Programación
1º Desarrollo Aplicaciones Web

- Unidad 1. Principios de la programación Orientada a Objetos
  - Concepto y estructura de un objeto
  - Métodos y propiedades
  - Relaciones entre objetos
  - Características de la programación OO
  - Ventajas e inconvenientes de la programación OO

José Manuel Pérez Lobato

TÉCNICAS DE PROGRAMACIÓN

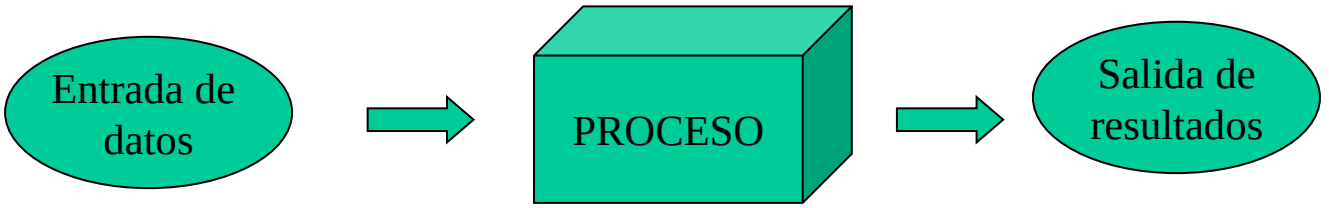
- ***Programación estructurada:*** conjunto de técnicas que incorporan:
  - Diseño descendente (Top-down)
  - Posibilidad de descomponer una acción compleja en acciones más simples
  - Uso de estructuras básicas de control (secuencial, alternativa, repetitiva)

- ***Programación modular:*** división de un programa en módulos de manera que cada uno de ellos tenga encomendada la ejecución de una única tarea. Cada módulo es programado y depurado con independencia del resto.

- ***Programación orientada a objetos:*** Al contrario que en la programación estructurada, en la cual las instrucciones y los datos están separados, en la P.O.O. los programas se construyen con un conjunto de **objetos** que presentan una serie de propiedades (**atributos**) y un conjunto de operaciones (**métodos**) que operan sobre esos atributos. Estos objetos cooperan entre si para resolver el problema.

PARTES DE UN PROGRAMA

- Todo programa está constituido por un conjunto de órdenes capaces de manipular un conjunto de daos. Estas instrucciones pueden clasificarse en tres grandes bloques:
  - ***Entrada de datos:*** toman datos de un periférico externo, depositándolos en memoria central para poder ser procesados
  - ***Proceso o algoritmo:*** instrucciones encargadas de procesar la información
  - ***Salida de datos o resultados:*** envían los datos de memoria principal a un periférico de salida



Concepto y estructura de un objeto: Encapsulación

- Componentes básicos en la programación tradicional
  - Código
  - Datos
- Componentes de un programa en OO
  - Objeto (encapsula el código que manipula los datos con la declaración y almacenamiento de esos datos)

Encapsular=: interfaz simplificada visible + detalles ocultos

Niveles: instrucciones > funciones > clases > librerías > frameworks > aplicaciones

Clases

- Definiciones
  - Una Clase es un proyecto, o prototipo, que define los atributos y los métodos comunes a un cierto tipo de objetos.
  - Las clases son las matrices de las que luego se pueden crear múltiples objetos del mismo tipo. La clase define los atributos y los métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas operaciones.
  - Es la abstracción de las características comunes de un tipo determinado de objetos. Describe un tipo de objetos con características comunes

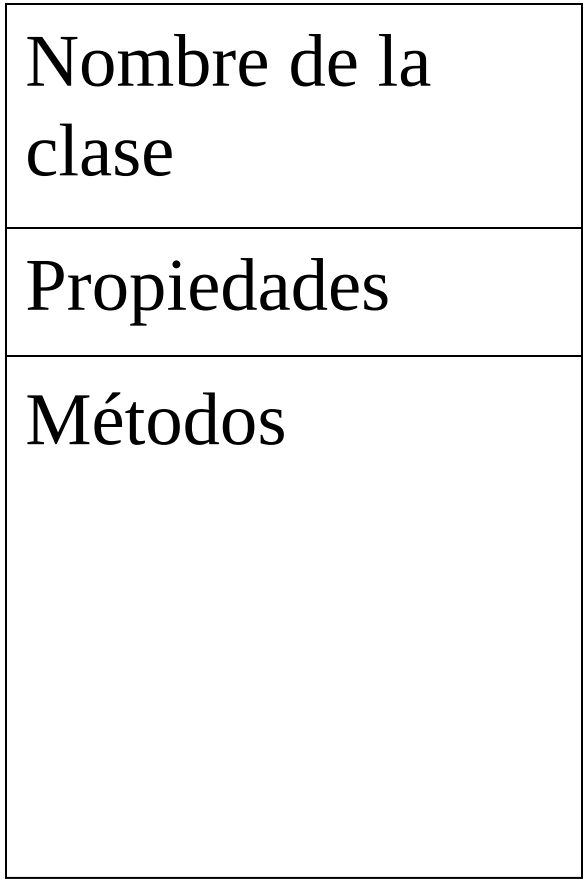
Clases (II)

