UNIVERSIDAD TECNOLOGICA NACIONAL

27-8-2025

# PROGRAMACIÓN II

Introducción a POO


**Alumno:**

- Ezequiel Alejandro Ventura


**Unidad:** 3

**Coordinador:** Carlos Martinez

**Profesor:** Ariel Enferrel

**Tutor:** Tomás Ferro

**Github:**

https://github.com/equimdq/Programacion2

## Contenido



# 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

| Concepto | Aplicación en el proyecto |
|---|---|
| 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 |

# Caso Práctico

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

## Ejercicio 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.

Atributos

```java
package Ejercicio01;


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

Constructor

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

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;
}

public void setCalificacion(double calificacion) {
    this.calificacion = calificacion;
}
```

Métodos

```
    // Métodos
    public void mostrarInfo() {
        System.out.println("Estudiante: " + nombre + " " + apellido);
        System.out.println("Curso: " + curso);
        System.out.println("Calificación: " + calificacion);
        System.out.println("--------------------------");

    }

    public void subirCalificacion(double puntos) {
        this.calificacion += puntos;
    }

    public void bajarCalificacion(double puntos) {
        this.calificacion -= puntos;
        if (this.calificacion < 0) {
            this.calificacion = 0; // Evitar calificación negativa
        }
    }


}
```

Main

```
package Ejercicio01;


public class Main1 {


    public static void main(String[] args) {


        Estudiante estudiante1 = new Estudiante("Jose", "Manna", "Programación I", 7.2);

        //Mostrar información inicializada
        estudiante1.mostrarInfo();

        //Subir calificación
        estudiante1.subirCalificacion(2.5);
        estudiante1.mostrarInfo();

        //Bajar calificación
        estudiante1.bajarCalificacion(1);
        estudiante1.mostrarInfo();

    }

}
```

## Ejercicio 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.

Atributos

```java
package ejercicio02;


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

Constructor y getters/setters

```java
//Constructor

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

}

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;
}

public void setEdad(int edad) {
    this.edad = edad;
}
```

Metodo y tooString

```java
//Métodos

@Override
public String toString() {
    return "Mascota{" + "nombre=" + nombre + ", especie=" + especie + ", edad=" + edad + '}';
}

public void cumplirAnios(int anios) {
    this.edad += anios;

    System.out.println("Se incrementó la edad en " + anios + " año(s).");


}
```

Main

```java
package ejercicio02;


public class Main2 {


    public static void main(String[] args) {
        Mascota mascota1 = new Mascota("Patan", "Perro", 3);
        System.out.println(mascota1);
        mascota1.cumplirAnios(1); // edad a sumar
        System.out.println(mascota1); //muestra la edad incrementada

    }

}
```

## Ejercicio 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.

Clase Libro

```java
private String titulo;
private String autor;
private int añoPublicacion;

// Constructor

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


}

public String getTitulo() {
    return titulo;
}

public String getAutor() {
    return autor;
}

public int getAñoPublicacion() {
    return añoPublicacion;
}

public void setAñoPublicacion(int añoPublicacion) {
    if (añoPublicacion > 868 && añoPublicacion <= 2025) {
    this.añoPublicacion = añoPublicacion;
    }   else {
        System.out.println("El año de publicación debe ser posterior al 868.");
    }
// Metodo
}

@Override
public String toString() {
    return "Libro{" + "titulo=" + titulo + ", autor=" + autor + ", a\u00f1oPublicacion=" + añoPublicacion + '}';
```

Main

```java
package Ejercicio03;

public class Main3 {


    public static void main(String[] args) {
      Libro miLibro2 = new Libro("Rayuela", "Cortazar", 2020);
      System.out.println(miLibro2);

      miLibro2.setAñoPublicacion(500); // validación

      miLibro2.setAñoPublicacion(1963); // Año válido
    }

}
```

## Ejercicio 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.

Clase Gallina

```java
package Ejercicio04;


public class Gallina {
    private int idGallina;
    private int edad;
    private int huevosPuestos;

