ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS CON UML

(Parte I)

Ing. Luis Zuloaga Rotta

Por qué el software es inherentemente complejo?

- Complejidad del dominio del problema.
- Dificultad de administrar el desarrollo de procesos.
- Problemas al caracterizar el comporta-miento de sistemas discretos.
- Flexibilidad posible a través del software.

Complejidad del Dominio del Problema

- Los usuarios y los desarrolladores tienen perspectivas diferentes sobre la naturaleza del problema y realizan distintas suposiciones sobre la naturaleza de la solución.
- Una complicación adicional frecuente es el cambio de los requerimientos del software del sistema.

Dificultad al Gestionar el proceso de Desarrollo

- La tarea fundamental del equipo desarrollo es dar vida a una ilusión de simplicidad y defender a los usuarios de una vasta y a menudo arbitraria complejidad.
- Asociado al tamaño del proyecto esta el equipo de desarrollo; un mayor número de miembros implica una comunicación mas compleja y difícil.

Consecuencias de la Complejidad ilimitada

- Cuanto más complejo sea el sistema, más abierto está al derrumbamiento total.
- Nuestro fracaso en dominar la complejidad del software lleva a proyectos retrasados, que exceden el presupuesto, y que son deficientes respecto a los requerimientos fijados.

Problemas al caracterizar el comportamiento de Stmas. Discretos

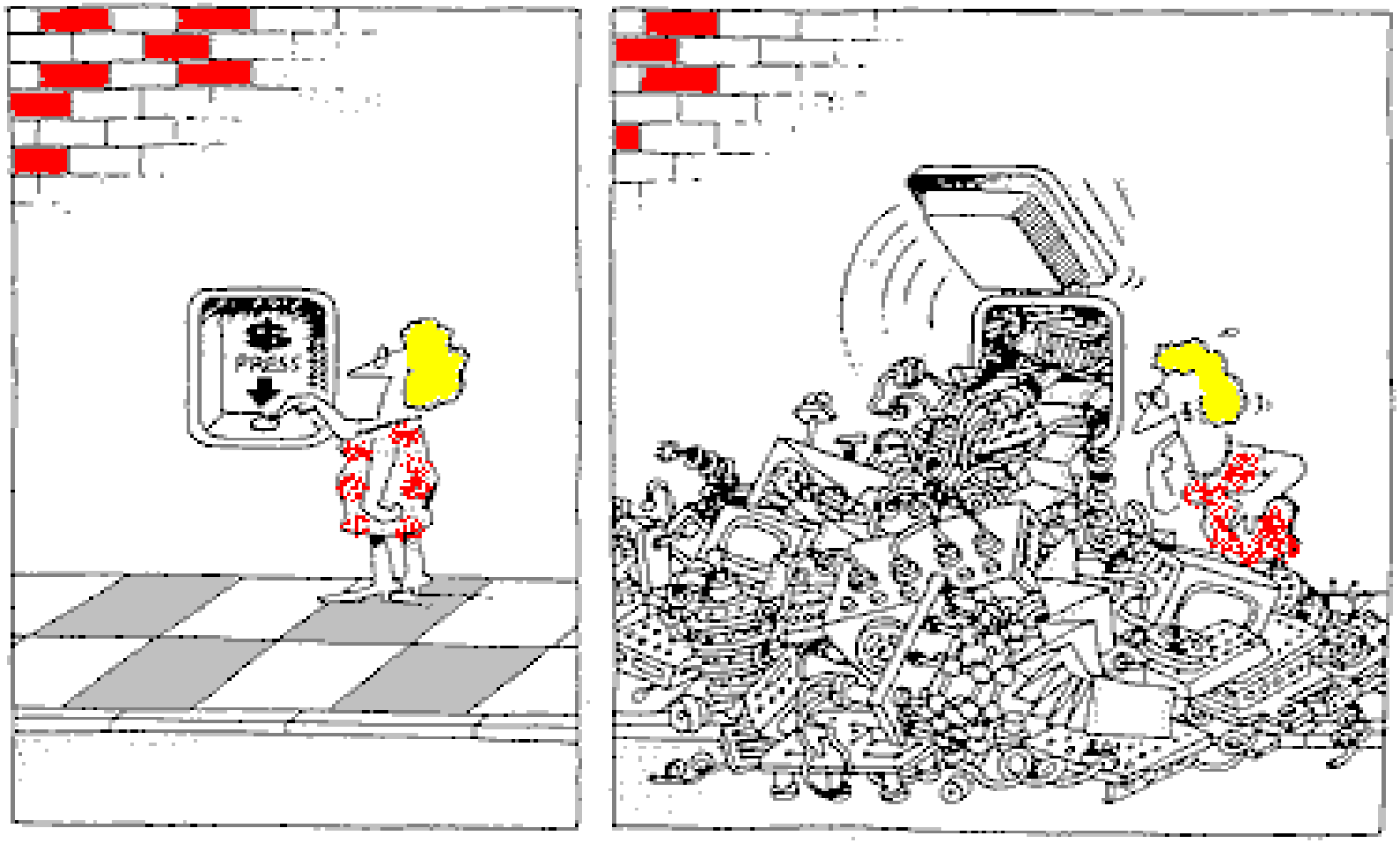
- Al ejecutarse el software en computadoras digitales, se tiene un stma. con estados discretos.
- Los stmas. discretos por su naturaleza tienen un nro. finito de estados posibles.
- Un evento externo puede corromper el estado del stma., porque sus diseñadores olvidaron tener en cuenta ciertas interacciones entre eventos.

Flexibilidad a través del software

- El software ofrece la máxima flexibilidad, por lo que un desarrollador puede expresar casi cualquier clase de abstracción.

Consecuencias de la Complejidad ilimitada

- Cuanto más complejo sea el sistema, más abierto está al derrumbamiento total.
- Nuestro fracaso en dominar la complejidad del software lleva a proyectos retrasados, que exceden el presupuesto, y que son deficientes respecto a los requerimientos fijados.



La tarea del equipo de desarrollo de software OO es ofrecer una ilusión de simplicidad

Qué es Orientado a Objetos ?

- En 1er lugar es una forma de pensar - una forma de modelar una solución a un problema.
- En 2do lugar, es una extensión a las metodologías de desarrollo previas, semejantes a la programación estructurada.
- En 3er lugar la orientación a objetos reconoce que los procesos naturales de pensamiento humano obtienen muchas ventajas evolucionarias, y por lo tanto trata de darle un adecuado soporte.

Por qué la orientación a objetos ?

- Por la estabilidad del modelado respecto a las entidades del mundo real.
- Por la construcción iterativa facilitada por el acoplamiento débil entre componentes.
- Por la posibilidad de reutilizar elementos entre desarrollos, y
- Por la simplicidad del modelado en base a 5 conceptos fundamentales

  (objetos, mensajes, clases, herencia y polimorfismo).

Qué propone la Orientación a Objetos ?

- A diferencia del método tradicional de la descomposición funcional, en la que hay que descomponer para comprender y componer para construir,
- Propone un método de descomposición, no basado únicamente en lo que hace el sistema, sino más bien en la integración de lo que el sistema es y hace.

Objetivos de la orientación a Objetos

- El modelado de las propiedades estáticas y dinámicas del entorno en el que se definen las necesidades, y que es llamado el ámbito del problema.
- Formalizar nuestra percepción del mundo y de los fenómenos que se dan en él, logrando corresponder el espacio del problema con el espacio de la solución, preservando la estructura y el comportamiento del sistema analizado.