- Encapsulan operaciones y atributos propios cada tipo de objeto: el código fuente de un objeto puede escribirse y mantenerse de forma independiente a los otros objetos contenidos en la aplicación.
- Permiten tener un control total sobre 'quién' o 'qué' puede acceder a sus miembros, es decir, los objetos pueden tener miembros públicos a los que podrán acceder otros objetos o miembros privados a los que sólo puede acceder él. Estos miembros pueden ser tanto atributos como operaciones

Componentes de una clase

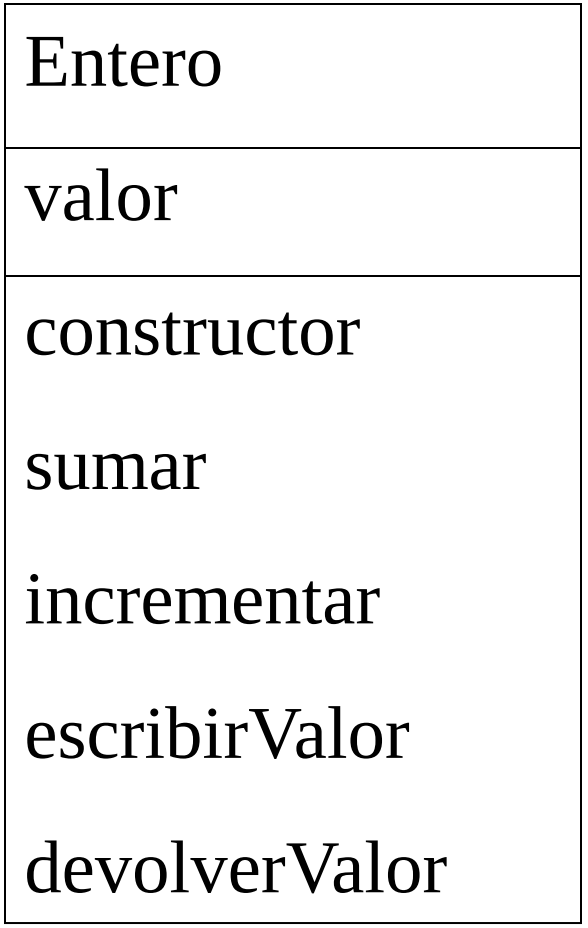
- **Propiedades o Atributos**: Elementos que mantienen el estado del objeto. Pueden ser de instancia o de clase. Tienen asociado un tipo
- **Métodos**: Definen el comportamiento. Mensaje para realizar alguna acción en un objeto. Es similar a las funciones en los lenguajes de programación tradicionales. Permiten el paso de datos (parámetros) y devolución de resultados.

Estructura de una Clase



Ejemplo de una Clase

```
CLASE Entero
Propiedad valor
Método constructor (a) {valor=a}
Método sumar (a) {valor=valor+a)}
Método incrementar ()
{valor=valor+1}
MÉTODO escribirValor() {IMPRIMIR
valor}
MÉTODO int devolverValor() (DEVOLVER
valor)
Método liberar() {liberar_recursos}
```



Objetos (instancias de clase)

- Un OBJETO es una instancia única de una clase que mantiene la estructura y comportamiento definidos por la clase.
- Una clase describe los atributos generales de un objeto, incluyendo los tipos de cada atributo y los métodos que pueden operar en el objeto. Una instancia (objeto) es un elemento concreto de una clase de objetos.
- Los valores que toman las propiedades son los elementos caracterizan el estado de un objeto y lo hace diferente de los restantes que forman parte de la misma organización.

