Programación II

Programación Orientada a Objetos (POO)

Introducción a los Paradigmas de Programación

Evolución de los paradigmas:

- **Programación estructurada:** Enfoque secuencial (instrucciones una detrás de otra)

Limitaciones en programas complejos:

- Bloques de código muy largos (miles de líneas)
- Código espagueti difícil de mantener
- Fragilidad ante errores (toda la aplicación se cae)
- Escalabilidad limitada

¿Necesitamos una mejor solución?

¿Qué es la Programación Orientada a Objetos?

Un nuevo paradigma:

- Traslada la realidad al código de programación
- Observa los problemas como objetos
- Clave en los lenguajes de programación modernos

Ventajas principales:

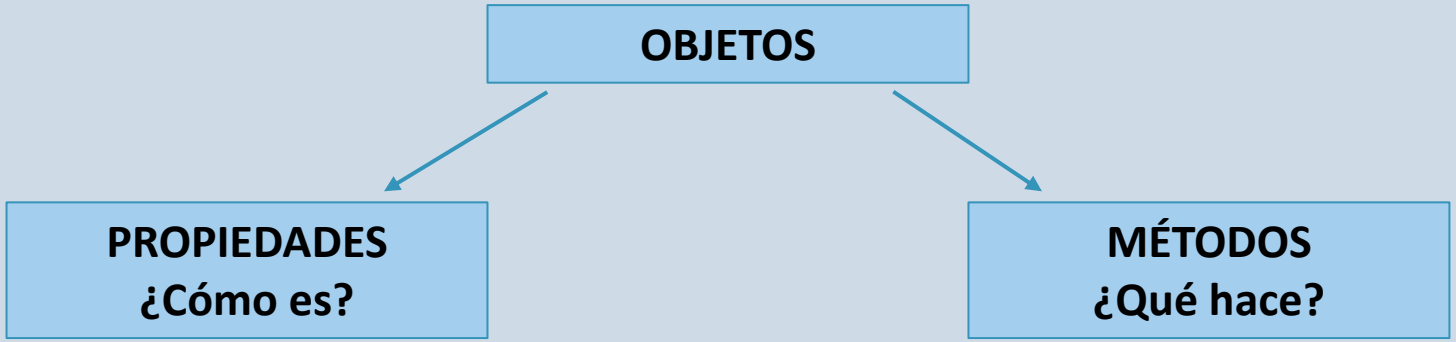
- Código organizado y modular
- Mayor mantenibilidad
- Mejor escalabilidad
- Reutilización de código

Objetos del Mundo Real vs. Digital

Analogías entre ambos mundos:

Mundo Físico	Mundo Digital
Auto	Ventana
Árbol	Usuario
Mesa	Botón
Televisor	Menú
Silla	Conexión

¿Qué es un Objeto?



¿Qué es un Objeto?

Definición: Abstracción de un objeto del mundo real.

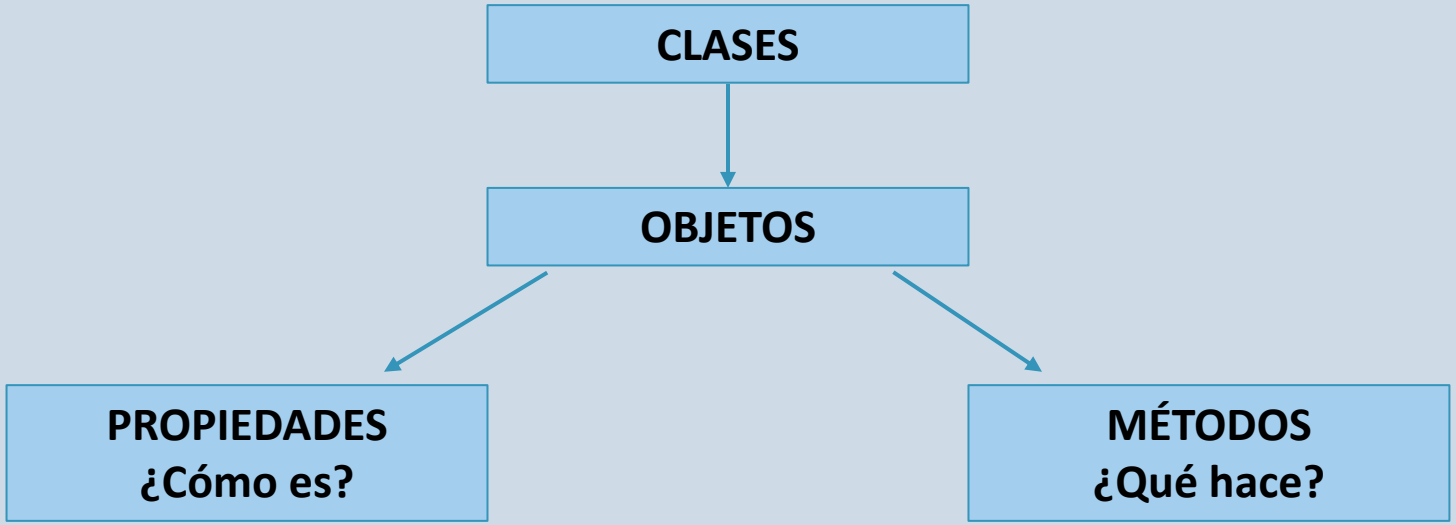
¿Pero que es la abstracción?