La fortaleza de la Orientación a Objetos

- Su capacidad de agrupar lo que se ha separado, construir lo complejo a partir de lo elemental y, sobre todo, de integrar estáticamente y dinámicamente los constituyentes de un sistema.

OO - Revolución Industrial Software

- Describe el movimiento hacia una era donde el software es compilado a partir de objetos componentes reusables.
- Los componentes son internamente muy complejos pero simples para interactuar con ellos. Serán cajas negras.
- Es necesario progresar desde una era de paquetes de software monoliticos, donde un proveedor construye el software total, a una era en la que el software es ensamblado desde componentes y paquetes de muchos proveedores, similar a la forma en que se ensamblan autos y computadoras.

Cuáles son los beneficios de OO ?

- Reuso
- Calidad
- Modelamiento mundo real
- Resistencia a los cambios (fácil mantenimiento)

Qué es un objeto ?

- ’’ Un objeto tiene estado, comportamiento, e identidad; la estructura y comportamiento de objetos similares son definidos en su clase común; los terminos instancia y objeto son intercambiables ’’.

  Grady Booch ’91

… Otro Concepto

- ’’ Un objeto es cualquier cosa a la que se le aplica un concepto, el que representa una idea o noción que nosotros compartimos y aplicable a ciertos objetos en nuestro conocimiento ’’.

James Martin ’92

… Otro Concepto

- ’’ Nosotros definimos un objeto como un concepto, abstracción o cosa con un significado y límites bien definidos para el problema a manejar ’’.

  Rumbaugh ’91

Los Objetos

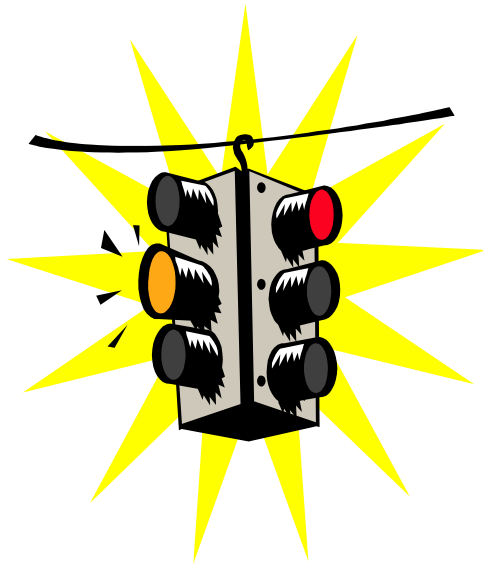
- El objeto es una unidad atómica formada por la unión de un estado y de un comportamiento.
- El objeto revela su verdadero papel y su verdadera responsabilidad cuando, por medio del envío de mensajes, se inserta es un escenario de comunicación.

Características de un Objeto

- **OBJETO = ESTADO**
  + COMPORTAMIENTO
  + IDENTIDAD

El Estado

- Agrupa los valores instantáneos de todos los atributos de un objeto sabiendo que un atributo es una información que cualifica al objeto que la contiene.
- El estado evoluciona con el tiempo, es variable y puede verse como la consecuencia de sus comportamientos pasados.



El Comportamiento

- Agrupa todas las competencias de un objeto y describe sus acciones y reacciones.
- Cada átomo de comportamiento se llama operación.
- Las operaciones de un objeto se desencadenan a consecuencia de un estímulo externo, representando en forma de un mensaje enviado por otro objeto.

Estado y Comportamiento

- El estado y el comportamiento están relacionados: el comportamiento en un instante dado depende del estado actual, y el estado puede ser modificado por el comportamiento.

  “ Sólo es posible hacer aterrizar un avión si este está volando ....”

La Identidad

- Distingue los objetos de forma no ambigua, independientemente de su estado.
- Distingue dos objetos en los que todos los valores de atributos son idénticos.
- La identidad es un concepto, no se representa de manera específica en el modelado. Cada objeto posee una identidad de manera implícita.

Oid (Object Identifier)

- Cada objeto posee un oid. El oid establece la identidad del objeto y tiene las siguientes características:
  - Constituye un identificador único y global para cada objeto dentro del sistema.
  - Es determinado en el momento de la creación del objeto.
  - Es independiente de la localización física del objeto, es decir, provee completa independencia de localización.

... Oid (Object Identifier)

- Es independiente de las propiedades del objeto, lo cual implica independencia de valor y de estructura
- No cambia durante toda la vida del objeto. Además, un oid no se reutiliza aunque el objeto deje de existir
- No se tiene ningún control sobre los oids y su manipulación resulta transparente

- Sin embargo, es preciso contar con algún medio para hacer referencia a un objeto utilizando referencias del dominio (valores de atributos).

Categorías de comportamiento

- Los objetos interactúan para realizar las funciones de la aplicación.
- Según la naturaleza de las interacciones, es decir, según la dirección de los mensajes intercambiados, es posible describir de manera general el comportamiento de los objetos.
- Las tres categorías de comportamiento: los actores, servidores y agentes.

Los Objetos Actores

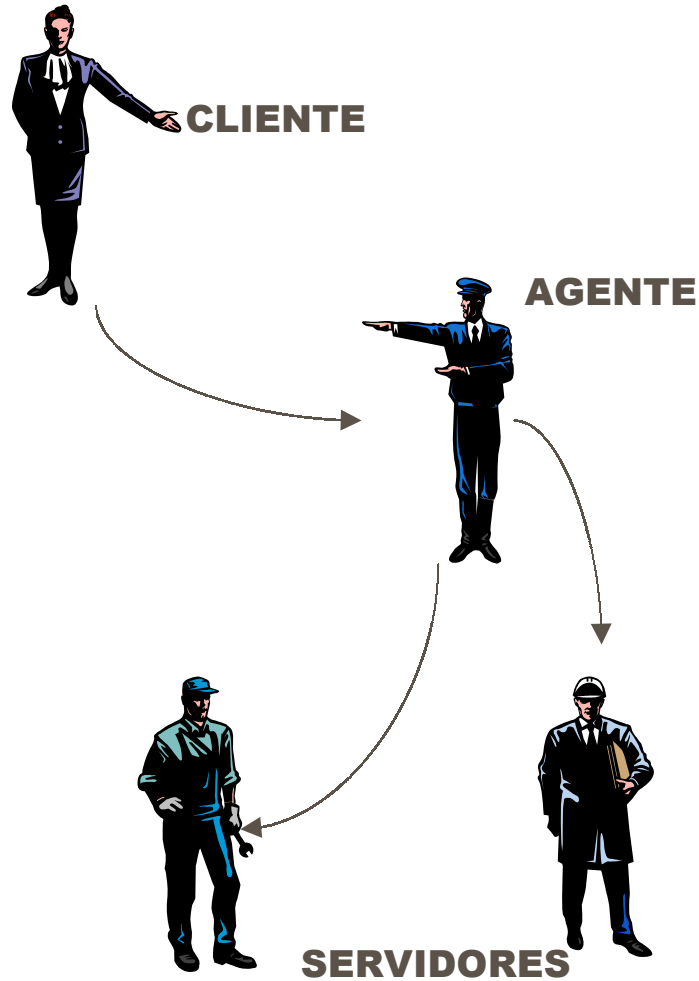
- Los actores son siempre objetos en el origen de una interacción.
- Son generalmente objetos activos, es decir, poseen un hilo de ejecución (thread) y son quienes pasan el testigo a los otros objetos.

Los Objetos Servidores

- Los servidores, por el contrario, nunca son el origen de una interacción, son siempre destinatarios de los mensajes.
- A menudo son objetos pasivos que esperan que otro objeto requiera de sus servicios. En ese caso, el flujo de control se pasa al servidor desde el objeto que envía el mensaje y se recupera tras la ejecución del servicio.

