

/PROGRA
MADO>
RES**3.0**

Programación Orientada a Objetos



Programación Orientada a Objetos

Historia

- Nace en los años 60 aplicado a simulaciones de sistemas físicos.
- Diseño de programas paralelamente al sistema físico.
- Su objetivo principal era reducir la complejidad de desarrollo y mantenimiento del software.
- A lo largo de los años han ido apareciendo lenguajes de programación que implementan estas ideas:
Eiffel, SmallTalk, C++, Java.....

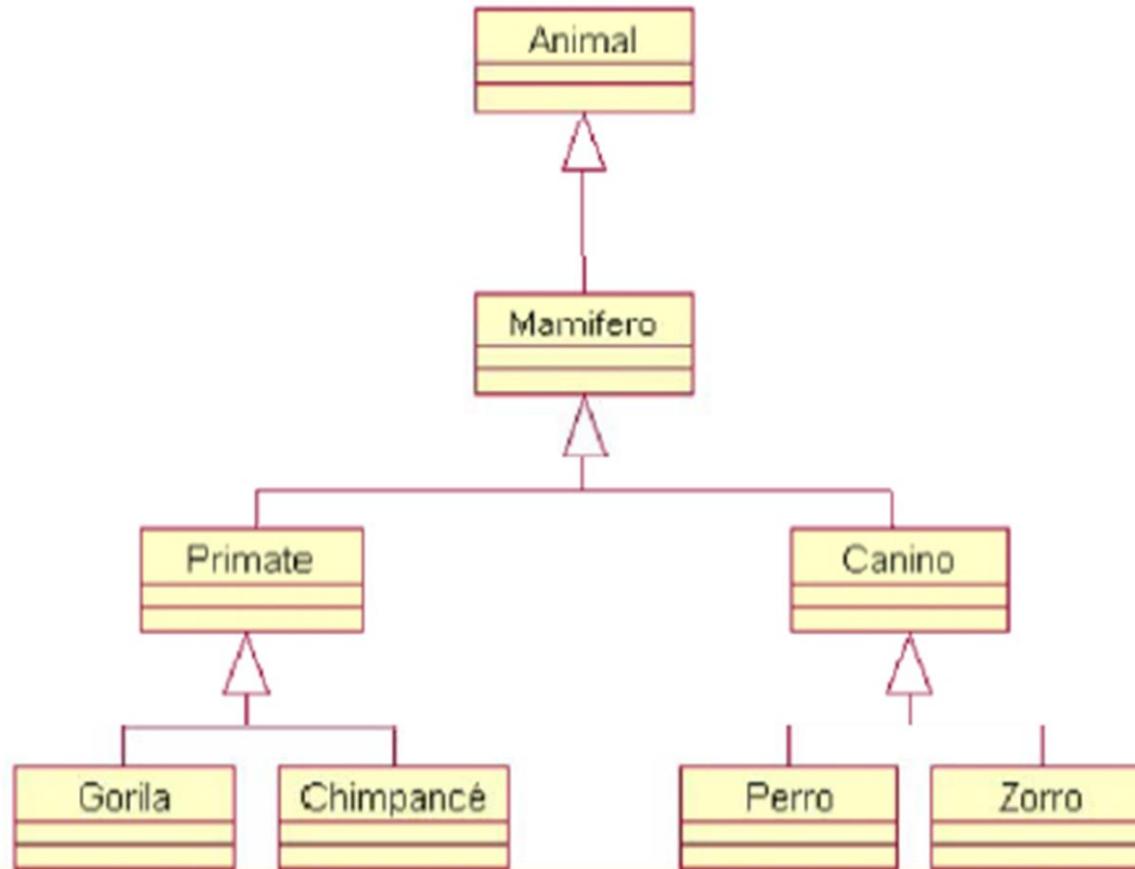
Programación Orientada a Objetos

Ventajas de la O.O.

- Suministra modelos similares a los del mundo real.
- Facilita el desarrollo de sistemas complejos.
- Facilita la reutilización.
- Permite el desarrollo iterativo de aplicaciones.
- Facilita la interoperabilidad de aplicaciones.

Programación Orientada a Objetos

Similitud con el ‘mundo real’





Programación Orientada a Objetos

Facilita el desarrollo de Sistemas Complejos

- Elementos fundamentales del modelo de Objetos:
 - Abstracción.
 - Encapsulamiento.
 - Modularidad.
 - Herencia.



