


Programación Orientada a Objetos (POO)

JAIME ALBERTO GUZMAN LUNA, Ph.D
Universidad Nacional de Colombia
Medellín



Contenido

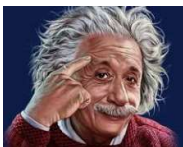
- Paradigmas de programación
- Modelado Orientado a Objetos con UML

Paradigmas de programación

Visión general

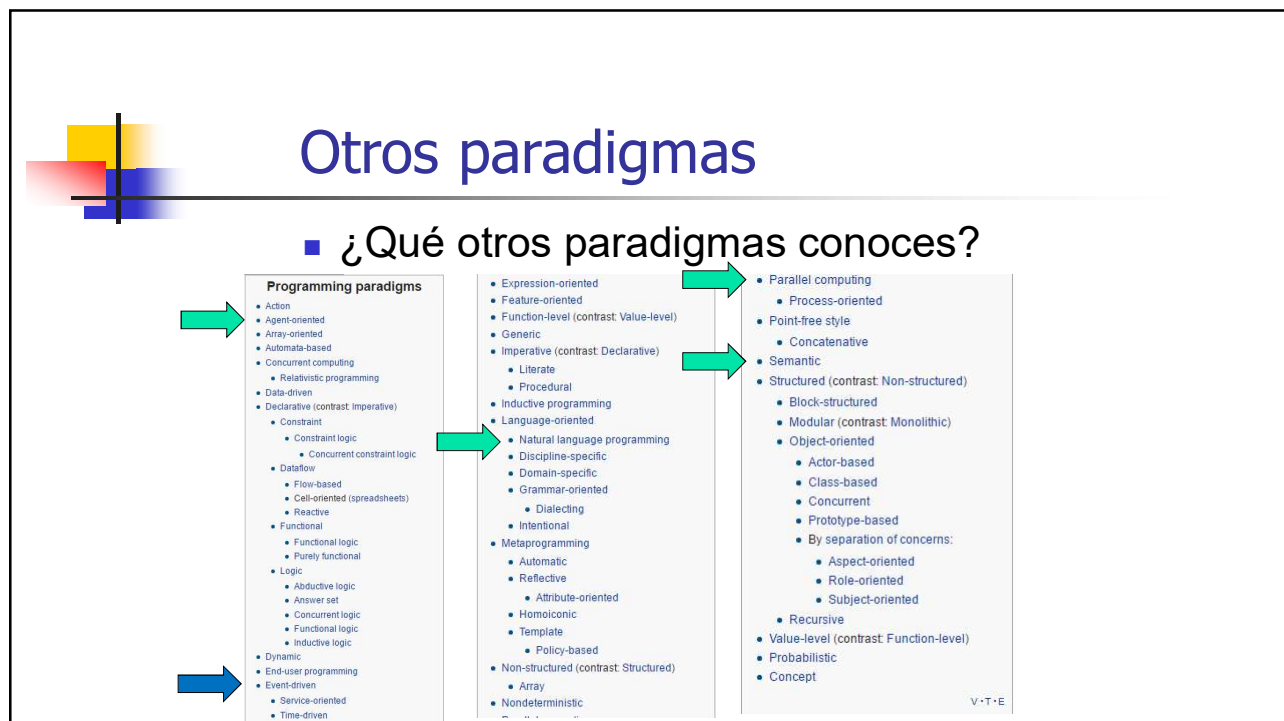
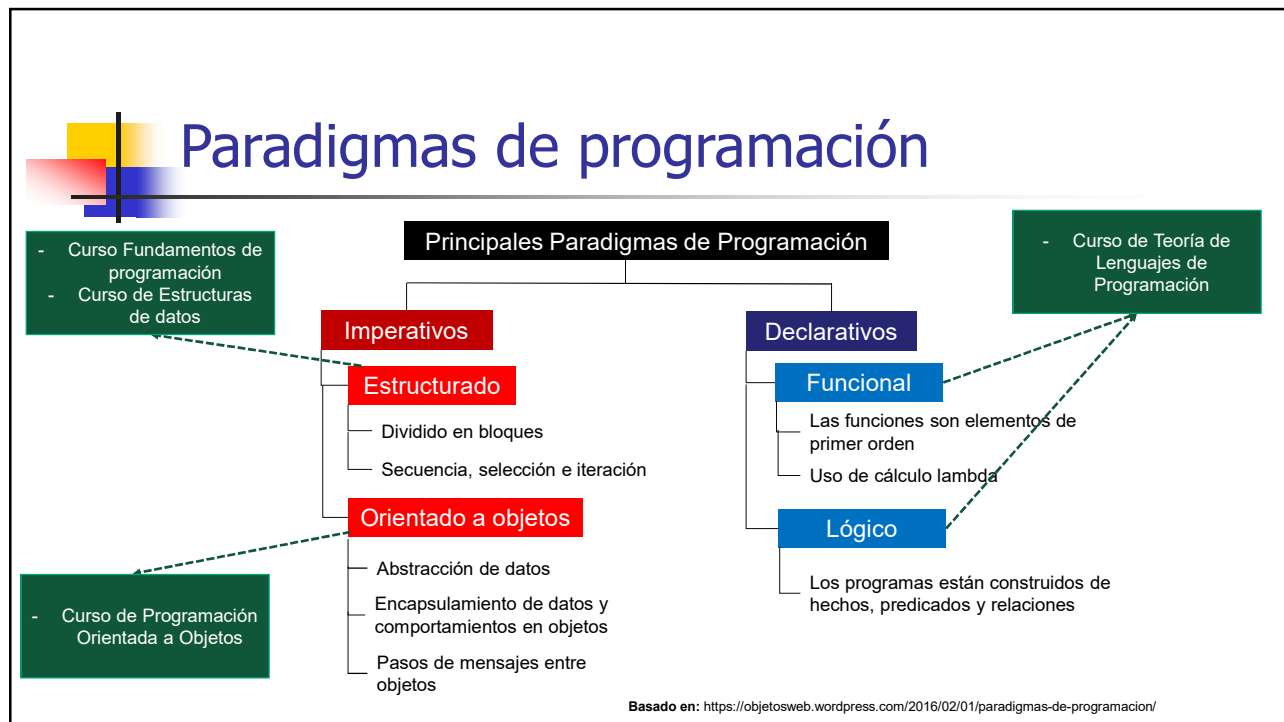
Paradigma de programación

