Programación Orientada a Objetos

Dra. Maricela Claudia Bravo Contreras

mari_clau_18@hotmail.com

Introducción al Paradigma de Programación Orientada a Objetos

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco

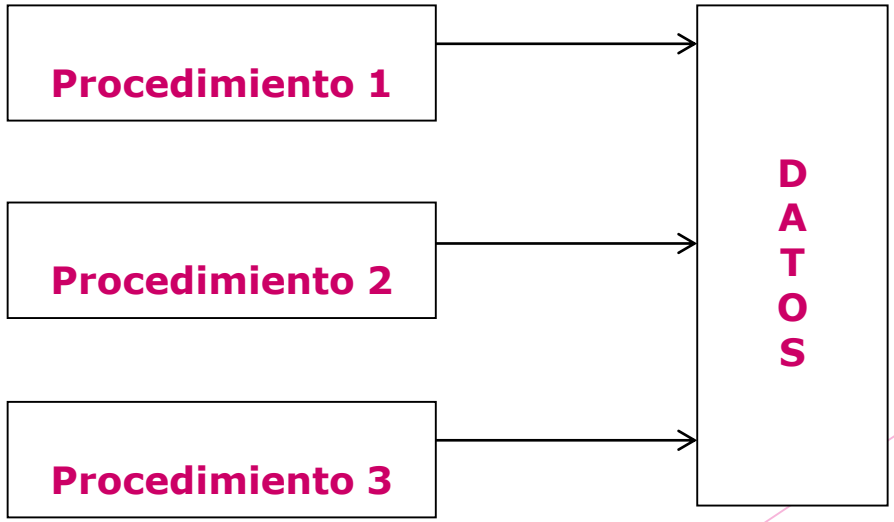
¿Qué es un paradigma?

- **Paradigma de programación**
  - Los paradigmas indican las diversas formas que han sido aceptadas como estilos para programar y para resolver los problemas por medio de una computadora.

- **Paradigmas**
  - Programación Procedimental
  - Programación Funcional
  - Programación Lógica
  - Programación Orientada a Objetos

Programación Procedimental

- Es considerado el más común
- El programador se concentra en el procesamiento, en el algoritmo requerido para llevar a cabo el cómputo deseado.
- Fortran es el lenguaje de procedimientos original; los lenguajes como C, Basic o Pascal surgieron posteriormente con el mismo paradigma.
- La programación estructura se considera como el componente principal de la programación procedimental.



```vb
Module Module1

    Sub Main()

        'Bloque de codigo donde declaro las variables
        Dim a As Integer, b As Integer
        Dim suma, resta, producto, div As Double
        Dim modulo As Double

        'Bloque de codigo para realizar los procesos
        'Variables que reciben valores ingresados por el usuario
        a = 20
        b = 50

        suma = a + b

        'Variables que reciben valores dentro del código
        suma = a + b
        resta = a - b
        producto = a * b
        div = a / b
        modulo = a Mod b

        'Bloque de codigo para mostrar resultados
        Console.WriteLine("*** OPERACIONES MATEMATICAS ***\n")
        Console.WriteLine("Suma: {0}", suma)
        Console.WriteLine("Resta: ", resta)
        Console.WriteLine("Producto: ", producto)
        Console.WriteLine("Divicion: ", div)
        Console.WriteLine("Resto div: ", modulo)
        Console.WriteLine("Operaciones realizadass con exito")
```

```
C     PROGRAMA PARA CALCULAR FACTORIALES
      READ(5,100) N
  100 FORMAT(I5)
      NFAC=1
  500 CONTINUE
      IF (N) 1000, 1000, 600
  600 CONTINUE
      NFAC=NFAC*N
      N=N-1
      GOTO 500
 1000 WRITE(6,110) NFAC
  110 FORMAT(I10)
      STOP
      END
```

```
program factorial(input,output);
var
  n:integer;

function fact(m:integer):integer;
begin
  if m>0 then
    fact:=m*fact(m-1)
  else
    fact:=1;
end;
```

Programación Procedimental

Programación Funcional

Es un paradigma de programación declarativa basado en la utilización de funciones aritméticas que no maneja datos mutables o de estado.