Programación Orientada a Objetos

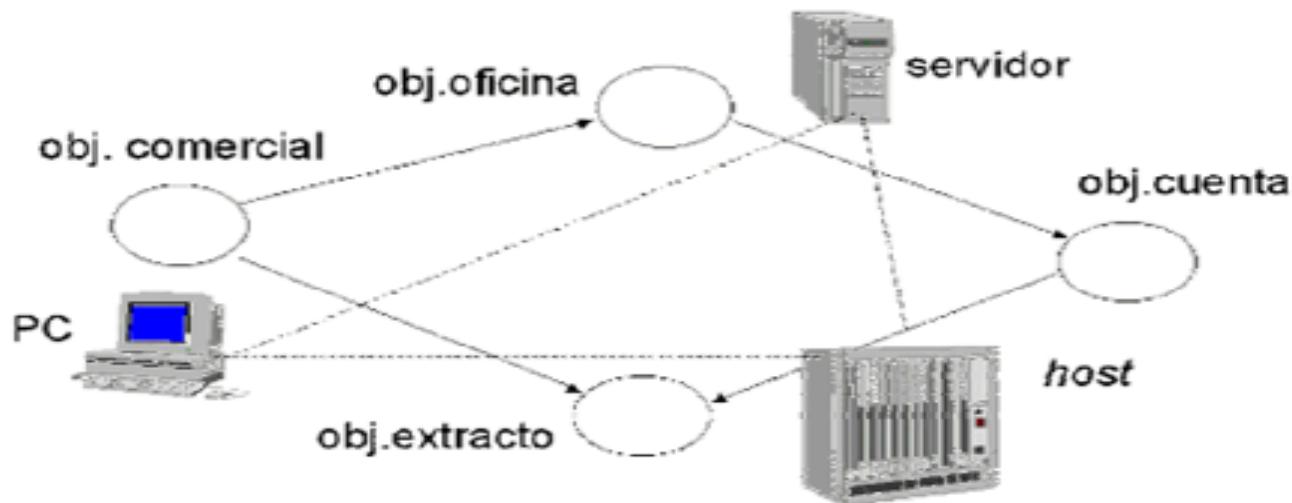
Facilita la reutilización

- La O.O. soporta la reutilización basada en la herencia, composición y parametrización.
- La O.O. soporta la reutilización basada en la utilización de librerías de componentes, patrones de diseño y arquitecturas (también conocidas con el nombre de framework).

Programación Orientada a Objetos

Facilita la interoperabilidad

- Las arquitecturas Orientadas a Objetos permiten un mejor aislamiento de las dependencias de plataforma.





CONCEPTOS BÁSICOS

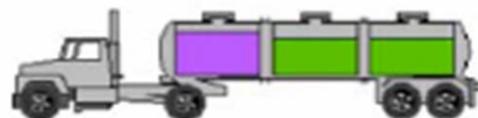
Definición de objeto



Un objeto es:



Una cosa tangible y/o visible.



(camión)



Algo que puede comprenderse intelectualmente.



(proceso)



Una entidad software.



(lista)

Características de un objeto

-  Un objeto tiene:
 -  Identidad: un identificador único.
 -  Estado: un conjunto de propiedades (atributos).
 -  Comportamiento: un conjunto de operaciones (métodos).
-  Los términos objeto e instancia son intercambiables.

Otras definiciones

- Un objeto se caracteriza por un número de **operaciones** y un **estado** que recuerda el efecto de estas operaciones.
- Un objeto tiene un **estado**, **comportamiento** e **identidad**; la estructura y comportamiento de objetos similares se definen en sus clases comunes.

Ivar Jacobson

Grady Booch

Ejemplo práctico



Modelo físico:



Botón para mostrar la Hora

Botón para mostrar el Día



Modelo informático.

