POO

POO

Uno de los paradigmas de programación mas importantes es la programación orientada a objetos.

Para entender las características de la POO, necesitamos unos conceptos básicos.

El modelado de un elemento en el mundo real esta dado por:

- **Atributos** = datos
- Comportamiento = funciones.

Dado a que la unión de estos elementos esta plasmada en la vida real, esto es un **objeto**.

En la POO el programa no se divide en tareas, se divide en modelos de objetos físicos o simulados.

Los objetos se pueden agrupar por categorías, al conjunto de objetos de una misma categoría se le denomina una **Clase**.

POO

Pasos para solucionar problemas utilizando POO:

- Identificación de los objetos del problema
- agrupamiento por clases
- identificar datos y operaciones por clase
- identificar relación entre clases

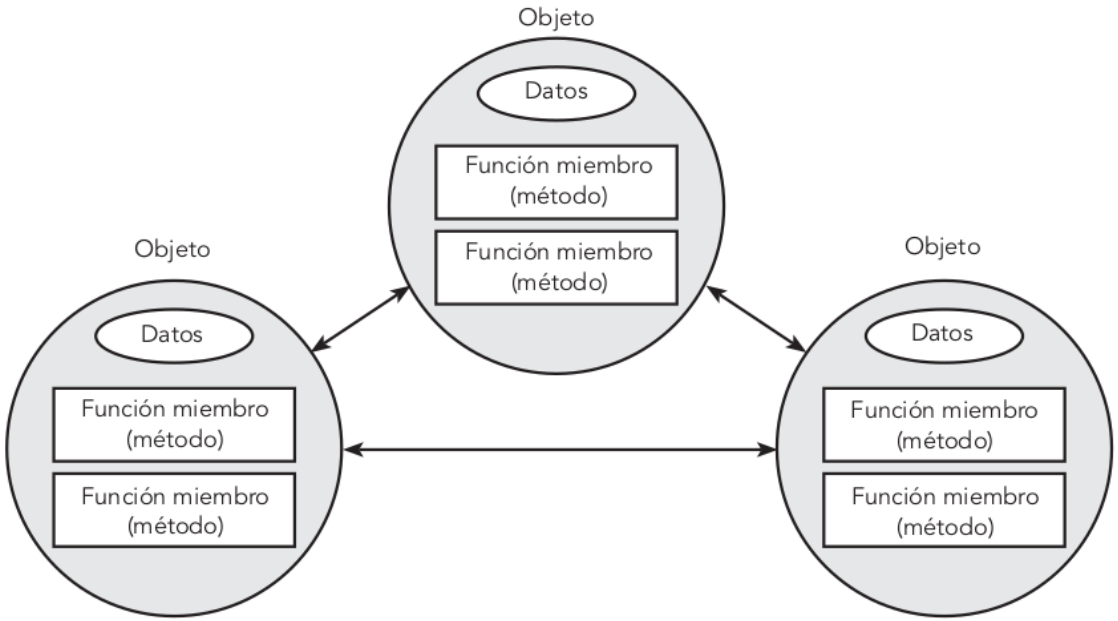
Una clase es la implementación de un tipo abstracto de datos, describe los atributos y operaciones.

Una clase es un tipo de dato, del cual se pueden crear variables de este tipo denominadas **instancias**.

A las operaciones definidas para los objetos las llamamos **métodos**, los cuales activan por medio de *mensajes*.

En pocas palabras: **un objeto es una instancia de una clase**.

POO



	Mundo Real	**En OOP**
Clase Generalización de características (atributos y comportamientos)	**Perro** Raza, Color, Edad, Corre,	**Clase** Define datos y métodos
Objeto Instancia de una clase distinguible por sus características específicas	**Tino** Pastor Alemán Marrón 7 meses Veloz	**Objetos** Ocupa espacio, se crea y se destruye

POO

Todo objeto tiene:
Estado: conjunto de valores de los atributos en un instante de tiempo.
Comportamiento: conjunto de operaciones que puede realizar un objeto.
Identidad: diferencia los objetos no importando los estados de estos.