Uso de los Objetos

- En una aplicación los objetos se crean (constructor), se utilizan (métodos del objeto) y se eliminan (destructor o recolector de basura).
- Creación de 2 objetos de la clase Entero

```
numero1= Entero.constructor(5);
numero2= Entero.constructor(10);
```

- Utilización

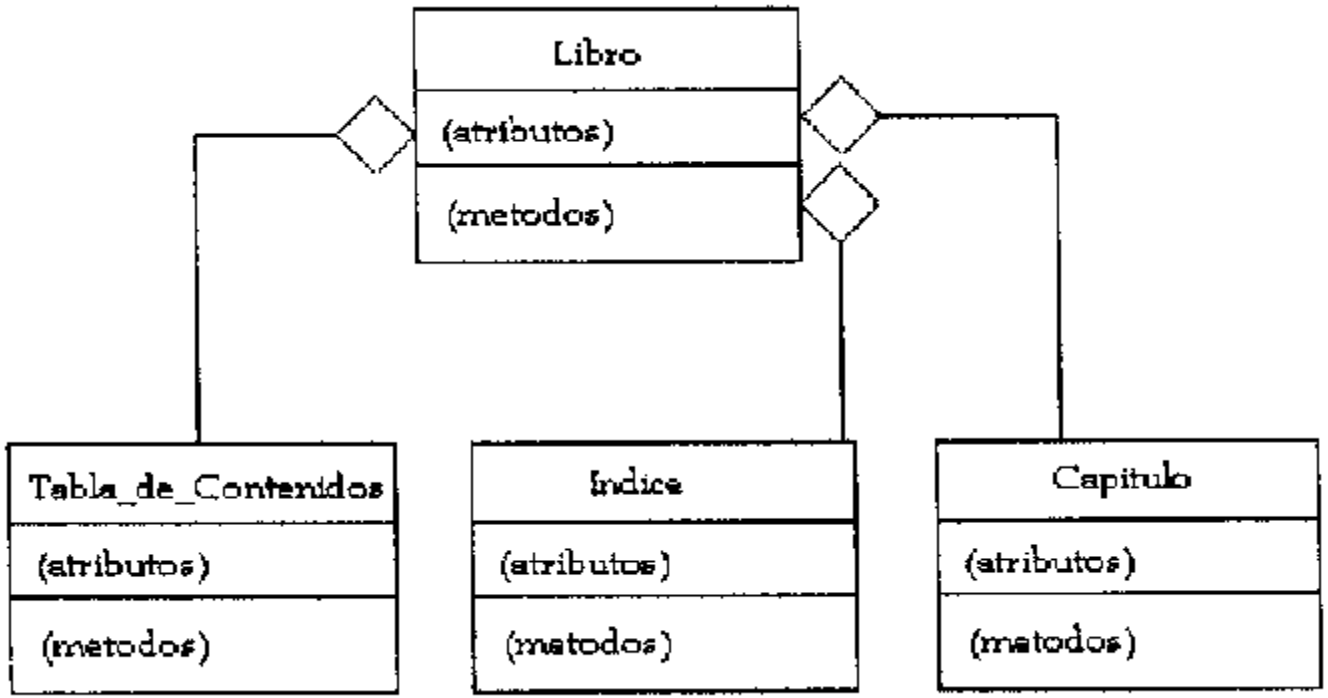
```
numero1.incrementar();
numero1.sumar(6);
numero1.escribirValor();
x= numero1.devolverValor();
```

