TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

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

Autor: Federico Iacono
Comisión: 6

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.

MARCO TEÓRICO

Clases y Objetos	Modelado de entidades como Estudiante, Mascota, Libro, Gallina y NaveEspacial
Atributos y Métodos	Definición de propiedades y comportamientos para cada clase
Estado e Identidad	Cada objeto conserva su propio estado (edad, calificación, combustible, etc.)
Encapsulamiento	Uso de modificadores de acceso y getters/setters para proteger datos

Modificadores de acceso Uso de private, public y protected para controlar visibilidad

Getters y Setters Acceso controlado a atributos privados mediante métodos

Reutilización de código Definición de clases reutilizables en múltiples contextos

TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

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.

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.

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.

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.

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.

CONCLUSIONES ESPERADAS

- Comprender la diferencia entre clases y objetos.
- Aplicar principios de encapsulamiento para proteger los datos.
- Usar getters y setters para gestionar atributos privados.
- Implementar métodos que definen comportamientos de los objetos. ● Manejar el estado y la identidad de los objetos correctamente. ● Aplicar buenas prácticas en la estructuración del código orientado a objetos. ● Reforzar el pensamiento modular y la reutilización del código en Java.

1. Clase Estudiante

```
public class Estudiante {

    private String nombre;
    private String apellido;
    private String curso;
    private double calificacion;

    public String getNombre() {
        return nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public String getCurso() {
        return curso;
    }

    public double getCalificacion() {
        return calificacion;
    }

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

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

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

    public void setCalificacion(double calificacion) {
        // Validación para que la calificación no sea negativa
        if (calificacion >= 0) {
            this.calificacion = calificacion;
        } else {
            System.out.println("Error: La calificación no puede ser negativa.");
        }
    }
```

```
    // Métodos
    public void mostrarInfo() {
        System.out.println("Información del Estudiante:");
        System.out.println("Nombre: " + getNombre());
        System.out.println("Apellido: " + getApellido());
        System.out.println("Curso: " + getCurso());
        System.out.println("Calificación: " + getCalificacion());
    }

    public void subirCalificacion(double puntos) {
        if (puntos > 0) {
            this.calificacion += puntos;
            System.out.println("La calificación ha subido en " + puntos + " puntos.");
            mostrarInfo();
        } else {
            System.out.println("Error: Los puntos a subir deben ser un valor
positivo.");
        }
    }

    public void bajarCalificacion(double puntos) {
        if (puntos > 0) {
            // Se asegura que la calificación no baje de 0
            if (this.calificacion - puntos >= 0) {
                this.calificacion -= puntos;
                System.out.println("La calificación ha bajado en " + puntos + "
puntos.");
                mostrarInfo();
            } else {
                this.calificacion = 0;
                System.out.println("La calificación no puede ser menor a 0. Se ha
establecido en 0.");
                mostrarInfo();
            }
        } else {
            System.out.println("Error: Los puntos deben ser un valor positivo.");
        }
    }
}
```

Clase Main

```
public class Main {
    public static void main(String[] args) {
        // 1. Instanciar a un estudiante
        Estudiante estudiante1 = new Estudiante();

        // Usar los setters para inicializar sus atributos
        estudiante1.setNombre("Ana");
```

```
        estudiante1.setApellido("López");
        estudiante1.setCurso("Matemáticas");
        estudiante1.setCalificacion(8.5);

        // 2. Mostrar su información
        System.out.println("Estado Inicial:");
        estudiante1.mostrarInfo();
        System.out.println("--------------------");

        // 3. Aumentar y disminuir calificaciones
        System.out.println("Aumentando calificación...");
        estudiante1.subirCalificacion(1.0);
        System.out.println("--------------------");

        System.out.println("Disminuyendo calificación...");
        estudiante1.bajarCalificacion(2.0);
        System.out.println("--------------------");
    }
}
```

2. Clase Mascota

```
public class Mascota {

    private String nombre;
    private String especie;
    private int edad;

    public String getNombre() {
        return nombre;
    }

    public String getEspecie() {
        return especie;
    }

    public int getEdad() {
        return edad;
    }

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

    public void setEspecie(String especie) {
        this.especie = especie;
    }

    public void setEdad(int edad) {
        // Validación para que la edad no sea negativa
```

```
        if (edad >= 0) {
            this.edad = edad;
        } else {
            System.out.println("Error: La edad no puede ser negativa.");
        }
    }

    // Métodos
    public void mostrarInfo() {
        System.out.println("Información de la Mascota:");
        System.out.println("Nombre: " + getNombre());
        System.out.println("Especie: " + getEspecie());
        System.out.println("Edad: " + getEdad() + " años");
    }

    public void cumplirAnios() {
        // Aumenta la edad de la mascota en 1
        this.edad += 1;
        System.out.println("¡Feliz cumpleaños, " + getNombre() + "! Ahora tiene "
+ getEdad() + " años.");
    }
}
```

