {  1 usage
        System.out.println("Datos del estudiante:");
        System.out.println("Nombre: " + nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificación: " + calificacion);
        System.out.println("-------------------------");
    }
    public void subirCalificacion(double puntos) {  1 usage
        calificacion = calificacion + puntos;
    }

    public void bajarCalificacion(double puntos) {  1 usage
        calificacion = calificacion - puntos;

    }
    public double getCalificacion() {  2 usages
        return calificacion;
    }
```

2)

```java
/*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.
 */

    Mascotas perro= new Mascotas();

    perro.nombre = "Firulais";
    perro.especie = "Caniche Toy";
    perro.edad = 5;

    perro.monstrarInfo();
    perro.cumplirAños(2);
    perro.monstrarInfo();
```

```java
public class Mascotas {  2 usages

    String nombre;  2 usages
    String especie;  2 usages
    int edad;  4 usages


public void monstrarInfo(){  2 usages
    System.out.println("Nombre Mascota: " + nombre);
    System.out.println("Especie:" + especie);
    System.out.println("Edad: " + edad);
    }
public int cumplirAños(int año){  1 usage
    edad += año;
    return edad;
    }
```

3)

```java
/*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.

 */
        Libro libro1 = new Libro();


        libro1.setTitulo("Platero y yo");
        libro1.setAutor("Juan Ramon Jimenez");


        boolean ok1 = libro1.setAnioPublicacion(-2);
        System.out.println("¿Se aceptó -2? " + ok1);


        boolean ok2 = libro1.setAnioPublicacion(1914);
        System.out.println("¿Se aceptó 1914? " + ok2);


        libro1.getInfo();
```

```java
public class Libro {  2 usages

private String titulo;  4 usages
private String autor;  4 usages
private int anioPublicacion;  4 usages


private static final int ANIO_MIN = 1450;  3 usages
private static final int ANIO_MAX = 2025;  2 usages

// Constructor por defecto
@Contract(pure = true)
public Libro() {  1 usage
    titulo = "(sin titulo)";
    autor = "(sin autor)";
    anioPublicacion = ANIO_MIN;
}


public String getTitulo() { return titulo; }  no usages
public String getAutor() { return autor; }  no usages
public int getAnioPublicacion() { return anioPublicacion; }  no usages


public void setTitulo(String tituloArg) {  1 usage
    if (tituloArg != null && !tituloArg.isEmpty()) {
        titulo = tituloArg;
    }
}

public void setAutor(String autorArg) {  1 usage
    if (autorArg != null && !autorArg.isEmpty()) {
        autor = autorArg;
    }
}
```

```java
public boolean setAnioPublicacion(int nuevoAnio) {  2 usages
    if (nuevoAnio >= ANIO_MIN && nuevoAnio <= ANIO_MAX) {
        anioPublicacion = nuevoAnio;
        return true;
    }

    System.out.println("Año inválido: " + nuevoAnio + " (permitido " + ANIO_MIN + " a " + ANIO_MAX + ")");
    return false;
}

public void getInfo() {  1 usage
    System.out.println("Título: " + titulo);
    System.out.println("Autor: " + autor);
    System.out.println("Año de publicación: " + anioPublicacion);

}}
```

4)

```
/*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.
 */
Gallina gal1 = new Gallina( id: 123,  edadInicial: 2,  huevosActuales: 4);
Gallina gal2 = new Gallina( id: 456,  edadInicial: 3,  huevosActuales: 2);

    gal1.mostrarEstado();
    gal2.mostrarEstado();

    gal1.ponerHuevo();
    gal1.envejecer();
    gal1.mostrarEstado();

    gal2.ponerHuevo();
    gal2.envejecer();
    gal2.mostrarEstado();
```

```
public class Gallina {  4 usages
    private int idGallina;  2 usages
    private int edad;  3 usages
    private int huevosPuestos;  3 usages

@Contract(pure = true)
public Gallina(int id, int edadInicial, int huevosActuales){  2 usages
    idGallina = id;
    edad = edadInicial;
    huevosPuestos = huevosActuales;
}

void ponerHuevo(){  2 usages
    huevosPuestos += 1;
}
void envejecer(){  2 usages
    edad += 1;
}
void mostrarEstado(){  4 usages
    System.out.println(" Id de Gallina: "+ idGallina);
    System.out.println("Edad: " + edad);
    System.out.println("Huevos Puestos: " + huevosPuestos);
}

}
```

5)

```
/* 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.
 */


        NaveEspacial nave = new NaveEspacial( n: "Andrómeda",  c: 50.0);

          nave.despegar();
          nave.avanzar( distanciaKm: 120);          // intenta avanzar sin recargar
          nave.recargarCombustible( cantidad: 40); // recarga correctamente
          nave.avanzar( distanciaKm: 120);          // ahora sí puede avanzar
          nave.mostrarEstado();       // estado final


    }
}
```

```
public class NaveEspacial {  2 usages
    String nombre;  3 usages
    double combustible;  9 usages
    final double LIMITE_COMBUSTIBLE = 100.0; // máximo permitido  2 usages
    final double CONSUMO_POR_KM = 0.5;        // unidades por km  1 usage

    @Contract(pure = true)
    NaveEspacial(String n, double c) {  1 usage
        nombre = n;
        combustible = c;
    }
    void despegar() {  1 usage
        System.out.println("La nave " + nombre + " ha despegado.");
    }
    void avanzar(double distanciaKm) {  2 usages
        double necesario = distanciaKm * CONSUMO_POR_KM;
        if (combustible >= necesario) {
            combustible = combustible - necesario;
            System.out.println("✈Avanzó " + distanciaKm + " km. Combustible consumido: " + necesario);
        } else {
            System.out.println("No hay suficiente combustible para avanzar " + distanciaKm + " km.");
        }
    }
    void recargarCombustible(double cantidad) {  1 usage
        if (combustible + cantidad <= LIMITE_COMBUSTIBLE) {
            combustible = combustible + cantidad;
            System.out.println("Recargó " + cantidad + " unidades. Combustible actual: " + combustible);
        } else {
            System.out.println("⚠No se puede recargar " + cantidad + " unidades. Superaría el límite de " + LIMITE_COMBUSTIBLE);
        }
    }

    void mostrarEstado() {  1 usage
        System.out.println("Estado de la nave " + nombre + ": Combustible disponible = " + combustible + " unidades");
    }
}
```