Relaciones entre objetos

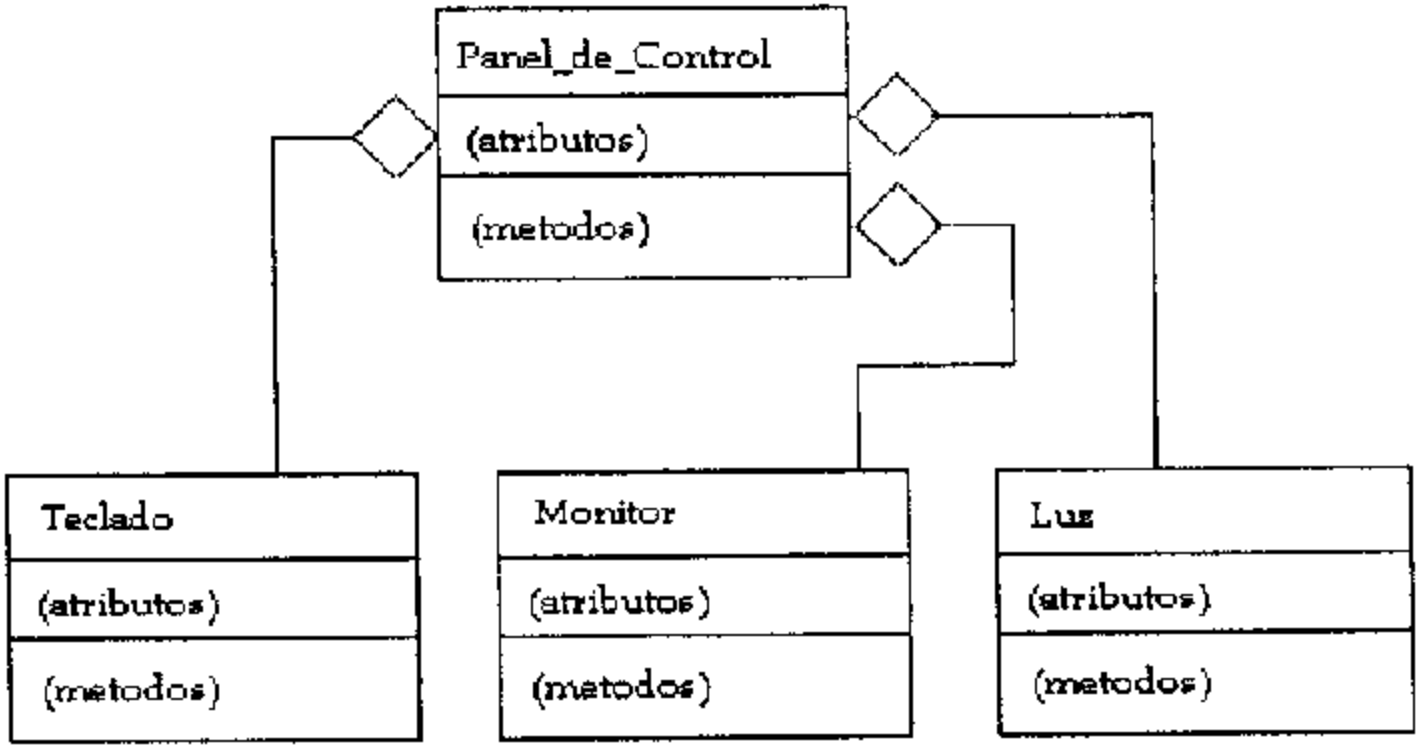
- **Agregación** : Una clase puede contener como atributos a objetos de otras clases. Libro->Capítulos.
- **Generalización** (Herencia): Los objetos hijo de un objeto padre son una especialización de éste. Permite crear una clase partiendo de otra que ya existe. La nueva clase tendrá todas los atributos y los métodos de su 'superclase', y además se le podrán añadir otros atributos y métodos propios. Ej. Ave-->Gallina

  Se llama 'Superclase' a la clase de la que desciende una clase.
- **De uso**. Para poder crear una aplicación se necesita más de un objeto, y estos objetos no pueden estar aislados unos de otros, para comunicarse esos objetos se envían mensajes. Los mensajes son simples llamadas a los métodos del objeto con el se quiere comunicar para indicarle que haga una determinada acción.