- Que es un paradigma?
 - Paradigma debe ser concebido como un conjunto de **métodos**, **reglas** y **generalizaciones** utilizadas conjuntamente por aquellos entrenados para **realizar el trabajo científico de investigación...**
- Que es un paradigma de programación?
 - Es un conjunto de **patrones conceptuales** que **moldea la forma de:**
 - Razonar sobre problemas
 - Formular soluciones
 - Estructurar programas



Tomado de: <http://www.iqelaya.itc.mx/~vicente/Programacion/Paradigmas.pdf>

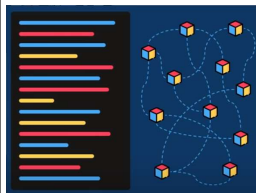
Tomado de: <https://cmappublic.ihmc.us/rid=1S43NBKR3-290XV27-ZZWZ/Paradigmas%20de%20programacion.cmap>



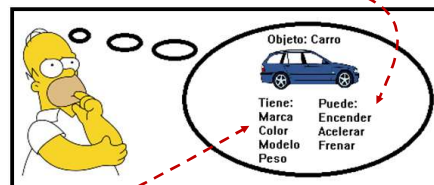
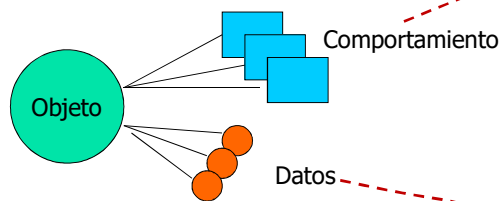
Programación orientada a Objetos