Los Objetos Agentes

- Los agentes reúnen las características de los actores y los servidores.
- Estos objetos pueden interactuar con los otros objetos en todo momento, ya sea por iniciativa propia o a consecuencia de una solicitud externa.
- Los agentes aislan a los objetos clientes de los objetos servidores, introduciendo una indirección en el mecanismo de propagación de los mensajes.



El concepto de mensaje

- La unidad de comunicación entre objetos se llama mensaje.
- El mensaje es el soporte de una relación de comunicación que vincula, de forma dinámica, los objetos que han sido separados por el proceso de descomposición.

Formas de sincronización de mensajes

- La sincronización precisa la naturaleza de la comunicación y las reglas que rigen el paso de mensajes.
- Existen 5 categorías de mensajes:
  - Simple
  - Síncrono
  - Esperado
  - Cronometrado
  - Asíncrono

Tipos de Objetos

- **Concretos**: Persona-Carro-PC
- **Roles**: Doctor-Propietario-Maestro
- **Relacional**: Sociedad-Hijo-Matrimonio
- **Eventos**: Venta-Compra-Arribo
- **Exponibles**: Icono-Window-Imagen-Menú

Actividades iniciales del Análisis OO

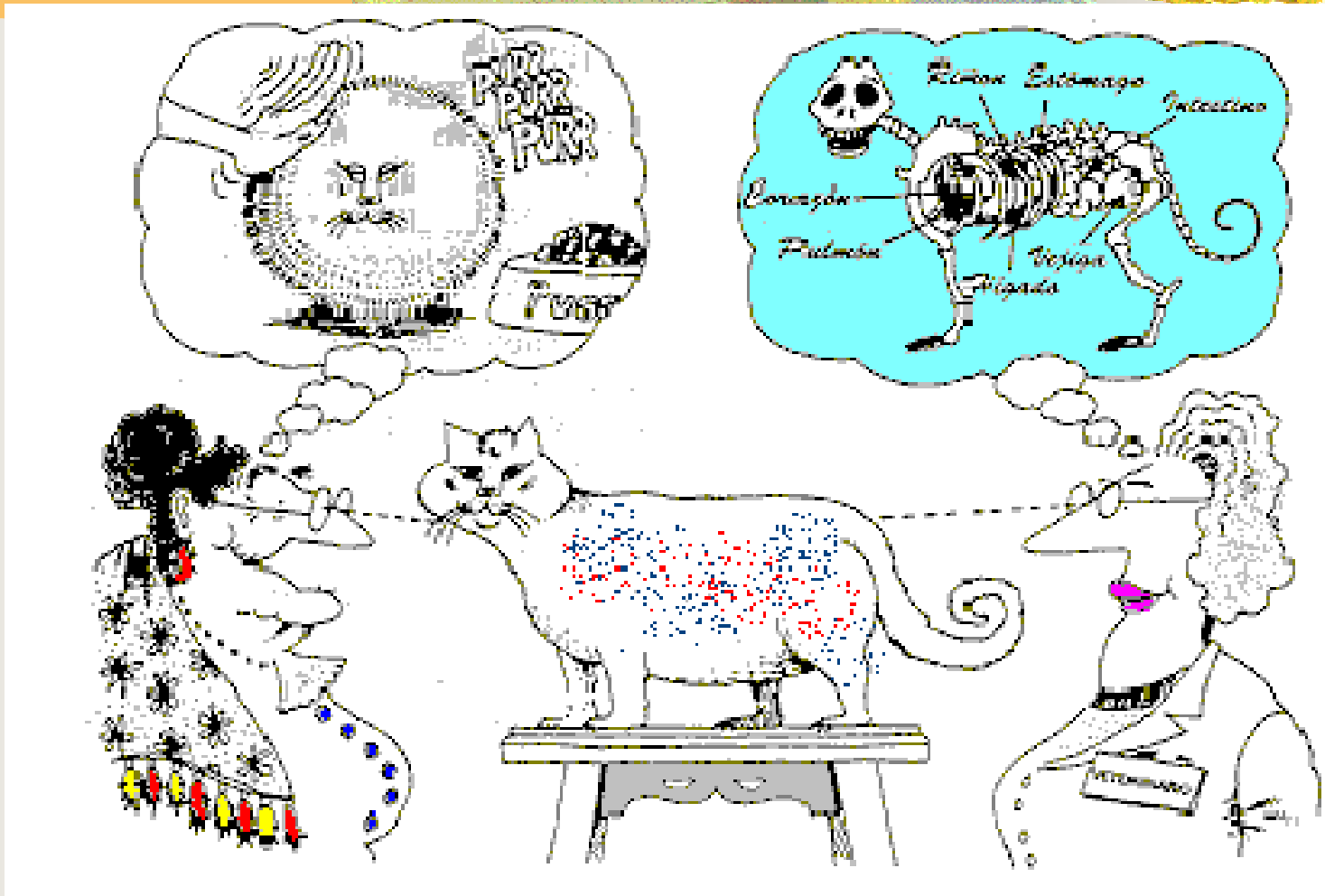
- Observar las cosas del sistema real y determinar sus propiedades :
  - Color
  - Tamaño
  - Forma
  - Genealogía
  - Relaciones a otras cosas.

Como identificar Objetos

- Siendo un objeto una abstracción de la realidad, y entendiendo que será una abstracción creada por un desarrollador para satisfacer necesidades de una aplicación, observe e identifique :
  - Cosas tangibles
  - Roles
  - Incidentes o eventos
  - Interacciones

Qué es la Abstracción ?

- Es una de las vías fundamentales por la que los humanos combatimos la complejidad.
- La abstracción surge del reconocimiento de las similitudes entre ciertos objetos, situaciones o procesos del mundo real, y en la decisión de concentrarse en esas similitudes e ignorar por el momento las diferencias.



La abstracción centra su atención en las características esenciales de un objeto en relación a la perspectiva del observador

La Abstracción

- Es una facultad de los seres humanos que consiste en la identificación de las características comunes a un conjunto de elementos, hacia la descripción condensada de estas características en lo que se ha convenido en llamar una **CLASE**.
- El método de abstracción es arbitrario. Lo que permite que un objeto pueda ser visto a través de abstracciones diferentes.

Descubrimiento e invención

- Con el descubrimiento se llega a reconocer las abstracciones utilizadas por expertos del dominio; si el experto del dominio habla de ella, entonces la abstracción suele ser importante.
- Mediante la invención, se crean nuevas clases y objetos que no son parte del dominio del problema pero que son útiles en el diseño e implantación.

Describiendo Objetos

- Después de identificar un objeto, se crea una abstracción para describir esta como un conjunto de atributos y operaciones. Los atributos almacenan valores de datos relativos a un objeto.
- Ejemplo de atributos :
  - Peso
  - Tamaño
  - Nro Identidad

Describiendo Objetos

- Operaciones o métodos son actividades llevadas a cabo por un objeto como resultado de un evento.

  Ejemplo de actividades :
  - Listar Facturas
  - Calcular Volumen
  - Totalizar Items
- Se describen relaciones a otros objetos.

  Ejemplo de relación :
  - Factura pertenece a Cliente

Atributo

- Es un valor de dato válido para los objetos en una clase. Diferentes instancias de objetos pueden tener el mismo o diferentes valores para un atributo dado.
- Cada nombre de atributo es único en una clase, regla que no necesariamente tiene que cumplirse para clases diferentes.
- Un atributo debería ser un valor de dato puro, no un objeto.