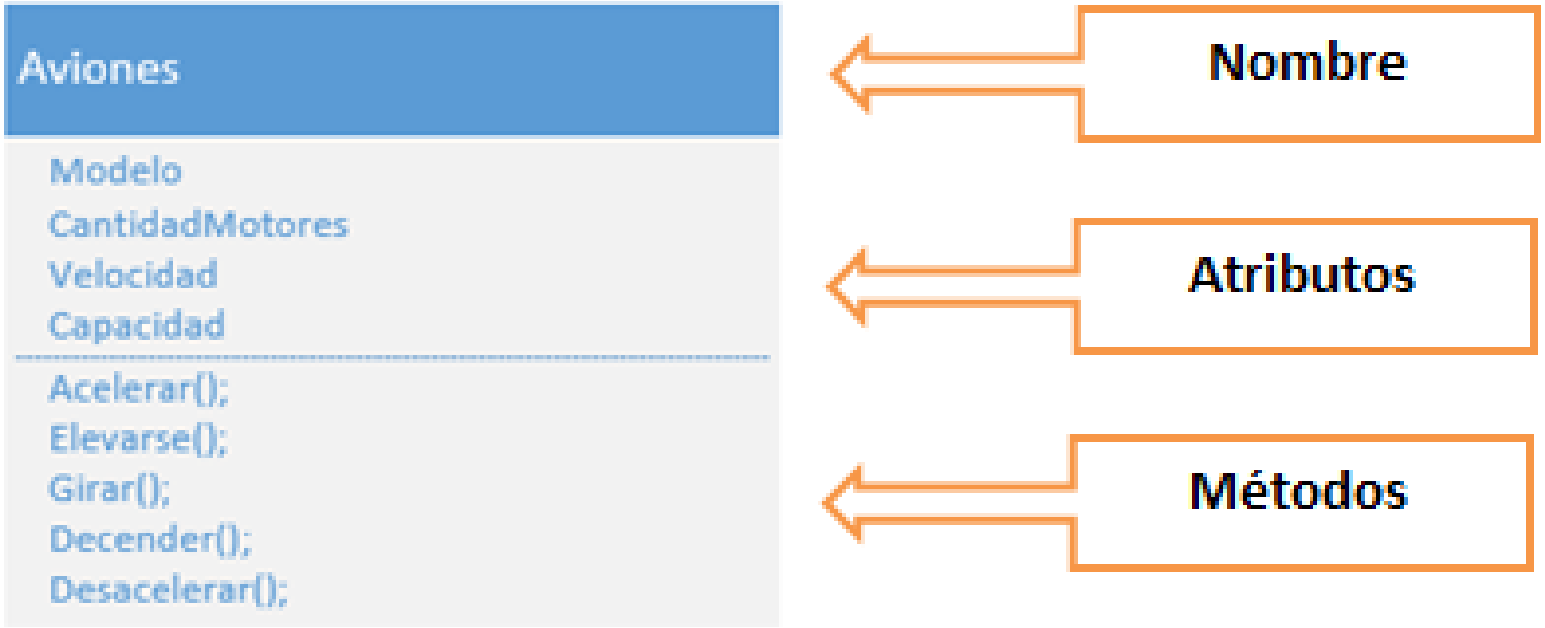
Propiedades fundamentales de este paradigma de programación son:

- **Abstracción:** es una forma de reducir la complejidad de un sistema en características o procesos simples.
- **Encapsulamiento y ocultación de datos:** agrupación de datos y operaciones bajo la misma entidad, las cuales se ocultan de otros objetos.
- **Herencia:** relación de generalización, los objetos tienen que organizarse en jerarquía.
- **Reusabilidad:** las clases la pueden utilizar otros programadores. *Como las bibliotecas*
- **Polimorfismo:** operaciones con el mismo nombre en diferentes clases y diferentes acciones.

Lenguaje unificado de modelado

OBJETO = Estado + Comportamiento + Identidad

La representación gráfica en UML es necesaria para la POO.



Pseudocódigo

```
#formato de clase
clase NombreClase
#elementos de clase
    #Declaracion de atributos constantes o variables
    #[privado| publico|protegido] <tipo>:<nombre>=<valor>
    #pertenece a la clase
    #[estatico][privado| publico|protegido] <tipo>:<nombre>=<valor>
    const
        privado real: PI = 3.1416
    var
        estatico publico real: precio
    #Declaracion de metodos
    constructor NombreClase([lista_de_parametros])
        #declaracion de variables locales
    fin_constructor
    #abstracto para crear clases hermanas y tener metodos
    diferentes
    #[estatico][abstracto][privado| publico|protegido] <
    tipo_retorno> funcion <nombre_funcion>([lista_de_parametros])
    publico entero funcion devolverx()
        inicio
        devolver (x)
    fin_funcion
```

Pseudocódigo

```
    #[estatico][abstracto][privado| publico|protegido]
    procedimiento <nombre_procedimiento>([lista_de_parametros])
    publico procedimiento fijarx(E entero: cx)
        inicio
        x <- cx
    fin_procedimiento
    destructor NombreClase()
        #declaracion de variables locales
    fin_constructor
fin_clase
```

Nota: método con tipo de retorno que no sea void debe de utilizar **return**

Comportamiento de cada tipo de encapsulamiento.

Tipo de miembro	miembro misma clase	amiga	miembro clase derivada	función no miembro
Privado	x	x		
Protegido	x	x	x	
Publico	x	x	x	x

Ejemplo Pseudocódigo

Dimensiones de varios libros, se crea la clase libro y objetos pertenecientes a esta.

```
clase Libro
    var
        real: ancho
        real: alto
        real: profundidad
    constructor Libro (real:a,b,c)
        inicio
            ancho <- a
            alto <- b
            profundidad <- c
    fin_constructor
fin_clase
```

Ya declarada la clase se puede crear objetos.

```
Libro analisisNumerico                  # declaracion del objeto
analisisNumerico = nuevo Libro()        # creacion del objeto
```

Si usar un atributo o un método.

```
analisisNumerico.ancho = 10                 # atributo
analisisNumerico.nostrarDimensiones()       # metodo
```

Ejemplo

Se le solicita un programa el cual le permita ingresar diferentes fechas y mostrarlas en pantalla utilizando los conceptos de programación orientada a objetos.

Ejemplo

Se le solicita un programa el cual le permita ingresar diferentes fechas y mostrarlas en pantalla utilizando los conceptos de programación orientada a objetos.

Atributos:

Métodos:

Ejemplo

Se le solicita un programa el cual le permita ingresar diferentes fechas y mostrarlas en pantalla utilizando los conceptos de programación orientada a objetos.

Atributos:
elementos de fecha

Métodos:
obtener e imprimir la información

Ejemplo

Se le solicita un programa el cual le permita ingresar diferentes fechas y mostrarlas en pantalla utilizando los conceptos de programación orientada a objetos.

Atributos:
elementos de fecha

Métodos:
obtener e imprimir la información