(POO-OOP)

Aspectos básicos de la POO



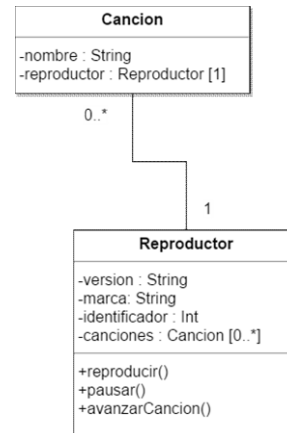
- En la POO la solución de un problema se plantea en términos de **objetos** y las relaciones entre ellos.
- **Objeto** = tipo abstracto de datos con estado (atributos) y comportamiento (Operaciones) propios



Modelado de objetos

- ¿Cómo modelamos objetos?
 - Generalmente se utilizan los diagramas de clases de UML.

UML (lenguaje unificado de modelado) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad



Ejemplo de UML

Que es la POO?

- Programación orientada a objetos (OOP) es el proceso de convertir el diseño o modelado orientado a objetos en un programa de software.

```

class Carro:
    ruedas = 4
    def __init__(self, color, aceleracion):
        self.color = color
        self.aceleracion = aceleracion
        self.velocidad = 0

    def acelera(self):
        self.velocidad = self.velocidad + self.aceleracion

    def frena(self):
        v = self.velocidad + self.aceleracion
        if v < 0:
            v = 0
        self.velocidad = v
  
```

Annotations in the image:

- Atributos** (green arrow) points to `ruedas = 4`.
- Métodos** (green arrow) points to `def frena(self):`.
- Nombre de la clase** (red arrow) points to `class Carro:`.
- Cuerpo de la clase** (red arrow) points to the body of the class (the code between `class Carro:` and the end of the class).

Versión Python

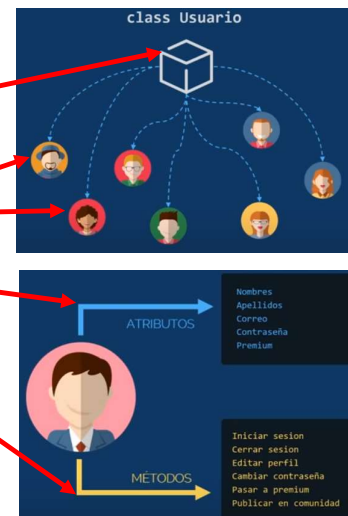
Características de la POO

- Organización de los programas de manera que representen la interacción de las cosas en el mundo real.
- Un programa consta de un conjunto de objetos.
- Los objetos son abstracciones de cosas del mundo real.
- Nos interesa qué se puede hacer con los objetos más que cómo se hace.
- Cada objeto es responsable de unas tareas.
- Los objetos interactúan entre sí por medio de mensajes.
- Cada objeto es un ejemplar de una clase.
- Las clases se pueden organizar en una jerarquía de herencia.

Conceptos de la POO (1)

■ Básicos:

- **Clases:** categorías de objetos con propiedades y operaciones comunes
- **Objeto:** una instancia de una clase.
- **Atributos:** Estados del objeto
- **Métodos:** operaciones sobre los objetos
- **Relaciones** (objetos compuestos)



Basado en: https://www.youtube.com/watch?v=DlphYPc_HKk



Conceptos de la POO (2)

■ Ciclo de vida de los objetos

■ Creación

- Empleado x = crear Empleado (...)
- Constructores: inicialización de atributos

■ Manipulación

- Acceso a atributos: x . nombre
- Invocación de métodos: x . salario_neto ()

■ Destrucción

- Explícita (C++)
- Automática (Java y Python)→ *garbage collection*
- Ejemplo: Al desaparecer un directivo, actualizar relación de subordinados



Conceptos de la POO (3)

■ Pilares de la Programación Orientada a Objetos



Abstracción



Encapsulamiento



Polimorfismo



Herencia

Conceptos de la POO (4)