Seleccionando Atributos

- Definir atributos que son independientes de cualquier otro.
- Ejemplo de atributos dependientes :
  - Una locomotora tiene los siguientes atributos
    - Hecho en, y
    - Motor tipo
  - En el siguiente caso ellos son dependientes
    - Hecho en puede ser General Electric u Otro
    - Motor tipo es diesel si es hecho por GE, en cualquier otro caso es a gas.

Seleccionando Atributos ...

- La mejor solución es crear dos objetos:
  - Locomotora_GE, y
  - Otra_Locomotora
    - Locomotora_GE tiene motor diesel.
    - Otra_Locomotora tiene motor a gas.

Encontrar todos los Atributos Necesarios

- Los siguientes valores de datos de atributos no son útiles para identificar el objeto.
  - Tiene puertas
  - Tiene cilindros
  - Usado por personas
- Valores de datos que deben conocerse para ayudar a identificar un objeto:
  - Hecho por Ford
  - Tiene un motor de 300HP
  - Tiene seis cilindros

Operaciones y Métodos

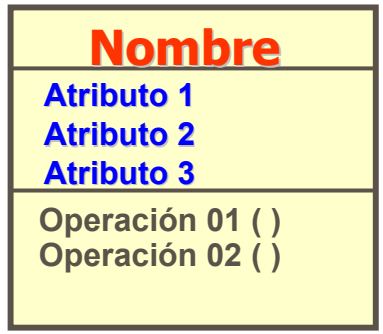
- Una operación es una función o transformación que puede ser aplicada a o por objetos en una clase.
- Cada operación tiene un objeto destino como un argumento implícito. La misma operación puede aplicarse a muchas clases diferentes.
- Un método es la implementación de una operación para una clase.

Ejemplo de Operación y método

- La clase FILE tiene una operación PRINT.
- Diferentes métodos podrían ser implementados para imprimir archivos ASCII, archivos Txt y archivos de imágenes digitalizadas (BMP).
- Todos estos métodos lógicamente ejecutan la misma tarea, imprimir un FILE; esto es, nos podemos referir a ellos por la operación genérica PRINT, sin embargo cada método puede ser implementado por un diferente grupo de líneas de código.

Representación Gráfica de las CLASES

- Cada clase se representa bajo la forma de un rectángulo dividido en tres compartimentos.
- El primero contiene el nombre de la clase, el segundo los tributos y el tercero las operaciones.



Las Clases

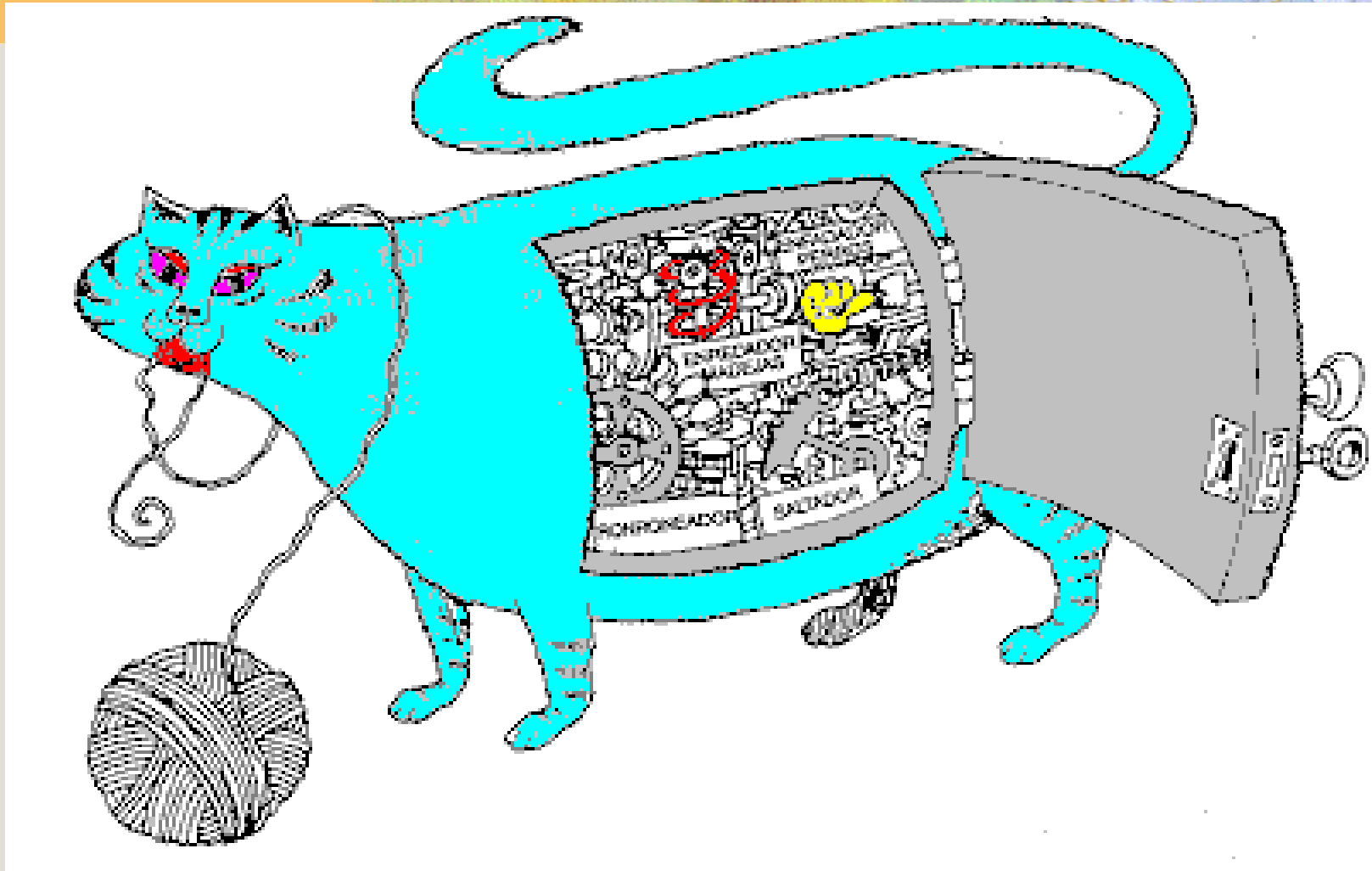
- La clase describe el ámbito de definición de un conjunto de objetos. Cada objeto pertenece a una clase.
- Las generalidades están contenidas en la clase y las particularidades están contenidas en los objetos.

Descripción de las CLASES

- La **especificación** : describe al ámbito de definición y las propiedades de las instancias de esta clase, y que corresponde a la noción de tipo tal como se define en los lenguajes de programación clásicos
- La **realización** : describe cómo se realiza la especificación y contiene el cuerpo de las operaciones y los datos necesarios para su funcionamiento.

Qué es Encapsulación ?

- La abstracción y el encapsulamiento son conceptos complementarios: la primera se centra en el comportamiento observable de un objeto, mientras el encapsulamiento se centra en la implementación que da lugar a este comportamiento.
- Para que la abstracción funcione la implementación debe estar encapsulada.