Clase Main

```
public class Main {
    public static void main(String[] args) {
        // Crear una mascota
        Mascota miMascota = new Mascota();

        // Usar los setters para inicializar sus atributos
        miMascota.setNombre("Max");
        miMascota.setEspecie("Perro");
        miMascota.setEdad(3);

        // Mostrar su información inicial
        System.out.println("Estado inicial de la mascota:");
        miMascota.mostrarInfo();
        System.out.println("--------------------");

        // Simular el paso del tiempo y verificar
        System.out.println("Simulando el paso de un año...");
        miMascota.cumplirAnios();
        System.out.println("--------------------");

        // Mostrar la información actualizada para verificar el cambio
        System.out.println("Estado actualizado de la mascota:");
        miMascota.mostrarInfo();
        System.out.println("--------------------");
    }
```

```
}
```

3. **Clase Libro**

```
public class Libro {

    private String titulo;
    private String autor;
    private int anoPublicacion;

    public String getTitulo() {
        return titulo;
    }

    public String getAutor() {
        return autor;
    }

    public int getAnoPublicacion() {
        return anoPublicacion;
    }

    public void setAnoPublicacion(int anoPublicacion) {
        // Validación: el año no puede ser en el futuro ni antes del 1000 d.C.
        if (anoPublicacion <= 2025 && anoPublicacion >= 1000) {
            this.anoPublicacion = anoPublicacion;
            System.out.println("Año de publicación actualizado a " +
anoPublicacion);
        } else {
            System.out.println("Error: El año de publicación no es válido. Debe
ser entre 1000 y 2025.");
        }
    }

    // Métodos setters para inicializar los atributos
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

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

    // Método para mostrar la información del libro
    public void mostrarInfo() {
        System.out.println("Información del Libro:");
        System.out.println("Título: " + getTitulo());
        System.out.println("Autor: " + getAutor());
        System.out.println("Año de Publicación: " + getAnoPublicacion());
    }
}
```

Clase Main

```
public class Main {
    public static void main(String[] args) {
        // Crear una instancia de Libro y establecer sus atributos
        Libro miLibro = new Libro();
        miLibro.setTitulo("Cien Años de Soledad");
        miLibro.setAutor("Gabriel García Márquez");
        miLibro.setAnoPublicacion(1967); // Año de publicación inicial

        // Mostrar la información inicial del libro
        System.out.println("Estado Inicial:");
        miLibro.mostrarInfo();
        System.out.println("--------------------");

        // Tarea: Intentar modificar el año con un valor inválido
        System.out.println("Intentando modificar el año a 2030 (inválido)...");
        miLibro.setAnoPublicacion(2030); // Este valor será rechazado por la
validación
        System.out.println("--------------------");

        // Intentar modificar el año con un valor válido
        System.out.println("Intentando modificar el año a 1982 (válido)...");
        miLibro.setAnoPublicacion(1982); // Este valor será aceptado
        System.out.println("--------------------");

        // Mostrar la información final para verificar los cambios
        System.out.println("Estado Final:");
        miLibro.mostrarInfo();
        System.out.println("--------------------");
    }
}
```

4. **Clase Gallina**

```
public class Gallina {

    private int idGallina;
    private int edad;
    private int huevosPuestos;

    public void setIdGallina(int idGallina) {
        this.idGallina = idGallina;
    }

    public void setEdad(int edad) {
        if (edad >= 0) {
            this.edad = edad;
        } else {
            System.out.println("Error: La edad no puede ser negativa.");
        }
```

```
    }

    public void setHuevosPuestos(int huevosPuestos) {
        if (huevosPuestos >= 0) {
            this.huevosPuestos = huevosPuestos;
        } else {
            System.out.println("Error: El número de huevos no puede ser
negativo.");
        }
    }

    // Getters
    public int getIdGallina() {
        return idGallina;
    }

    public int getEdad() {
        return edad;
    }

    public int getHuevosPuestos() {
        return huevosPuestos;
    }

    // Métodos
    public void ponerHuevo() {
        this.huevosPuestos++;
        System.out.println("La gallina " + idGallina + " ha puesto un huevo.
Total: " + huevosPuestos);
    }

    public void envejecer() {
        this.edad++;
        System.out.println("La gallina " + idGallina + " ha envejecido. Ahora
tiene " + edad + " años.");
    }

    public void mostrarEstado() {
        System.out.println("--- Estado de la Gallina " + idGallina + " ---");
        System.out.println("ID: " + getIdGallina());
        System.out.println("Edad: " + getEdad() + " años");
        System.out.println("Huevos puestos: " + getHuevosPuestos());
    }
}
```