■ Abstracción

- Operación intelectual que ignora selectivamente partes de un todo para facilitar su comprensión
- Cuando se crea un sistema, se debe hacer un proceso mental para definir todas las clases y sus componentes

Ejemplo: Gestor académico



■ Proceso General: Identificar las clases

- Los Alumnos
- Los Cursos
- Los Departamentos

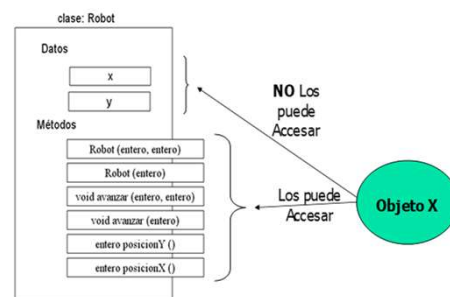
■ Proceso Detallado: Identificar en cada clase:

- Los Atributos
- Los Métodos

Conceptos de la POO (5)

■ Encapsulamiento

- Mecanismo que consiste en organizar los datos y métodos del objeto, conciliando el modo en que estos se acceden.
- Miembros privados y públicos
- Interfaz pública de una clase (miembros públicos: atributos y métodos)
 - Se pueden invocar desde fuera de la clase
- Ejemplo (clase Robot)
 - Datos:
 - **privado** x (entero)
 - **privado** y (entero)
 - Métodos:
 - **público** void avanzar (entero, entero)
 - **público** entero posicionX ()
 - **público** entero posicionY ()
 - **público** void avanzar (entero)



Conceptos de la POO (6)

■ Polimorfismo

- Capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación.
- En algunos lenguajes, el término polimorfismo es también conocido como "Sobrecarga de métodos"
 - Los objetos permiten aceptar distintos parámetros para un mismo método (diferentes implementaciones) generalmente con comportamientos distintos e independientes para cada uno de ellos.

Ejemplo

```
class Overload {
    void demo (int a) { System.out.println ("a: " + a); }
    void demo (int a, int b) {
        System.out.println ("a and b: " + a + "," + b); }
    double demo(double a) {
        System.out.println("double a: " + a); return a*a; }
}
```

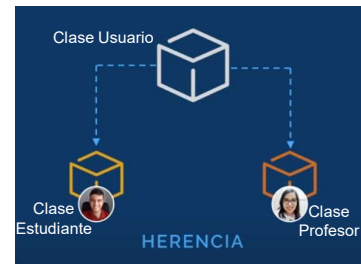
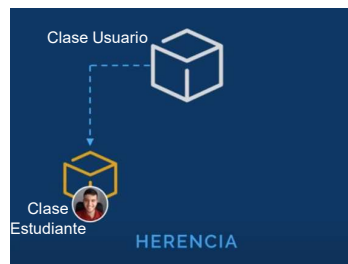
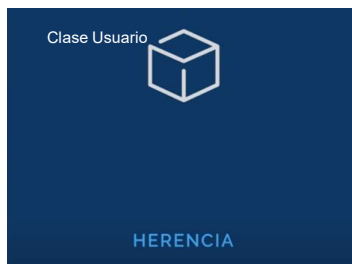
```
class MethodOverloading {
    public static void main (String args []) {
        Overload Obj = new Overload();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result); }
}
```

Versión Java

Conceptos de la POO (7)

■ La Herencia

- Propiedad que permite a los objetos ser construidos a partir de otros objetos.
- Tiene como objetivo final la **reutilización** de código.





Modelado Orientado a Objetos

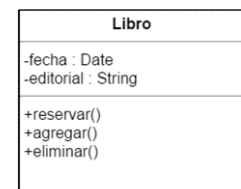
UML



Clases en UML

- En su forma más simple, una clase en UML se dibuja como un rectángulo dividido en hasta tres secciones.
 - La **sección superior** contiene el nombre de la clase (centrado y singular).
 - La **sección central** contiene los atributos o la información que contiene la clase.
 - **visibilidad nombreAtributo : tipo**
 - La **sección final** contiene las operaciones que representan el comportamiento que exhibe la clase.
 - **visibilidad nombreMétodo(parámetros) : tipoRetorno**

Nota: Las secciones de atributos y operaciones son opcionales.