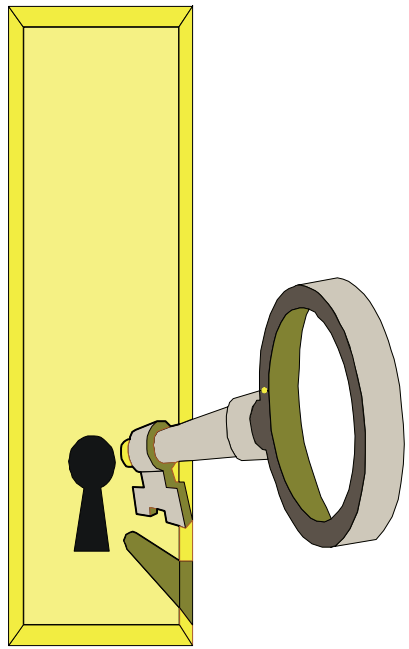
El encapsulamiento oculta los detalles de la implementación de un Objeto

Encapsulamiento

- De modo predeterminado, los valores de atributos de un objeto se encapsulan en el objeto y no puden ser manipulados directamente por los demás objetos.
- Todas las interacciones entre los objetos se efectúan invocando las diversas operaciones declaradas en la especificación de la clase y accesibles desde los demás objetos, de acuerdo a las reglas de visibilidad.

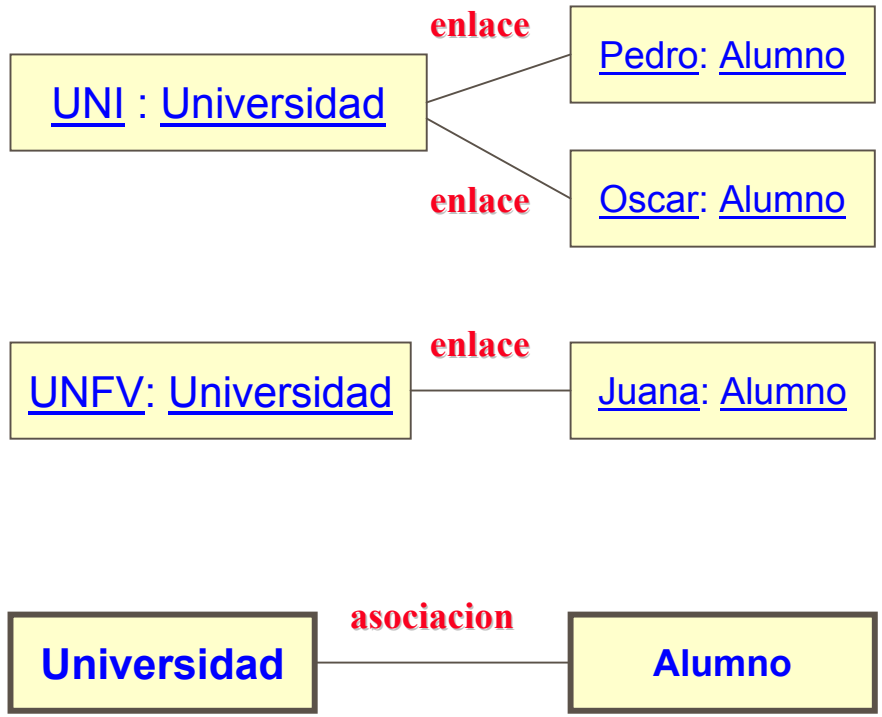
Reglas de Visibilidad

- **Privada** (-): visible sólo para la clase y para las clases amigas (C++).
- **Protegida**(#): visible sólo para las clases derivadas (subclases).
- **Pública** (+): visible para todas las clases con las que esta asociada.



La Asociación

- Expresa una conexión semántica bidireccional entre clases.
- Una asociación es una abstracción de los enlaces que existen entre los objetos instancias de las clases asociadas.



Clarificación de la Asociación

- Para mejorar la legibilidad de los diagramas, la asociación puede ir acompañada por una forma verbal activa o pasiva.
- Es posible precisar el rol de una clase al interior de una asociación.

Multiplicidad

- Los roles continen tambien una información de multiplicidad que precisa el número de instancias que participan en la relación.

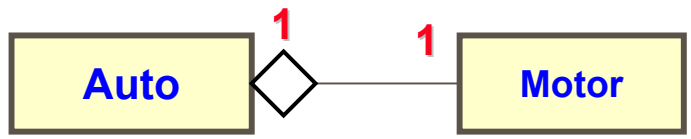
1	uno y sólo uno
0..1	cero o uno
m..n	de "m" a "n"
*	muchos
0..*	cero a muchos
1..*	uno a muchos

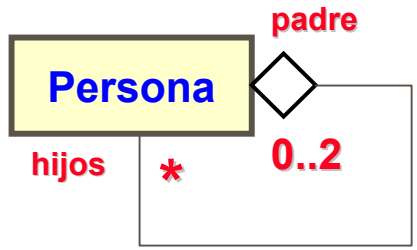
La Agregación

- Una relación expresa una forma de acoplamiento entre abstracciones.
- La fuerza del acoplamiento depende de la naturaleza de la relación en el ámbito del problema.
- De modo predeterminado, la asociación expresa un acoplamiento débil.
- La **agregación** es una forma particular de asociación que expresa un acoplamiento más fuerte entre clases.
- La **agregación** permite representar relaciones del tipo todo y partes o compuesto y componentes.

Representación de la agregación

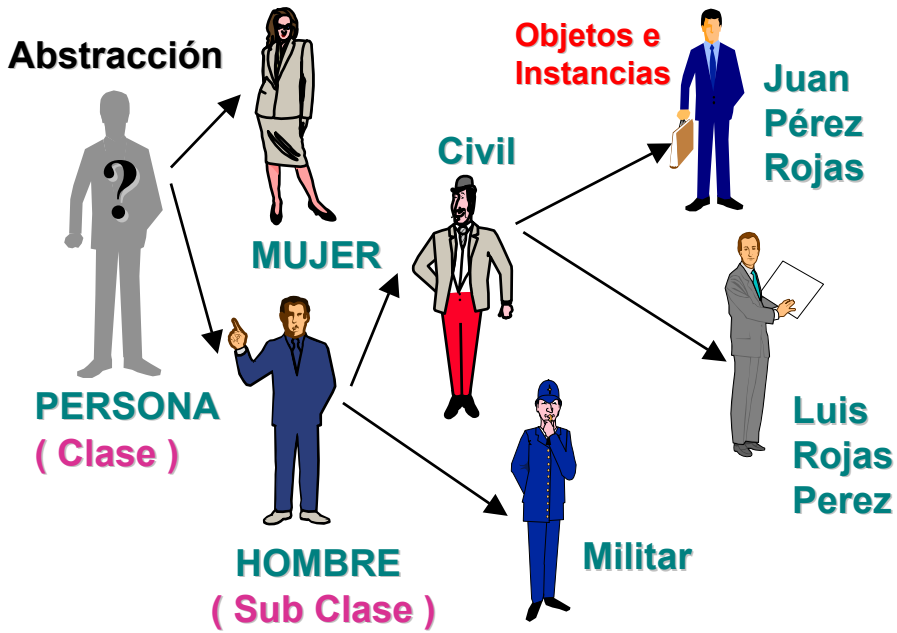
- Se representa como una asociación, con el añadido de un pequeño rombo colocado al lado del agregado.
- La agregación favorece la propagación de los valores de atributos y de operaciones del agregado hacia las partes o componentes.
- Cuando la multiplicidad del agregado vale 1, la destrucción del agregado entraña la destrucción de las partes.