    //Constructor

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

    // Metodo
    public void ponerHuevo() {
        huevosPuestos ++; // incrementa los huevos puestos
        System.out.println("La gallina " + idGallina + " puso " + huevosPuestos + " huevo/s.");
    }

    public void envejecer() {
        edad ++; // incrementa la edad
        System.out.println("La edad de la gallina " + idGallina +" es " + edad + " año/s");
    }

    @Override
    public String toString() {
        return "Gallina{" + "idGallina=" + idGallina + ", edad=" + edad + ", huevosPuestos=" + huevosPuestos + '}';
    }
```

Main

```java
package Ejercicio04;

public class Main4 {


    public static void main(String[] args) {
        Gallina g1 = new Gallina(1, 0, 0); // En este caso incilizamos los valores de edad y huevos en 0
        Gallina g2 = new Gallina(2, 3, 5); // En este caso los inicializamos en 3 y 5 respectivamente

        //Incrementos de edad y huevos para la primera gallina

        g1.envejecer();
        g1.ponerHuevo();
        g1.ponerHuevo();

        //Incrementos de edad y huevos para la segunda gallina

        g2.envejecer();
        g2.envejecer();
        g2.ponerHuevo();
```

## Ejercicio 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.

Atributos y Constructor

```java
package Ejercicio05;

public class NaveEspacial {
    private String nombre;
    private int combustible;
    private boolean enVuelo;
    private boolean fueRecargada; // validación de recarga antes de despegue
    private final int CAPACIDAD_MAXIMA = 100;

    // Constructor

    public NaveEspacial(String nombre, int combustible) {
        this.nombre = nombre;
        this.combustible = combustible;
        this.enVuelo = false;
        this.fueRecargada = false;

    }
```

Métodos y tooString

```java
    // Métodos

    public void despegar() {
        if (!fueRecargada) {
        System.out.println("La nave no puede despegar: debe recargar combustible antes.");
        return;
    }

        if (combustible >= 10) {
        combustible -= 10;
        enVuelo = true;
        System.out.println(nombre + " ha despegado.");
    }       else {
        System.out.println("Combustible insuficiente para despegar.");
    }
}



    public void avanzar(int distancia) {
        int consumo = distancia * 3; // reglaje de consumo de combustible por KMs. recorridos
        if (!enVuelo) { //validamos que no esté en vuelo
            System.out.println("La nave no ha despegado todavía.");
            return;
        } // validamos que tenga combustible suficiente para poder avanzar
        if (combustible >= consumo) {
            combustible -= consumo;
            System.out.println(nombre + " avanzó " + distancia + " KM.");
        } else {
            System.out.println("Combustible insuficiente para avanzar.");
        }
    }
```

```java
    public void recargarCombustible(int cantidad) {
    if (combustible + cantidad <= CAPACIDAD_MAXIMA) {
        combustible += cantidad;
        fueRecargada = true;
        System.out.println("Se recargaron " + cantidad + " litros. Combustible actual: " + combustible + " litros.");
    } else {
        combustible = CAPACIDAD_MAXIMA;
        fueRecargada = true;
        System.out.println("Se llenó el tanque al máximo (" + CAPACIDAD_MAXIMA + ").");
    }
}


    public void mostrarEstado() {
    System.out.println("\nESTADO DE LA NAVE");
    System.out.println("-----------------");
    System.out.println("Nave: " + nombre);
    System.out.println("Combustible: " + combustible);
    System.out.println("¿En vuelo?: " + (enVuelo ? "Sí" : "No"));
}

    // Usamos tooString para simplificar
    @Override
    public String toString() {
        return "NaveEspacial{" +
                "nombre='" + nombre + '\'' +
                ", combustible=" + combustible +
                ", enVuelo=" + enVuelo +
                '}';
    }
}
```

Main

```java
package Ejercicio05;

public class Main5 {
    public static void main(String[] args) {
        NaveEspacial nave = new NaveEspacial("Apolo I", 50);

        nave.despegar(); // Intento fallido de despegue
        nave.recargarCombustible(50); // Recarga de combustible al máximo (100 lts.)
        nave.despegar(); // Reeintento de despegue
        nave.avanzar(15); // Consume 45 (15 lts * 3 km)
        nave.mostrarEstado(); // Detalle del estado
        System.out.println(nave);
    }
}
```