unReloj

Atributos

hora (horas, min, seg)
dia (dia, mes, año)
modelo
numSerie

Métodos

getHora
getDia
incrementarHora
incrementarDia
limpiarPantalla

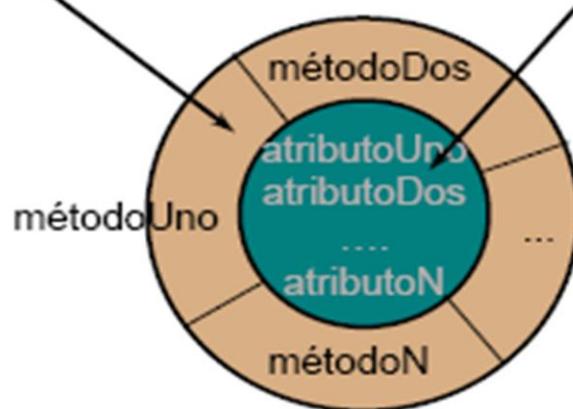
Estructura de un objeto

Métodos

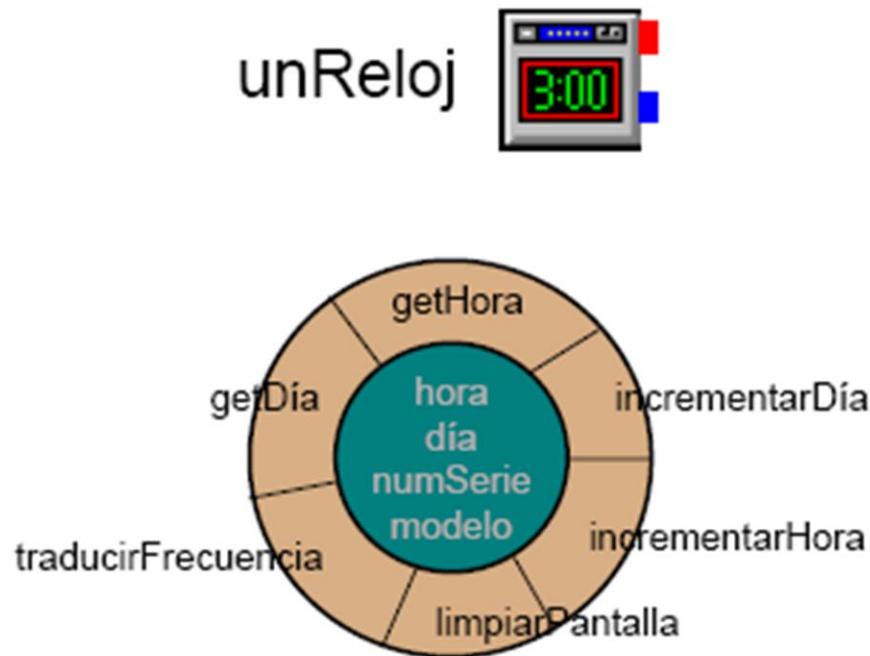
Operaciones permitidas.
Pueden estar o no ocultas para el usuario.

Atributos

Estructura encapsulada de los datos.



Ejemplo práctico



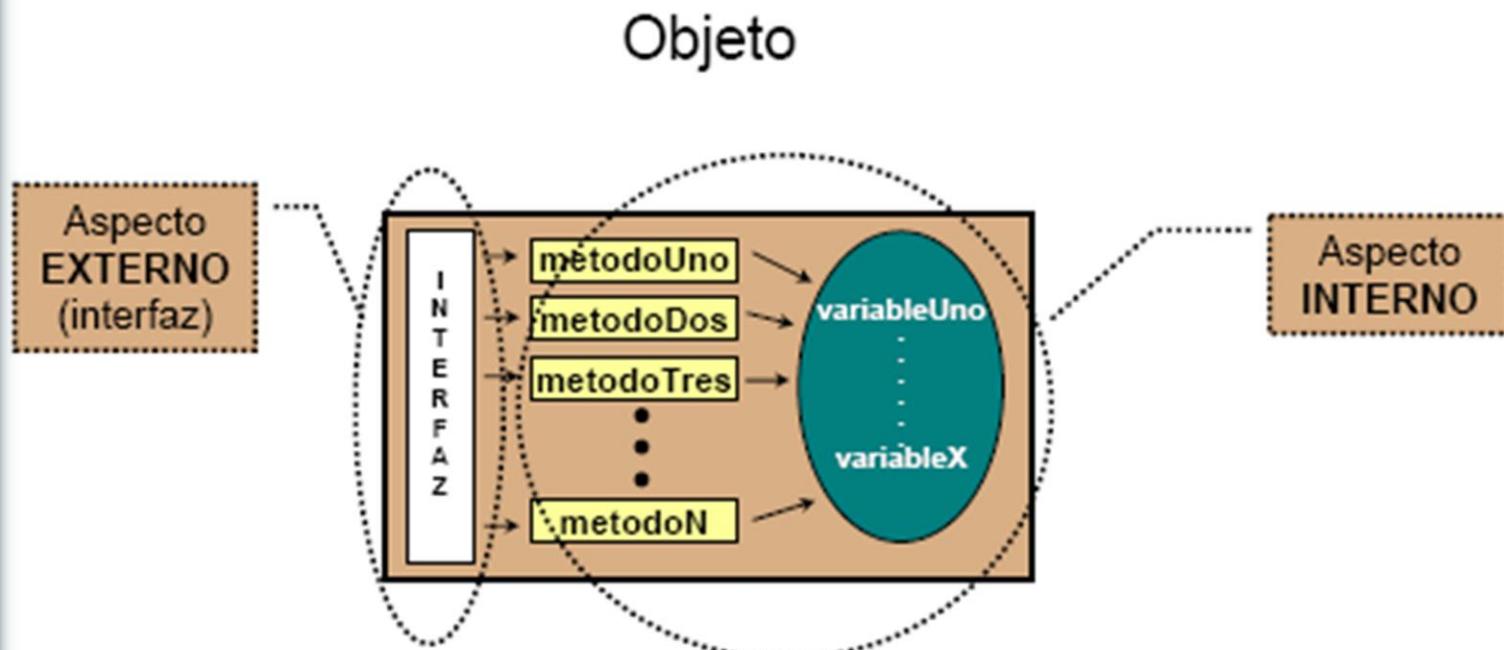
Definición de atributo

- Es una característica fundamental de cada objeto de una clase.
- Una clase puede definir un cierto número de atributos estáticos.
- Todos los atributos tienen algún valor. Este valor puede ser una cantidad, una relación con otro objeto, etc...