Clase Main

```
public class Main {
    public static void main(String[] args) {
        // Crea dos gallinas
        Gallina gallina1 = new Gallina();
```

```
        gallina1.setIdGallina(1);
        gallina1.setEdad(1);
        gallina1.setHuevosPuestos(5);

        Gallina gallina2 = new Gallina();
        gallina2.setIdGallina(2);
        gallina2.setEdad(2);
        gallina2.setHuevosPuestos(10);

        System.out.println("--- Estado inicial de las gallinas ---");
        gallina1.mostrarEstado();
        System.out.println();
        gallina2.mostrarEstado();
        System.out.println("---------------------------------------");

        // Simular acciones
        System.out.println("Simulando acciones...");
        gallina1.envejecer();
        gallina1.ponerHuevo();
        gallina1.ponerHuevo();

        System.out.println();

        gallina2.ponerHuevo();
        gallina2.ponerHuevo();
        gallina2.envejecer();
        System.out.println("---------------------------------------");

        // Mostrar estado final
        System.out.println("--- Estado final de las gallinas ---");
        gallina1.mostrarEstado();
        System.out.println();
        gallina2.mostrarEstado();
        System.out.println("---------------------------------------");
    }
}
```

5. **Clase NaveEspacial**

```
public class NaveEspacial {

    private String nombre;
    private double combustible;
    private final double CAPACIDAD_MAXIMA = 100.0; // Constante para el
límite de combustible

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
```

```
        this.nombre = nombre;
    }

    public double getCombustible() {
        return combustible;
    }

    public void setCombustible(double combustible) {
        if (combustible >= 0 && combustible <= CAPACIDAD_MAXIMA) {
            this.combustible = combustible;
        } else {
            System.out.println("Error: Cantidad de combustible inicial inválida.");
        }
    }

    // Métodos
    public void despegar() {
        if (this.combustible >= 10) { // Se asume que despegar consume 10
unidades
            this.combustible -= 10;
            System.out.println("¡Despegue exitoso! Consumo de 10 unidades de
combustible.");
        } else {
            System.out.println("No hay suficiente combustible para el
despegue.");
        }
    }

    public void avanzar(double distancia) {
        double consumo = distancia / 10; // Se asume que cada 10 km
consume 1 unidad
        if (this.combustible >= consumo) {
            this.combustible -= consumo;
            System.out.println("La nave " + nombre + " ha avanzado " +
distancia + " unidades. Combustible restante: " + String.format("%.2f",
this.combustible));
        } else {
            System.out.println("¡Alerta! No hay suficiente combustible para
avanzar " + distancia + " unidades.");
        }
    }

    public void recargarCombustible(double cantidad) {
        if (this.combustible + cantidad <= CAPACIDAD_MAXIMA) {
            this.combustible += cantidad;
            System.out.println("Recarga exitosa. Se añadieron " + cantidad + "
unidades. Combustible total: " + String.format("%.2f", this.combustible));
        } else {
            System.out.println("Error: La cantidad a recargar excede la
capacidad máxima de " + CAPACIDAD_MAXIMA + " unidades.");
        }
```

```
    }

    public void mostrarEstado() {
        System.out.println("--- Estado de la Nave Espacial ---");
        System.out.println("Nombre de la nave: " + getNombre());
        System.out.println("Combustible actual: " + String.format("%.2f",
getCombustible()));
        System.out.println("---------------------------------");
    }
}
```

Clase Main

```
public class Main {
    public static void main(String[] args) {
        // Crea una nave con 50 unidades de combustible
        NaveEspacial nave = new NaveEspacial();
        nave.setNombre("Apolo 8");
        nave.setCombustible(50.0);

        // Mostrar estado inicial
        System.out.println(">>> Estado inicial de la nave:");
        nave.mostrarEstado();

        // Intentar avanzar una distancia que consuma más de 50 unidades
        System.out.println(">>> Intentando avanzar una distancia de 600...");
        nave.avanzar(600);
        System.out.println("---------------------------------");
        nave.mostrarEstado();

        // Recargar combustible hasta el límite
        System.out.println(">>> Recargando combustible con 55 unidades...");
        nave.recargarCombustible(55.0); // Excede el límite de 100
        System.out.println("---------------------------------");

        System.out.println(">>> Recargando combustible con 50 unidades...");
        nave.recargarCombustible(50.0);
        System.out.println("---------------------------------");

        // Avanzar después de la recarga
        System.out.println(">>> Intentando avanzar una distancia de 800...");
        nave.avanzar(800);
        System.out.println("---------------------------------");

        // Mostrar el estado final
        System.out.println(">>> Estado final de la nave:");
        nave.mostrarEstado();
    }
}
```