La programación funcional tiene sus raíces en el cálculo lambda.

Los lenguajes de programación puramente funcionales son utilizados principalmente en ambientes académicos y no en el desarrollo de software comercial.

Algunos lenguajes de programación funcional son: Scheme, Erlang, Rust, Objective Caml y Haskell.

(sumaE xs) es la suma de los números de xs. Por ejemplo,

```
sumaE [2,3,5]  ⇝  10
```

```
sumaE :: [Int] -> Int
sumaE xs = sumaE' 0 xs

sumaE' :: Int -> [Int] -> Int
sumaE' v []     = v
sumaE' v (x:xs) = (sumaE' $! (v+x)) xs
```

Programación Funcional

Programación Lógica

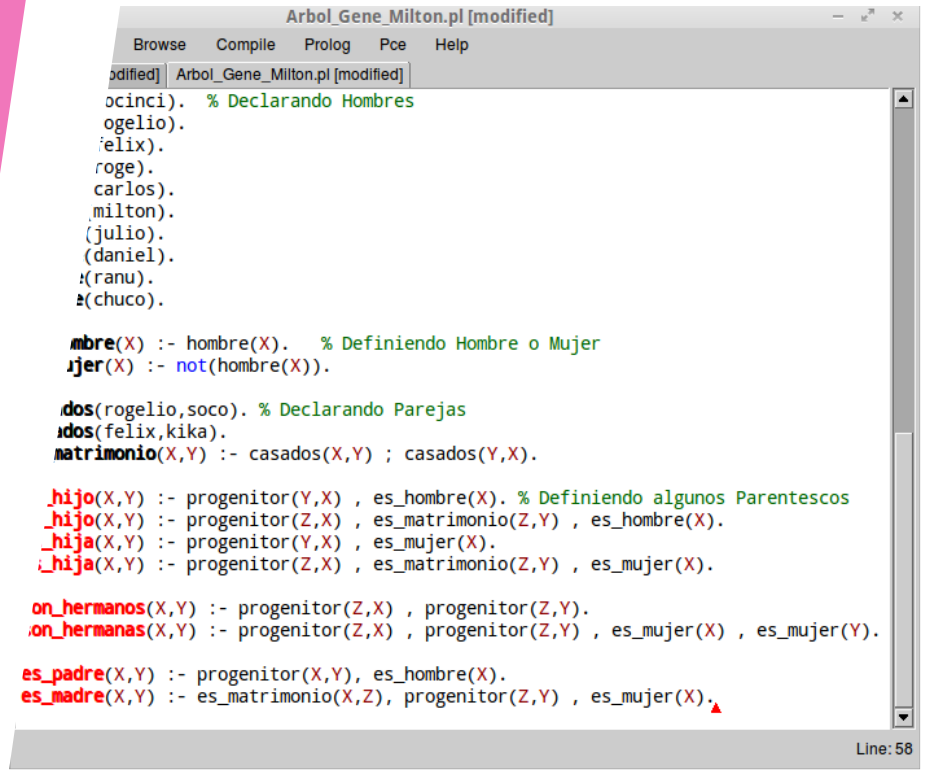
Es un tipo de paradigma de programación dentro de la programación declarativa.

Se basa en el concepto de predicado, o relación entre elementos.

Se utiliza en aplicaciones de inteligencia artificial o relacionadas, como son:

- sistemas expertos,
- demostración automática de teoremas, y
- reconocimiento de lenguaje natural.