Definición de método

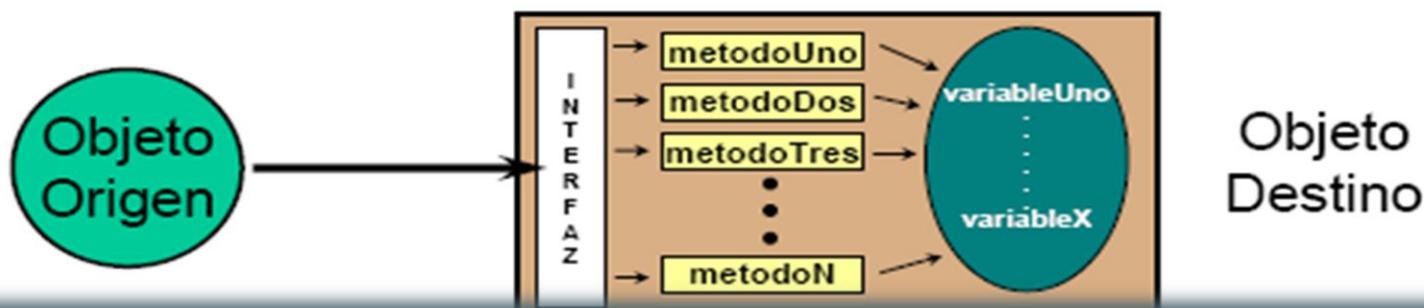
- Es una acción que se realiza sobre un objeto para consultar o modificar su estado.
- Tipos de operaciones:
- Modificador (setter): altera el estado de un objeto.
- Selector (getter): accede al estado de un objeto sin alterarlo.
- Iterador: permite acceder a todas las partes de un objeto.
- Constructor: crea un objeto e inicializa su estado.
- Destructor: limpia el estado de un objeto y lo destruye.
- Propósito general: la lógica del programa.

El aspecto de los objetos



Interfaz

- Es el aspecto externo del objeto. La parte visible y accesible para el resto de objetos.
- También se le define como el protocolo de comunicación de un objeto.
- Puede estar formado por uno o varios métodos. No todos los métodos de un objeto tienen porque formar parte del interfaz.



Definición de clase

- Una clase es la representación de la estructura y comportamiento de un objeto
- Es un patrón para la definición de atributos y métodos para un tipo particular de objetos.
- Todos los objetos de una clase dada son idénticos en estructura y comportamiento pero son únicos (aunque tengan los mismos valores en sus atributos).
- Instancia es el término utilizado para referirse a un objeto que pertenece a una clase concreta.