Visibilidad

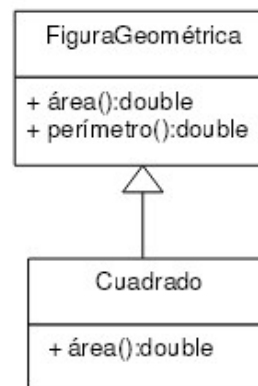
- ¿Cómo una clase revela sus operaciones y datos a otras clases?
 - R:// Usando **visibilidad**
- Hay cuatro tipos diferentes de visibilidad que se pueden aplicar a los elementos de un modelo UML (Publica, Protegida, de Paquete y Privada).

Table 3. Visibility Options on UML Class Diagrams

Visibility	Symbol	Accessible to
Public	+	All objects within your system
Protected	#	Instances of the implementing class and its subclasses
Private	-	Instances of the implementing class
Package	~	Instances of classes within the same package

Herencia

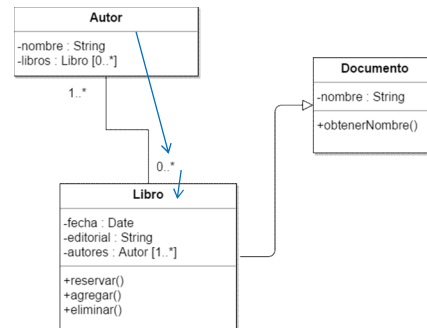
- Se representa con una flecha que va desde la clase hija a la clase padre.
- En este caso **Cuadrado** hereda todos los atributos y métodos de **FiguraGeométrica**, si se desea se pueden sobrescribir los atributos y métodos.



Relación

- Se representa como una línea continua entre 2 clases (puede contener el nombre de la relación encima de la línea).

0..* (cero o muchos)
 1..* (1 o muchos)
 0..1 (cero o uno)
 1 (uno)



Tips

Sobre las clases:

- Coloque el nombre de la clase en singular.
- Empiece colocando los atributos primitivos de las clases.

Sobre las relaciones:

- Una relación (línea) entre 2 clases (**A** y **B**) se convierten en 2 atributos no primitivos. Se añade a **A** como atributo no primitivo en la clase **B**; y se añade a **B** como atributo no primitivo en **A**. (a criterio del diseñador).

Sobre las operaciones:

- Asigne las operaciones a aquella clase que contiene la información necesaria para ejecutar la operación.
 - Por ejemplo: registrar persona debe ir en la clase Persona; eliminar un comentario debe ir en la clase Comentario.



Actividad: La Biblioteca

- Realizar el diagrama de clase del sistema para una biblioteca
 - Una biblioteca maneja dos clases de materiales: libros y revistas. Estas dos clases tienen en común dos características: título y año
 - Una revista, adicional a las dos características descritas anteriormente, tiene: issn, volumen y mes
 - La biblioteca tiene copias de libros. Un libro adicional a las dos características descritas anteriormente, se caracteriza por su isbn, tipo (novela, teatro, poesía, ensayo), editorial y autor.
 - Los autores se caracterizan por su nombre, nacionalidad y fecha de nacimiento
 - Cada copia tiene un identificador y puede estar en la biblioteca, prestada, con retraso o en reparación
 - Los lectores se identifican con un número de socio, tienen su nombre, teléfono y dirección. Cada lector puede tener un máximo de 3 libros en préstamo
 - Cada copia de libro se presta un máximo de 30 días, por cada día de retraso, se impone al lector una "multa" de dos días sin posibilidad de coger un nuevo libro
- Realizar un diagrama de clases y añadir los métodos necesarios para realizar el préstamo, la devolución de libros y multar. Asociar estos métodos al lector.