Ejemplos de lenguajes que pertenecen a este paradigma de programación son: Prolog y LISP.




```
.sp
 Guardar valores como una lista de caracteres
lefine (SumarSiguiente V)
    (cond ((null V) (progn (print "Suma=") 0))
          (T  (+ ( SumarSiguiente (cdr V)) (car V)) ) ))
ARSIGUIENTE
 Crear vector de valores de entrada
efun ObtenerEntrada(f c)
        (cond ((eq c 0) nil)
        (T (cons (read f) (ObtenerEntrada f (- c 1))))))
ENERENTRADA
lefun Hazlo()
        (progn
        ( setq archivoent (open '"lisp.data"))
        ( setq arreglo (ObtenerEntrada archivoent (read archiv
        (print arreglo)
        (print (SumarSiguiente arreglo))))
LO
azlo

 2 3 4)
uma="
```

Programación Lógica

Paradigma de Programación Orientada a Objetos

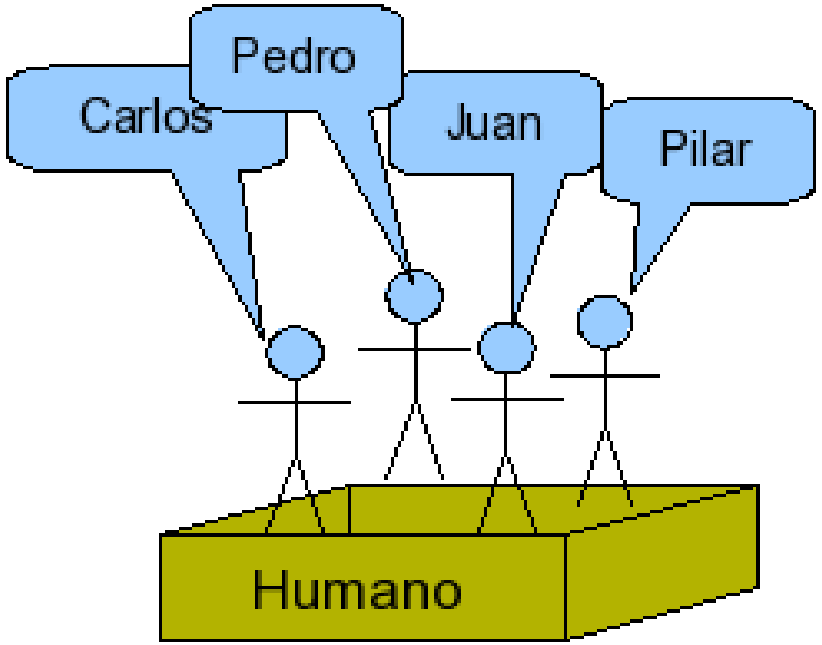
Programación Orientada a Objetos

Facilita la ***creación*** de software de calidad pues sus características facilitan:

- La mantenibilidad
- La extensibilidad
- La reutilización del software
- La usabilidad del software

La POO se basa en la ***idea natural*** de un mundo lleno de ***objetos*** y que la resolución de problemas se realiza mediante el modelo de objetos.

Los conceptos de clase y objeto son los más importantes de la POO.



Objeto y Clase

- **Objeto** es cualquier cosa tangible o intangible que se pueda imaginar definida mediante sus atributos y las operaciones que permiten modificar dichos atributos.

- Una **clase** es una plantilla que permite definir un conjunto de objetos.

Programación Orientada a Objetos

Es un paradigma de programación que usa los objetos en sus interacciones para diseñar aplicaciones y programas.

Se fundamenta en los siguientes principios:

Abstracción	Encapsulamiento	Modularidad	Herencia	Polimorfismo

Se popularizó a principios de la década de los años 1990.

La POO trata de aproximarse al modo de actuar y pensar del hombre y no al de la máquina.

Conceptos Fundamentales de la POO

Clases

Atributos / Variables

Métodos

Objetos

Clases

Clases

Son abstracciones que representan a un conjunto de objetos con un:

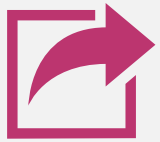
- Comportamiento
- Interfaz común

Es la implementación de un **tipo de dato** (considerando los objetos como instancias de las clases)

Permiten definir y representar colecciones de objetos

Proveen un modelo para la creación de objetos

Elementos de las Clases



Atributos. Representan el estado de un objeto

Métodos. Representan el comportamiento de un objeto (funciones miembro)

Atributos y variables de una Clase

Una variable es un espacio físico de memoria donde un programa puede almacenar un datos para su posterior utilización.