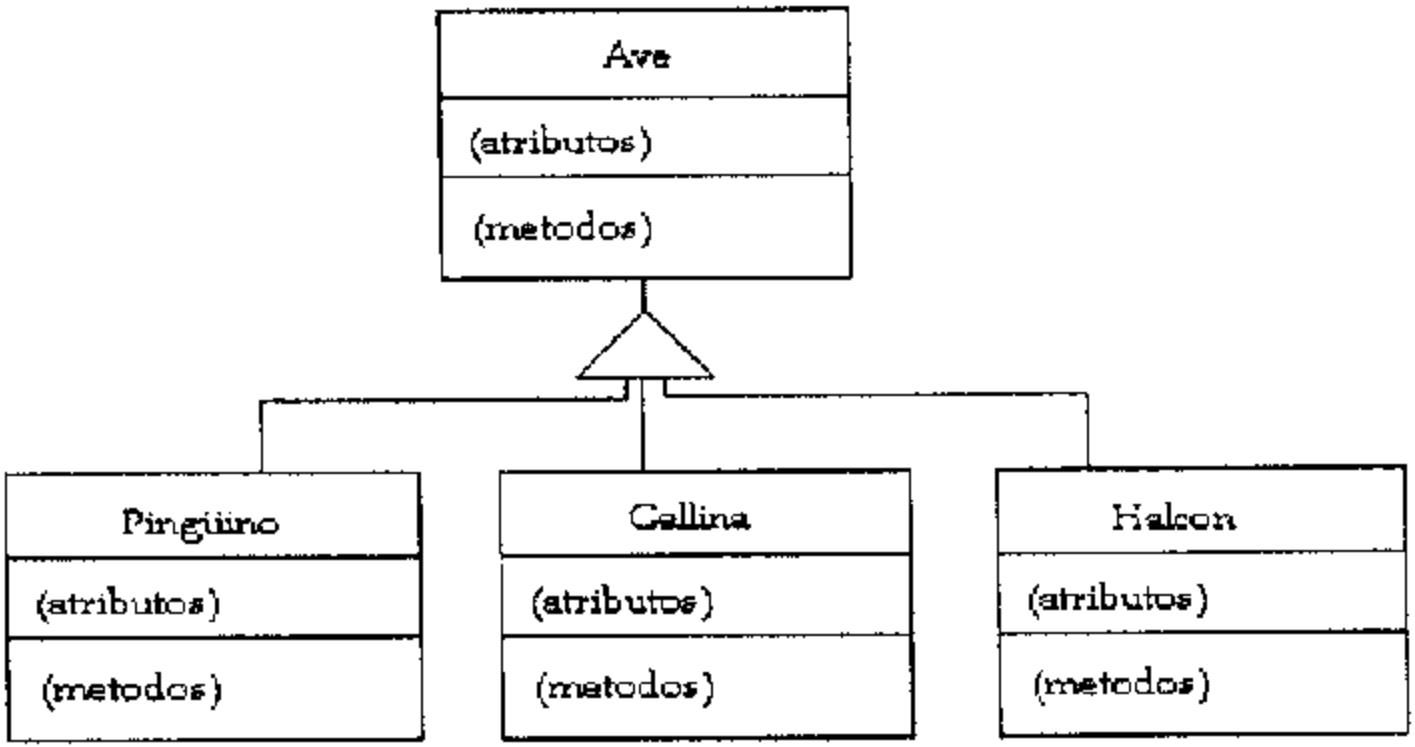
Ejemplo Agregación(I)



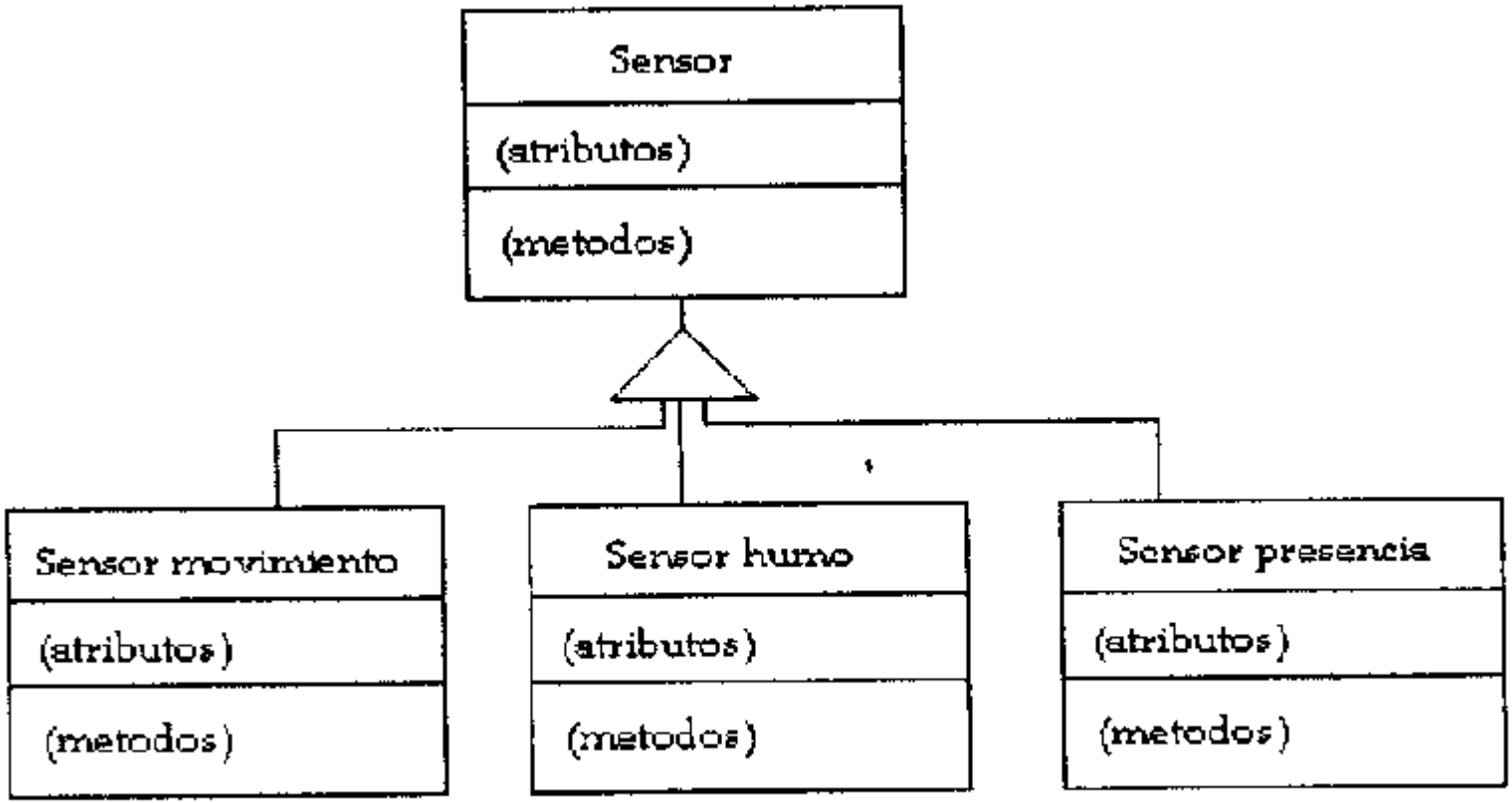
Ejemplo Agregación(II)