Estructura de una clase

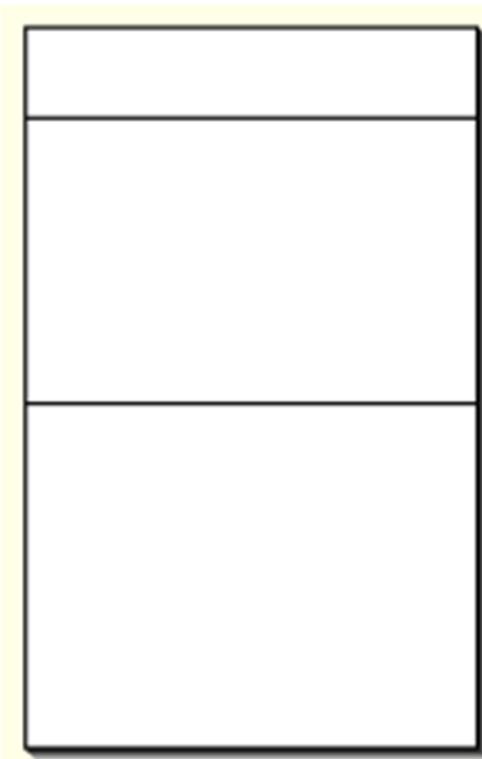
Nombre



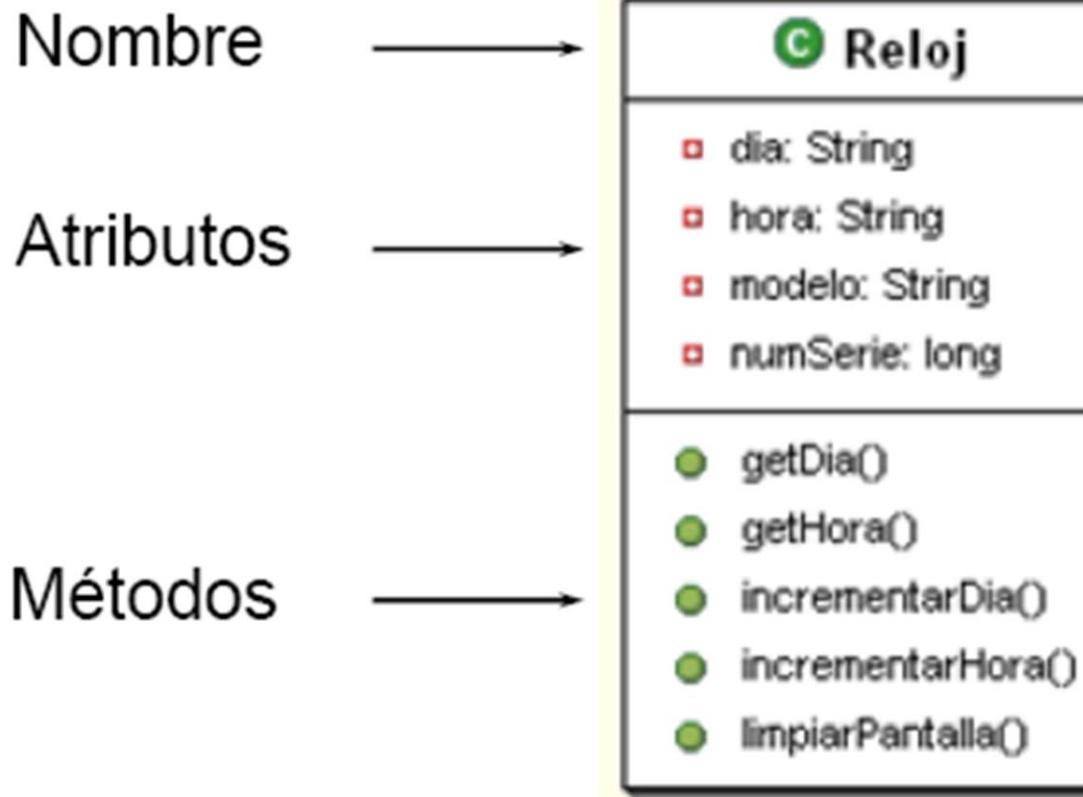
Atributos



Métodos



Ejemplo práctico



Clases vs. Objetos



Clase:

Un patrón para la definición del estado y el comportamiento de un tipo particular de objetos.

Todos los objetos de una clase dada son idénticos en estructura y comportamiento, pero tienen identidad única.



Objeto (instancia):

Pertenece a una clase en particular.

Los objetos son creados y destruidos en tiempo de ejecución. Residen en el espacio de memoria.

Clasificación

-  La clasificación es el medio por el que ordenamos el conocimiento:
-  Es fundamentalmente un problema de búsqueda de similitudes.
-  Al clasificar buscamos grupos de cosas que tengan una misma estructura o exhiban un comportamiento común.
-  Clasificación y desarrollo O.O.:
 -  Clasificación significa que los objetos con la misma estructura de datos y con el mismo comportamiento se agrupan para formar una clase.
 -  Esta es una de las tareas fundamentales en el análisis y diseño O.O.

Ejercicio

-  ¿Quién dice qué?: Clase, Objeto, Atributo, Método
-  El valor de mis atributos puede ser distinto al de los de mi semejante: _____.
-  Yo me comporto como una plantilla: _____.
-  A mi me gusta hacer cosas: _____.
-  Yo puedo tener muchos métodos: _____.
-  Yo represento el estado: _____.
-  Yo represento el comportamiento: _____.
-  Yo estoy en los objetos: _____.

Ejercicio (solución)

-  ¿Quién dice qué?: Clase, Objeto, Atributo, Método
-  El valor de mis atributos puede ser distinto al de los de mi semejante: **Objeto**.
-  Yo me comporto como una plantilla: **Clase**.
-  A mi me gusta hacer cosas: **Objeto, método**.
-  Yo puedo tener muchos métodos: **Clase, objeto**.
-  Yo represento el estado: **Atributo**.
-  Yo represento el comportamiento: **Método**.
-  Yo estoy en los objetos: **Atributo, método**.

Ejercicio



¿Quién dice qué?: Clase, Objeto, Atributo, Método



Yo vivo en memoria: _____.



Yo soy usado para crear instancias: _____.



Mi estado puede cambiar: _____.



Yo declaro métodos: _____.



Yo puedo cambiar en ejecución: _____.

Ejercicio

-  ¿Quién dice qué?: Clase, Objeto, Atributo, Método
-  Yo vivo en memoria: **Objeto**.
-  Yo soy usado para crear instancias: **Clase**.
-  Mi estado puede cambiar: **Objeto**.
-  Yo declaro métodos: **Clase**.
-  Yo puedo cambiar en ejecución: **Objeto, atributo**.

Paradigmas de la O.O.



- Los pilares de la Programación Orientada a Objetos son:
 - Abstracción.
 - Encapsulación.
 - Ocultamiento.
 - Herencia.
 - Polimorfismo.
- Cualquier lenguaje O.O. debe implementar estos conceptos.