Atributos de una clase

Los atributos de una clase son definidos según esta sintaxis:

[modifVisibilidad] [modifAtributo] tipo ***nombreVariable*** *[= valorInicial] ;*

nombreVariable es el nombre de la variable

los nombres de las variables empiezan con una letra minúscula	los nombres de las clases empiezan con una letra mayúscula	el nombre de variable no debe tener el mismo nombre que otras variables.

Atributos de una clase

Los atributos de una clase son definidos según esta sintaxis:

[modifVisibilidad] [modifAtributo] ***tipo*** *nombreVariable [= valorInicial] ;*

tipo es el tipo de datos de la variable

Los tipos de datos de una variable pueden ser:

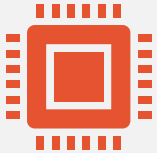
- Tipos primitivos
- Tipo objeto.

Tipos de datos Primitivos

Los tipos primitivos son aquellos tipos más usuales y básicos. Son los habituales de otros lenguajes de programación.

boolean: No es un valor numérico, solo admite los valores true o false.

char: Cada carácter ocupa 16 bits.



Tipos entero:

byte: 1 byte.

short: 2 bytes.

int: 4 bytes.

long: 8 bytes.

Tipos reales o flotante:

float: 4 bytes.

double: 8 bytes.

Tipos de Datos Primitivos y Variables

- boolean, char, byte, short, int, long, float, double etc.
- Estos tipos de datos primitivos **NO** son objetos.
- Esto significa que no se puede utilizar el operador **new** para crear una instancia de objeto para una variable.
- Declaración de variables con tipos primitivos:
  - `float valorInicial;`
  - `int valorRetorno, indice = 2;`
  - `double gamma = 1.2;`
  - `boolean valueOk = false;`

Declaraciones

```
int a ; // declaración de una variable a inicializada a 0

int b = 8; // declaración de una variable b inicializada a 8

Persona per1; // declaración de una variable per1 preparada para un objeto de la clase Persona

Persona per2; // lo mismo que en la variable anterior

per2 = new Persona(); // se crea un nuevo objeto de la clase Persona, y es asignado a la variable per2

per2 = per1; // Ahora también per2 tiene el mismo objeto a su cargo que per1
```

Atributos de una clase

Los atributos de una clase son definidos según esta sintaxis:

*[**modifVisibilidad**] [modifAtributo] tipo nombreVariable [= valorInicial] ;*

modifVisibilidad Se utilizan para definir la visibilidad de los miembros de una clase (atributos y métodos) y de la propia clase.

Existen cuatro modificadores de acceso que ordenados de menor a mayor visibilidad, son: private, ninguno, protected, public.

Modificadores de Visibilidad

private. Cuando un atributo o método es definido como ***private***, su uso está restringido al interior de la clase, solamente puede ser utilizado en el interior de su misma clase. Este modificador puede ser aplicado a métodos y atributos, pero no a la clase.

ninguno. A falta de la definición de modificador se utiliza el *acceso por defecto*. Si un elemento (clase, método o atributo) tiene acceso por defecto, únicamente las clases de su mismo paquete tendrán acceso al mismo.

Modificadores de Visibilidad

protected. Un método o atributo definido como ***protected*** en una clase puede ser utilizado por cualquier otra clase de su mismo paquete y además por cualquier subclase de ella. Una clase no puede ser ***protected***, solo sus miembros.

public. El modificador public ofrece el máximo nivel de visibilidad. Un elemento (clase, método o atributo) ***public*** será visible desde cualquier clase, independientemente del paquete en que se encuentren.

Modificadores de Visibilidad

	private	*default*	*protected*	*public*
Clase	No	Si	No	Si
Método	Si	Si	Si	Si
Atributo	Si	Si	Si	Si
Variable local	No	No	No	No

Atributos de una clase

Los atributos de una clase son definidos según esta sintaxis:

*[modifVisibilidad] [**modifAtributo**] tipo nombreVariable [= valorInicial] ;*

modifAtributo son características específicas del atributo, son:

static: El atributo pertenece a la clase, no a los objetos creados a partir de ella.	final: El atributo es una constante, en ese caso debe de tener valor inicial obligatoriamente. Las constantes se escriben en mayúsculas.	transient: Marca al atributo como transitorio, para no ser serializado.

