AF9, Programación orientada a objetos**.**


UANL

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN


FIME

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Lenguajes de Programación.

Ing. Karla Patricia Uribe Sierra.

Grupo 005.

M5.

Semestre enero – junio 2025.

Matthew Alejandro Martínez Zambrana.

2223170.

IAS.

San Nicolás de los Garza, N.L, viernes 23 de mayo de 2025.

**Introducción.**

La Programación Orientada a Objetos (POO) es un paradigma de programación que organiza el software en torno a objetos, los cuales son representaciones de entidades del mundo real. Cada objeto combina datos (atributos) y comportamientos (métodos o funciones), facilitando una programación más estructurada, modular y reutilizable.

En este enfoque, los programas se construyen usando los siguientes principios fundamentales:

- **Clases y Objetos**: Una clase es un molde o plantilla que define las características y comportamientos de un objeto. Un objeto es una instancia concreta de una clase.
- **Encapsulación**: Permite ocultar los detalles internos del objeto, protegiendo los datos y exponiendo solo lo necesario a través de métodos públicos.
- **Herencia**: Es la capacidad de una clase de heredar propiedades y métodos de otra, facilitando la reutilización del código y la creación de jerarquías.
- **Polimorfismo**: Permite que una misma operación actúe de manera diferente dependiendo del tipo de objeto que la ejecute. Esto se logra mediante funciones virtuales y punteros a clases base.
- **Abstracción**: Consiste en modelar objetos complejos ocultando los detalles innecesarios y mostrando solo la información relevante para el uso del objeto.

**Programa.**

```cpp
#include <iostream>
using namespace std;

class Persona {
protected:
    string nombre;
public:
    void setNombre(string n) {
        nombre = n;
    }
    virtual void mostrar() {
        cout << "Soy una persona y me llamo " << nombre << endl;
    }
};

class Estudiante : public Persona {
public:
    void mostrar() override {
        cout << "Soy un estudiante y me llamo " << nombre << endl;
    }
};

int main() {
    Persona* p;
    Estudiante e;
    string nombreUsuario;

    cout << "Ingresa tu nombre: ";
    getline(cin, nombreUsuario);

    e.setNombre(nombreUsuario);
    p = &e;
    p->mostrar();
    return 0;
}
```

```
input
Ingresa tu nombre: Matthew
Soy un estudiante y me llamo Matthew


...Program finished with exit code 0
Press ENTER to exit console.
```

Elementos aplicados en el código.

1. Encapsulación.

   El atributo nombre está protegido y no se puede modificar directamente desde fuera de la clase.

2. Herencia.

   La clase estudiante hereda de Persona y estudiante puede usar lo que tenga Persona.

3. Polimorfismo.

   El puntero p es de tipo Persona*, pero apunta a un objeto Estudiante. Al llamar p -> mostrar (); se ejecuta la version del método mostrar () de la clase Estudiante gracias al uso de virtual en la clase base.

4. Tipos de datos.

   String: Para guardar el nombre del usuario.

   Persona*: puntero a un objeto de tipo Persona.

   Estudiante: objeto de la clase derivada.