Abstracción

- Consiste en la generalización conceptual de los atributos y comportamiento de un determinado conjunto de objetos.
- La clave de la programación O.O. está en abstraer los métodos y los datos comunes a un conjunto de objetos y almacenarlos en una clase.
- Hay que centrarse en lo que es y lo que hace un objeto, antes de decidir cómo debería ser implementado.

Encapsulación y ocultamiento

- Consiste en separar el aspecto externo del objeto, al cual pueden acceder otros objetos, del aspecto interno del mismo, que es inaccesible para los demás.
- Permite tratar a un objeto como una caja negra.
- Permite que se modifique la implementación interna de un objeto sin afectar a los clientes que lo utilizan.

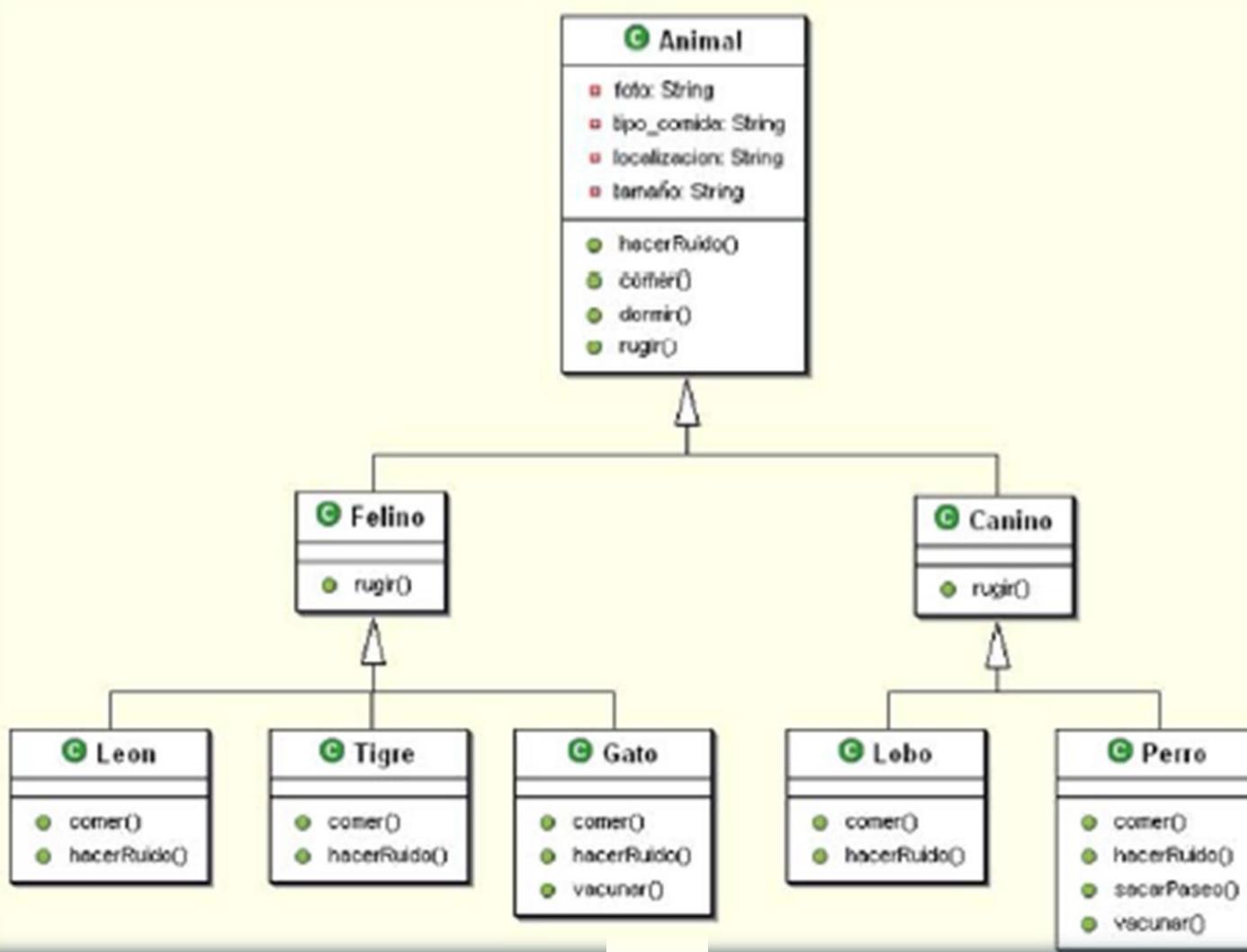
Relación de herencia

- Se basa en la existencia de relaciones de generalización/especialización entre clases.
- Las clases se disponen en una jerarquía, donde una clase hereda todos los atributos y operaciones de las clases superiores en la jerarquía.
- Una clase puede tener sus propios atributos y operaciones adicionales a lo heredado.
- Una clase puede modificar los atributos y operaciones heredadas.