Ejemplo Generalización(I)



Ejemplo Generalización(II)



Características de la programación OO

- Encapsulamiento
- Polimorfismo
- Herencia
- Paso de Mensajes
- Ligamiento Dinámico

Encapsulamiento

- **Objeto = Caja Negra**. La estructura interna permanece oculta, tanto para el usuario como para otros objetos diferentes, aunque formen parte de la misma jerarquía, reduciendo la propagación de efectos colaterales cuando ocurren cambios.
- La **información** contenida en el objeto será **accesible sólo a través de** la ejecución de los **métodos adecuados** creándose una interfaz para la comunicación con el mundo exterior.
- **Cada objeto es una cápsula** que contiene todos los datos y métodos ligados a él, facilita que un objeto determinado pueda ser transportado, con todo lo que contiene, a otro punto de la organización o a una organización diferente. Si el objeto se ha construido correctamente, sus métodos seguirán funcionando en el nuevo entorno, como si no hubiera sido transplantado.

Protección de un objeto

- Los atributos y métodos de un objeto pueden ser, básicamente, de 2 tipos
  - Privados: Sólo pueden ser utilizados por los métodos de esa clase (objetos de igual clase si pueden acceder)
  - Públicos: Pueden ser accedidos por cualquier objeto.
  - ……

Polimorfismo

- Un método tiene múltiples implementaciones que se seleccionan en base a qué tipo de objeto se utiliza o a los parámetros que recibe en la llamada al método => Se sobrecarga el método
- Al mismo nombre de método se le asocian implementaciones distintas.
- Ej.
  - Método : calcularArea
  - Objetos: Triángulo, Cuadrado.