La abstracción: Enfocarse en los aspectos relevantes para interactuar con el objeto sin conocer todos los detalles de implementación.

Componentes esenciales de un objeto:

- **Propiedades (Características):** Atributos que describen el objeto. Cada propiedad tiene un **estado interno** que la define.
- **Métodos (Acciones):** Acciones que puede realizar.
- **Identidad**: Nombre que le vamos a la variable, que identifica objeto.

Clases - Molde de los Objetos



Clases - Molde de los Objetos

Definición: Plantilla o molde que define:

- Características (propiedades)
- Comportamiento (métodos)

Relación clase-objeto:

- La clase es el plano o molde
- El objeto es la instancia concreta

Ejemplo: La clase "Ventana" define propiedades como ancho, alto, título; y métodos como abrir() y cerrar().

Clase "Ventana"

```cpp
#include <iostream>
#include <string>

class Ventana {
private:
    // Propiedades (privadas)
    int ancho;
    int alto;
    std::string titulo;

public:
    // Constructor (público)
    Ventana(int ancho, int alto, std::string titulo) {
        this->ancho = ancho;
        this->alto = alto;
        this->titulo = titulo;
    }

    // Métodos (públicos)
    void abrir() {
        std::cout << "Ventana abierta: " << titulo << std::endl;
    }

    void cerrar() {
        std::cout << "Ventana cerrada: " << titulo << std::endl;
    }

    // Getters
    int getAncho() const { return ancho; }
    int getAlto() const { return alto; }
    std::string getTitulo() const { return titulo; }

    // Setters
    void setAncho(int nuevoAncho) { ancho = nuevoAncho; }
    void setAlto(int nuevoAlto) { alto = nuevoAlto; }
    void setTitulo(std::string nuevoTitulo) { titulo = nuevoTitulo; }
};
```

```cpp
int main() {
    Ventana ventana1(800, 600, "Mi Ventana");
    ventana1.abrir();
    ventana1.cerrar();

    return 0;
}
```

Los Cuatro Pilares de la POO

Fundamentos conceptuales:

- **Abstracción** - Modelar aspectos relevantes de la realidad
- **Encapsulamiento** - Ocultamiento del estado interno
- **Herencia** - Mecanismo de especialización entre clases
- **Polimorfismo** - Múltiples formas para un mismo método

Encapsulamiento

Encapsulamiento: Principio de ocultamiento de información.

Dentro de la clase las propiedades y métodos se ubican en lugares distintos:

- las propiedades luego de **private**:
- y los métodos luego de **public**:

private y **public** son los especificadores de acceso, y establecen que será o no accesible fuera de la clase:

- private: (privado) establece que los miembros de la clase definidos son privados, lo que significa que no serán accesibles fuera de la clase.
- public: (público) establece que los miembros de la clase definidos son públicos, lo que significa que serán accesibles fuera de la clase.

Constructores

Métodos especiales para inicialización:

- Mismo nombre que la clase
- No devuelven valor (ni siquiera void)
- Se ejecutan automáticamente al crear objetos
- Permiten asignar valores iniciales a las propiedades

Ventaja: Garantizan que los objetos se creen en un estado válido, ya que nos posibilitan asignarles valores explícitos a las propiedades de los objetos al momento de la declaración de estos.

Beneficios de la POO

Ventajas sobre programación estructurada:

- Código mejor organizado y mantenible
- Mayor reutilización mediante herencia
- Menos propenso a errores
- Escalabilidad mejorada
- Modelado más natural de problemas reales

Conclusión

Resumen:

- La POO representa un avance fundamental en la programación
- Permite modelar problemas complejos de manera más natural
- Los cuatro pilares (abstracción, encapsulamiento, herencia, polimorfismo) proporcionan solidez conceptual
- Es esencial en el desarrollo de software moderno