Relación de herencia

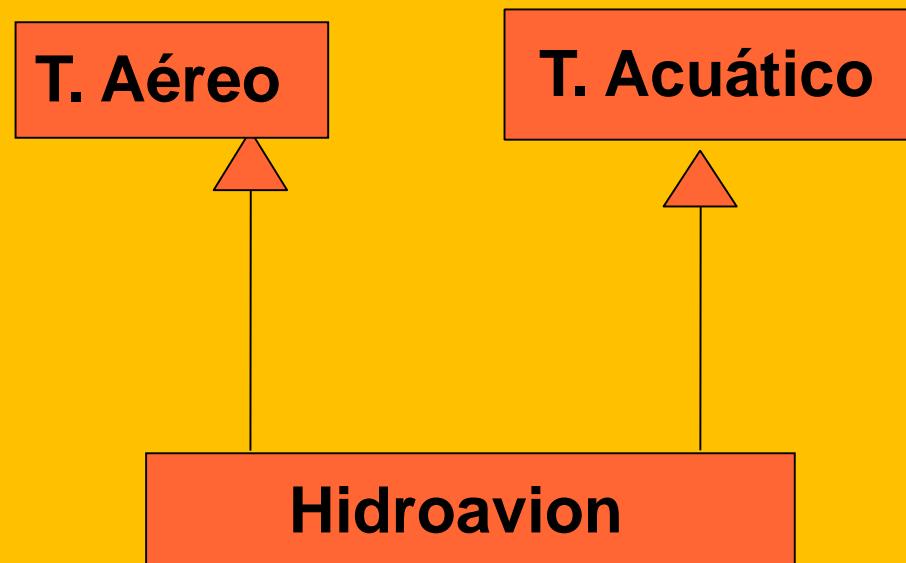
-  Las clases por encima en la jerarquía a una clase dada, se denominan superclases.
-  Las clases por debajo en la jerarquía a una clase dada, se denominan subclases.
-  Una clase puede ser superclase y subclase al mismo tiempo.
-  Tipos de herencia:
-  Simple.
-  Múltiple (no soportada por todos los lenguajes O.O.)

Ejemplo



Herencia Múltiple

- Herencia Múltiple: una clase derivada puede heredar de una o más clases base (C++ es un ejemplo de lenguaje que soporta este tipo de herencia)



Relación dinámica: mensaje

- Un mensaje es un comando o petición que se le envía a otro objeto.
- Requiere el conocimiento previo del interfaz del objeto receptor.
- Que es dinámica significa que se observa en ejecución, no en el diseño.