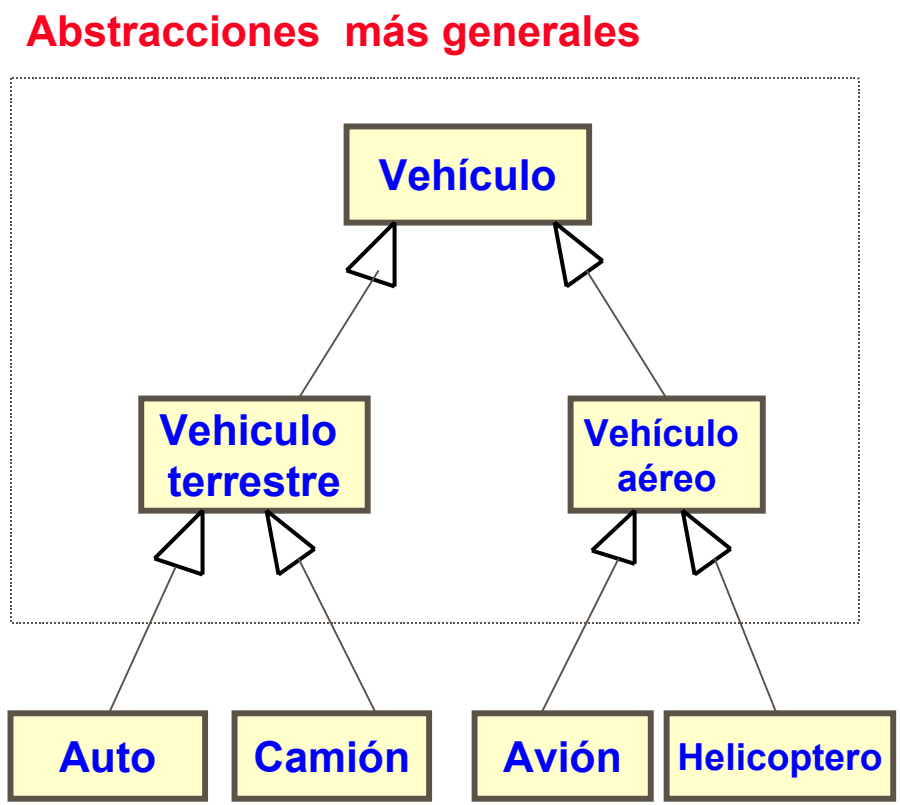
Las Jerarquías de CLASES

- Las jerarquías o clasificaciones permiten gestionar la complejidad ordenando los objetos dentro de árboles de clases de abstracción creciente.
- La generalización y la especialización son puntos de vista centrados en las jerarquías de clases.



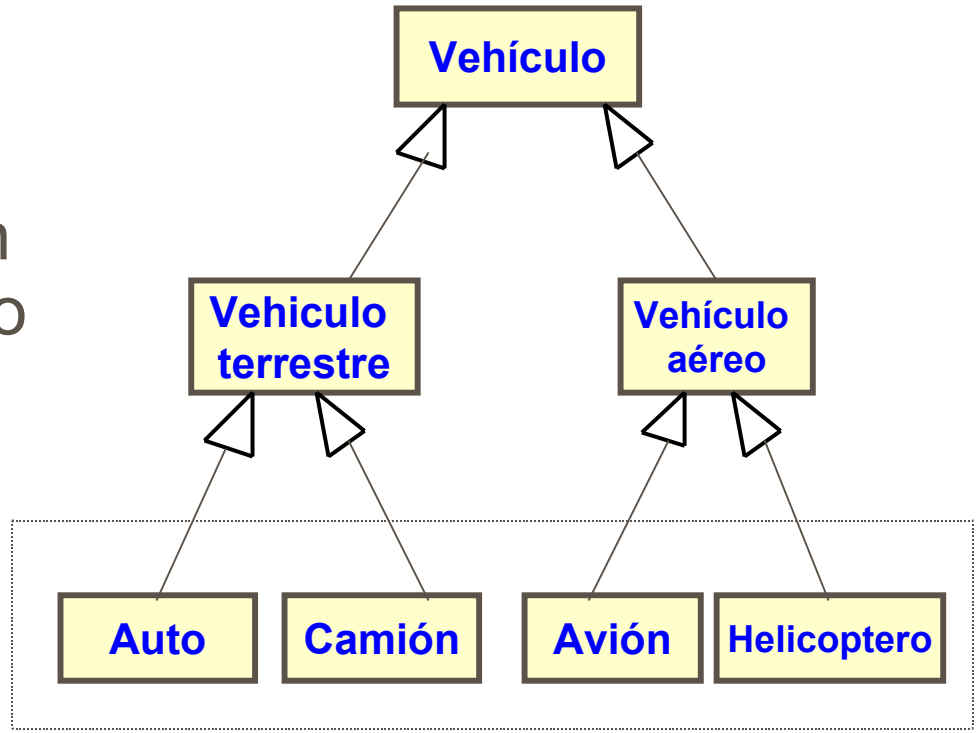
La Generalización

- Consiste en factorizar los elementos comunes (atributos, operaciones y restricciones) de un conjunto de clases en una clase más general llamada superclase.



La Especialización

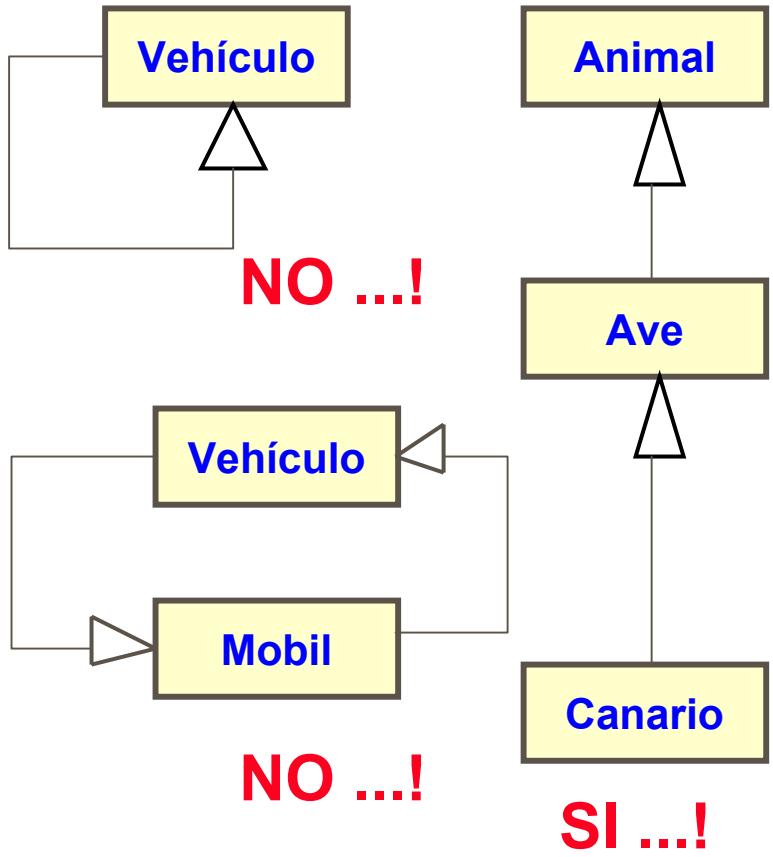
- Permite capturar las particularidades de un conjunto de objetos no discriminados por las clases ya identificadas.
- Las nuevas características se representan por una nueva clase, sub clase de una de las clases existentes.