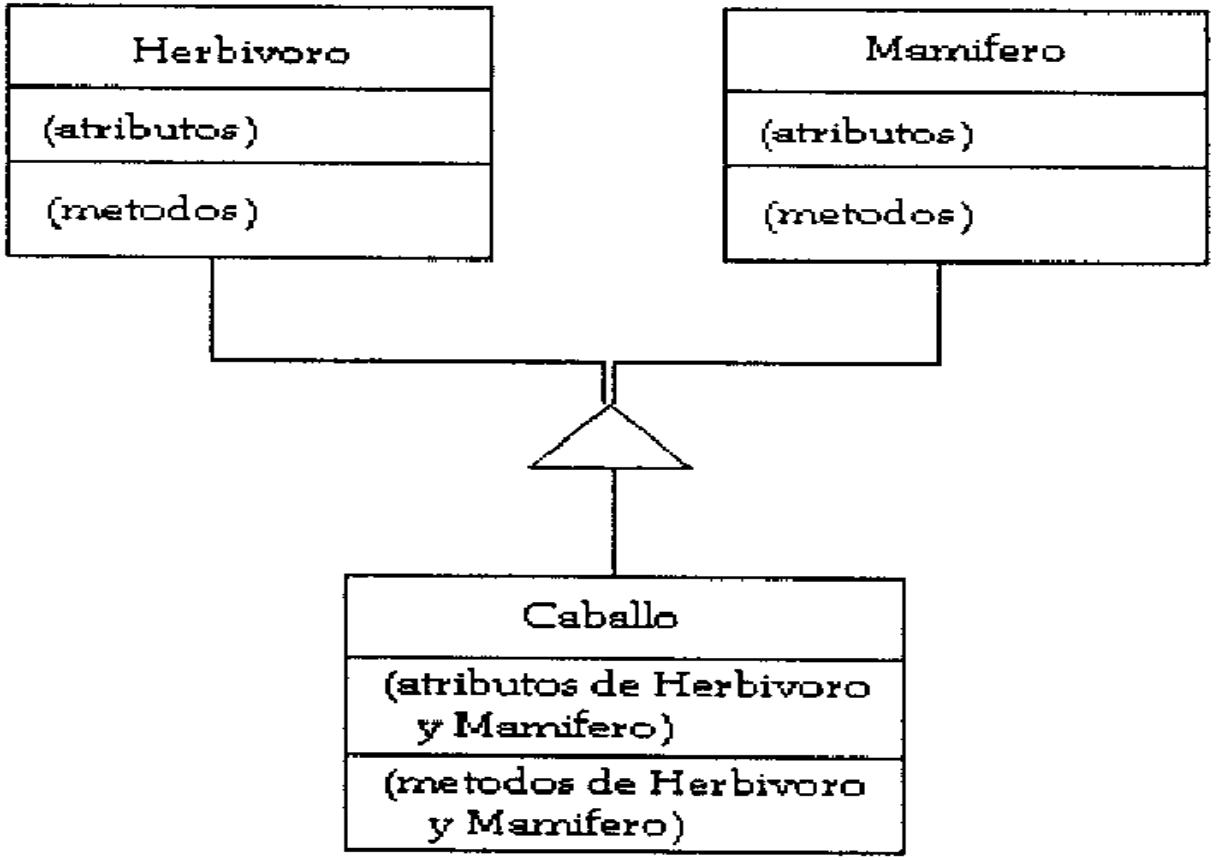
Tipos de Polimorfismo

- **Sobrecarga de operadores**: Se realizan acciones distintas dependiendo del tipo de datos sobre el que se actúa. Ej. Operador + sobre enteros o sobre Cadenas de Caracteres
- **Sobreescritura de operadores**: Redefinición del comportamiento de un método en una subclase. Ej.
  - CuentaAhorro.calcularInteres
  - CuentaAhorroEmpleado.calcularInteres: Utiliza un tipo de interés mayor por ser utilizado con un empleado de la empresa

Herencia

- Objetos toman propiedades y métodos de otros.
- Es unidireccional, un objeto hijo puede heredar de su padre, pero un padre no puede heredar de su hijo.
- Tipos
  - simple (Java) : se aplica cuando el objeto que recibe la herencia tiene un único padre
  - múltiple (C++): el objeto tiene más de un padre o algunos de sus antepasados tiene varios padres. Un ejemplo de herencia múltiple existiría en la definición de un objeto “Caballo” que heredaría propiedades y métodos de las clases “Herbívoro” y “Mamífero”.

Ejemplo Herencia Múltiple



Paso de Mensajes

- Peticiones enviadas por un objeto a otro para que se active un método. Ej.
  - objeto "Cuadrado”, método dibujar, éste método puede mandar un mensaje a un objeto "Rectángulo" pidiéndole que se dibuje con una altura igual a la base.

Vínculo Dinámico

- Capacidad de retrasar hasta el instante de la recepción del mensaje la decisión sobre la clase de objeto que lo recibe y el método concreto que debe ejecutarse.
- Hasta que no se produzca la llamada durante la ejecución de la aplicación no se tiene establecido cual es el método que se va a ejecutar, lo que permite una asignación dinámica de los recursos del sistema.

```
f es una figura Geométrica
f= Cuadrado.constructor();
f.calcularArea()
```

```
f es FiguraGeométrica
f= Circulo.constructor();
f.calcularArea()
```