Métodos

Métodos de una Clase

Un método es una abstracción de una operación que puede hacer o realizarse con un objeto.

Una clase puede declarar cualquier número de métodos que lleven a cabo operaciones diversas con los objetos.

Los métodos permiten manipular los datos definidos por la clase y, en muchos casos, brindan acceso a esos datos.

Sobrecarga de métodos

- Cuando en una misma clase se definen varios métodos con el mismo nombre, se llama sobrecarga de métodos.
- Para que un método pueda sobrecargarse es imprescindible que se cumpla la siguiente condición:
  - Cada versión del método debe distinguirse de las otras en el número o tipo de parámetros.
  - El tipo de devolución puede o no ser el mismo.

Método Constructor

- Es un método especial que es ejecutado en el momento en que se crea un objeto de la clase (cuando se llama al operador **new**).
- Los constructores se utilizan para añadir aquellas tareas que deban realizarse en el momento en que se crea un objeto de la clase, por ejemplo, la inicialización de los atributos.
- Un constructor es un método perteneciente a la clase que posee unas características especiales:
  - Se llama igual que la clase.
  - No devuelve nada, ni siquiera void.
  - Pueden existir varios, siguiendo las reglas de la sobrecarga de métodos.
  - De entre los que existan, tan sólo uno se ejecutará al crear un objeto de la clase.

Constructores

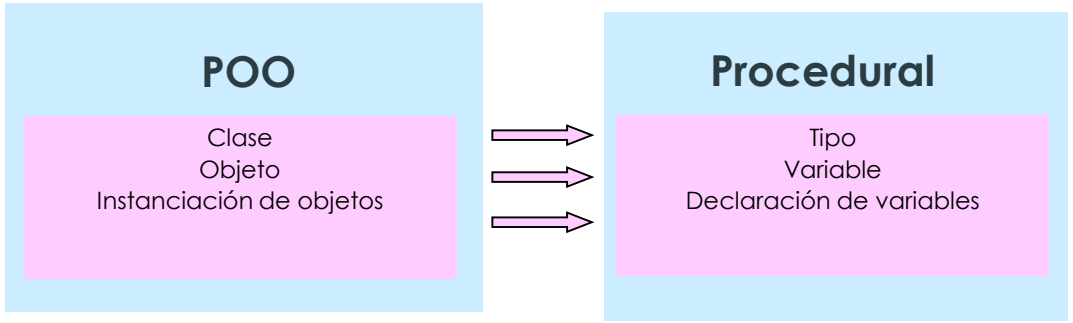
- Al crear un constructor hay que tener en cuenta las siguientes reglas:
  - El nombre del constructor debe ser el mismo que el de la clase.
  - El constructor no puede tener tipo de devolución.
  - Los constructores se pueden sobrecargar.
  - Toda clase debe tener al menos un constructor.

Métodos de Acceso

- Los atributos (también llamados variables) de una clase generalmente son privados por lo tanto para acceder a ellos se necesitan de ciertos métodos, los métodos que nos permiten este acceso son los métodos ***getters*** y ***setters*** también llamados métodos de acceso.

Objetos

Objetos



- En la POO se dice que un objeto: "integra datos y algoritmos"
- En la programación estructurada, variables y funciones están separadas

Objetos

- Cada objeto es responsable de inicializarse y destruirse en forma correcta
- Un objeto consta de:
  - Tiempo de vida
  - Estado
  - Comportamiento

Tiempo de Vida de un Objeto

La duración de un objeto en un programa siempre está limitada en el tiempo

La mayoría de los objetos sólo existen durante una parte de la ejecución del programa

Los objetos son creados mediante un mecanismo denominado instanciación

Los objetos dejan de existir cuando son destruidos

Estado de un Objeto

- Queda definido por sus atributos
- Con él se definen las propiedades del objeto, y el estado en que se encuentra en un momento determinado de su existencia.

Dudas O comentarios???

UNIVERSIDAD AUTONOMA METROPOLITANA
Casa abierta al tiempo Azcapotzalco