Extensión por especialización

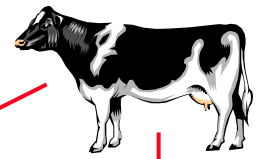
Propiedades de la Generalización

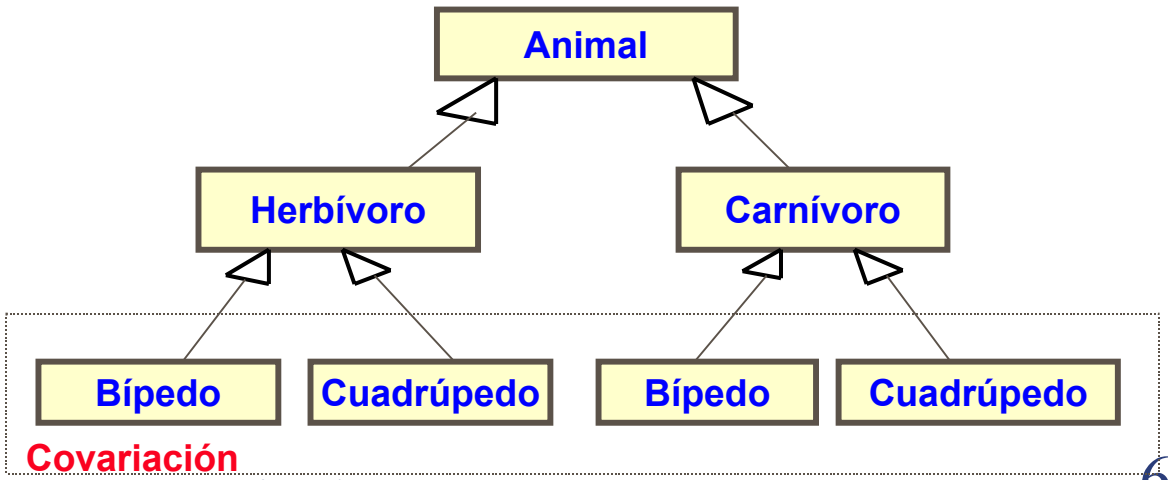
- La generalización solo afecta a las clases y significa : “es un” o “es una especie de”.
- No establece ninguna indicación de multiplicidad.
- No es una relación reflexiva.
- No es una relación simétrica
- Es una relación transitiva.



Criterio para la Generalización

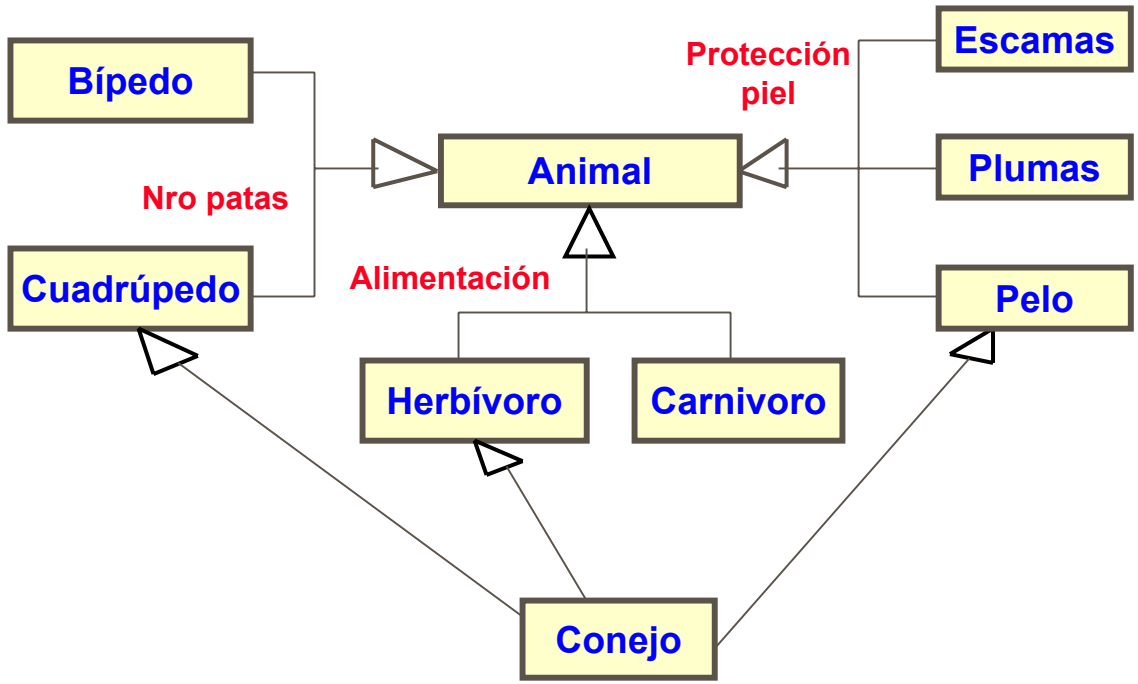
- De las estructuras planteadas en la clasificación de los animales mostradas, ninguna de las soluciones es satisfactoria, porque de acuerdo a los modelos propuestos el fenómeno de covariación induce puntos de mantenimiento múltiples.





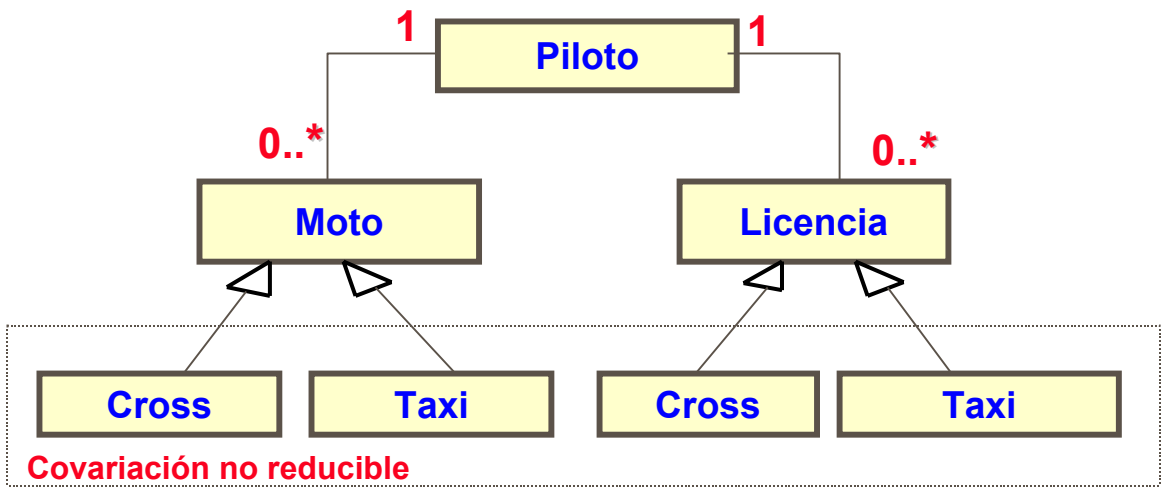
Generalización Múltiple

- Aporta una solución elegante para la construcción de clasificaciones con criterios independientes, díficiles de ordenar.