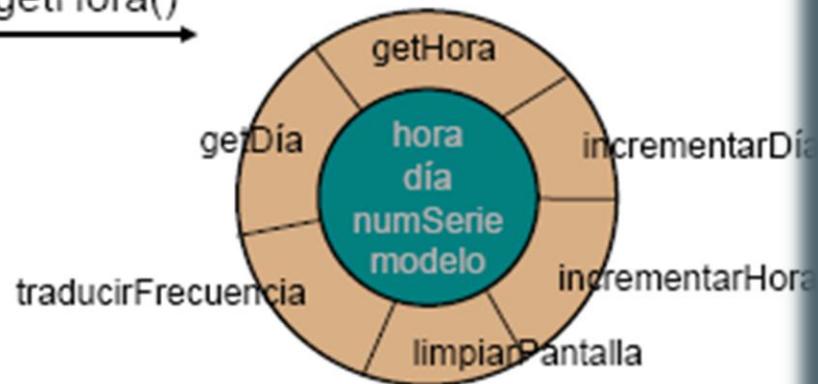
Ejemplo



unaPersona



unReloj

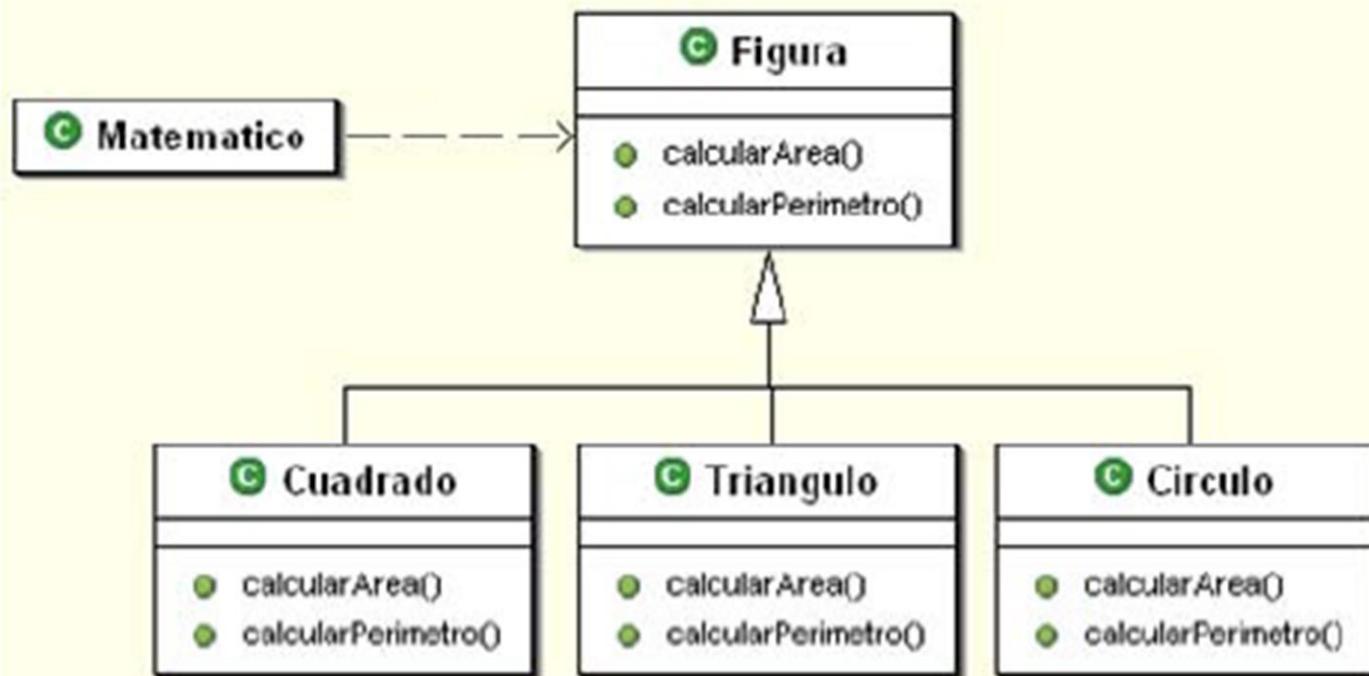


unReloj.getHora()

Polimorfismo

- Permite implementar múltiples formas de un mismo método, dependiendo cada una de ellas de la clase sobre la que se realice la implementación.
- Esto posibilita desencadenar operaciones diferentes en respuesta a un mismo mensaje, en función del objeto que lo reciba.

Ejemplo



**Analizar un problema
Mediante el:
Análisis
Orientado a Objetos**

La Universidad de La Punta ha contratado a tu empresa para que desarrolle un software que permita registrar la inscripción de Alumnos a las distintas Materias ofrecidas en cada una de las Carreras disponibles.

Pero sólo nos piden modelar a los alumnos de los cuales les interesa registrar, nro de legajo, nro de documento, apellido, nombre, edad, fecha de nacimiento, domicilio y mail.

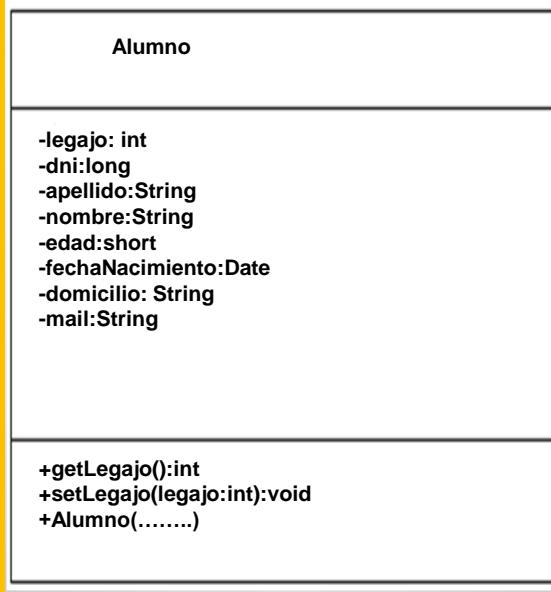
Siempre implementando el concepto de encapsulamiento.

Posibles atributos y operaciones para el caso de estudio “Registro a Materias ULP”

Alumno

-legajo: int
-dni:long
-apellido:String
-nombre:String
-edad:short
-fechaNacimiento:Date
-domicilio: String
-mail:String

+getLegajo():int
+setLegajo(legajo:int):void
+Alumno(.....)



Molde, Blueprint ...

Instancia, Objeto

Para el Video Juegos “Carrera Mortal”, nos piden modelar un Auto con las siguientes características y comportamiento para poder ser anexado al proyecto.

El auto debe tener: color, patente y combustible con una carga inicial de 50lts.

El auto sólo tendrá como comportamiento:

Avanzar: Al que le indicaremos la cantidad de metros que deseamos avance y deberemos tener en cuenta que por cada 10 mts consume 1lt de combustible y solo podrá avanzar si hay combustible suficiente.

Retroceder: Al que le indicaremos la cantidad de metros que deseamos retroceda y deberemos tener en cuenta que por cada 10 mts consume 1lt de combustible y solo podrá avanzar si hay combustible suficiente.

LlenarTanque: Llenará el tanque de este auto nuevamente con 50lts de combustible.