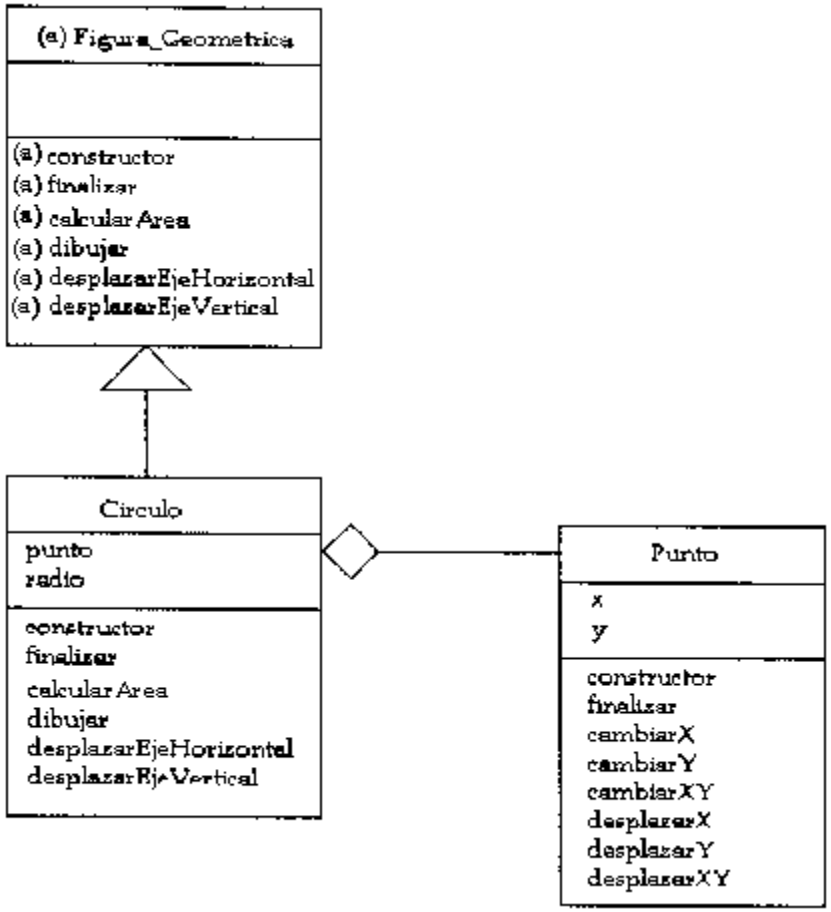
Ejemplo de diseño

```
CLASE FiguraGeometrica
METODO constructor(){}
METODO dibujar() {……}

CLASE Punto
PROPIEDAD real x
PROPIEDAD real y
METODO constructor (a, b) {x=a y=b}
METODO desplazarX (real dx) {x=x+dx;)

CLASE Circulo HEREDA DE FiguraGeometrica
PROPIEDADES  real radio; Punto punto;
METODO constructor (a, b, r)
{padre.constructor(), punto.constructor(a,b),
radio=r}
METODO real calcularArea() {DEVOLVER 2*3.1415* radio)
METODO dibujar() {……}
METODO desplazarEjeHorizontal(real dx) {punto.desplazarX(dx)}
```

Ejemplo Diseño (II)



Ventajas de la POO

•**Correctitud** del programa: el modelo mental desarrollado por el diseñador, utilizando entidades como objetos, es fácilmente transportable a un modelo directamente comprensible por el ordenador,
•**Modularidad**: Aplicación modular: compuesta por módulos independientes y robustos. Un objeto =un módulo, es una entidad de la realidad que se modeliza y posee una interfaz para la comunicación con otros objetos, eliminado la posibilidad de que otros objetos interfieran en su funcionamiento..

Ventajas de la POO(II)

•**Extensibilidad**: El desarrollo no finaliza con la implementación, posteriormente esta su utilización, mejoras, corrección de errores. Con la POO es fácil modificar la aplicación para introducir nuevos casos particulares de las clases existentes, nuevas clases y subclases, nuevas propiedades y nuevos métodos, extendiendo su ámbito.
•**Eliminación de redundancias:** La herencia permite eliminar redundancias de manera que no haga falta volver a redefinir propiedades y métodos ya existentes para algunos objetos.
•**Reutilización:** POO permite volver a utilizar objetos ya desarrollados para una aplicación integrándolos de una forma distinta en otra, con el consiguiente ahorro de recursos.

Desventajas de la POO

Necesidad de preparar de forma concienzuda a los programadores que los vayan a utilizar ya que la forma de pensar la solución del problema es distinta a la que hasta la actualidad se estaban utilizando.

El gran número de estos lenguajes existentes viene a indicar que cada uno interpreta las características de la orientación a objetos de una forma distinta por lo que el paso de uno a otro no se puede considerar inmediato..