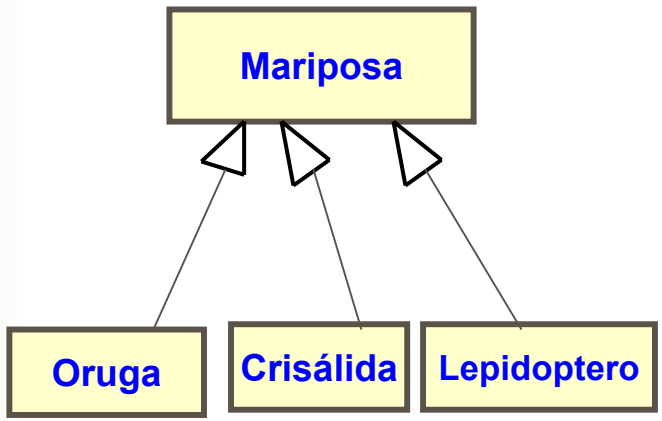


La covariación no reducible

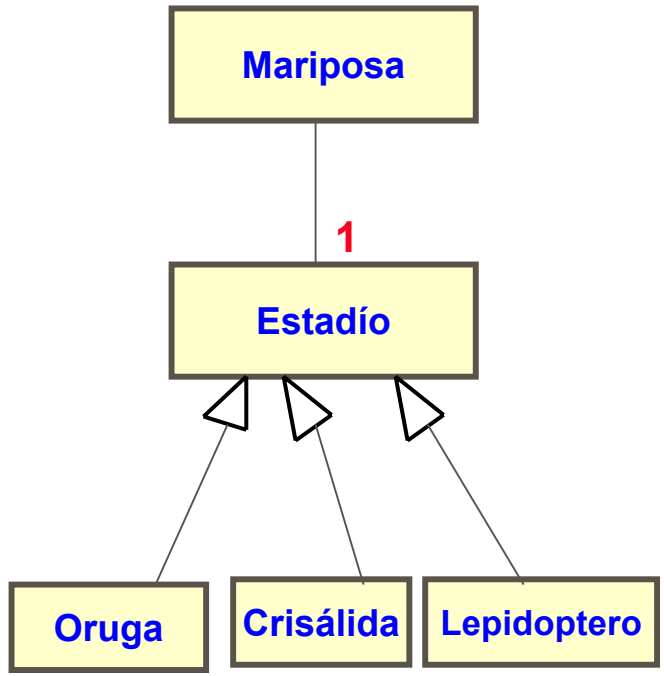
- En algunos casos la covariación existe por la propia naturaleza del ámbito de la aplicación.
- Esta forma de covariación no es reducible porque afecta a dos jerarquías distintas.



Como representaría una Metamorfosis ?



Es esto correcto ? **NO ...!**



Esto es Correcto..!

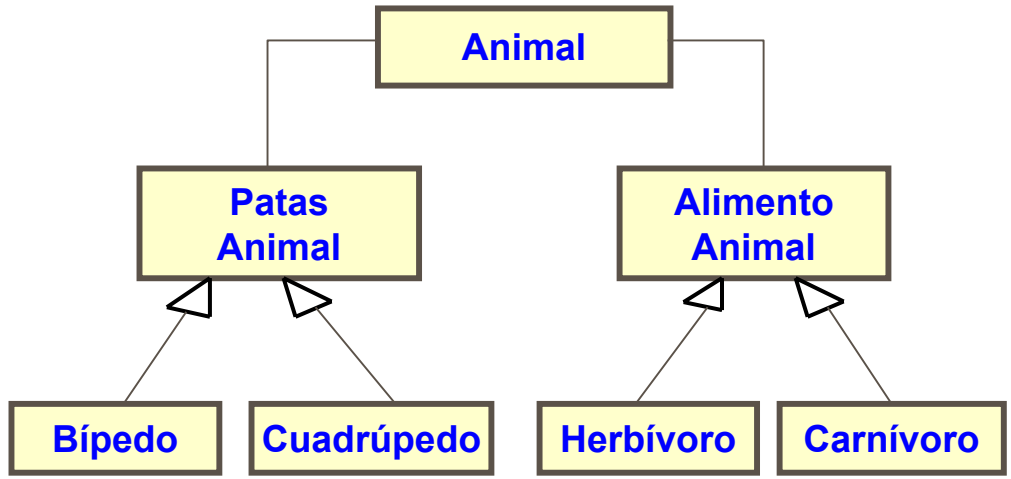
La generalización no está adaptada para representar las metamorfosis.

La Herencia

- Es una técnica de los lenguajes de programación para construir una clase a partir de una o varias otras clases, compartiendo atributos, operaciones y, en ocasiones, restricciones, dentro de una jerarquía de clases.
- Las clases hijos heredan las características de sus clases antecesoras; los atributos y las operaciones declaradas en la clase padre son accesibles en la clase hijo, como si se hubieran declarado localmente.

La Delegación en lugar de la herencia múltiple

- La herencia no es una necesidad absoluta y siempre puede reemplazarse por la delegación.



- La delegación presenta la ventaja de reducir el acoplamiento en el modelo: por una parte, el cliente no conoce directamente al proveedor, y por otra, el proveedor puede ser modificado sobre la marcha.

El Principio de la Sustitución

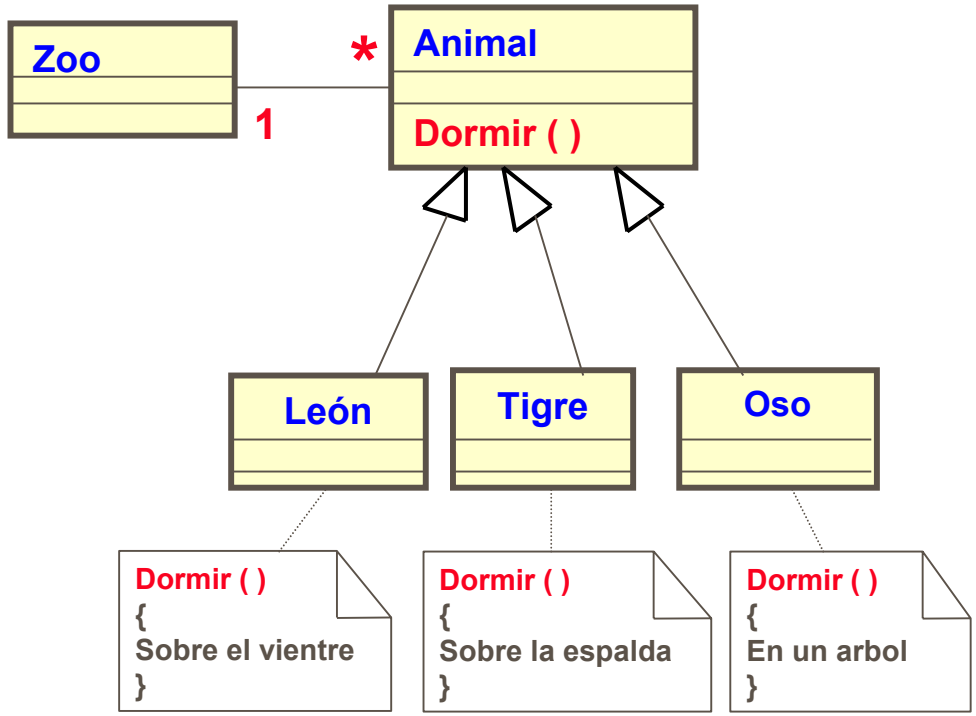
- La clasificación propaga el estado, el comportamiento y las restricciones. No existen medias tintas: todas las propiedades de la clase padre son válidas integramente en la clase hijo.
- El principio de la sustitución afirma que: "*Debe ser posible sustituir cualquier objeto instancia de una subclase por cualquier objeto instancia de una superclase sin que la semántica del programa escrito en los términos de la superclase se vea afectada*".

El Polimorfismo

- Describe la característica de un elemento que puede tomar varias formas, como el agua que se encuentra en estado sólido, líquido o gaseoso.
- El polimorfismo de operación, desencadena operaciones diferentes en respuesta a un mismo mensaje. Cada subclase hereda de la especificación de las operaciones de sus superclases, pero tiene la posibilidad de modificar localmente el comportamiento de estas operaciones.

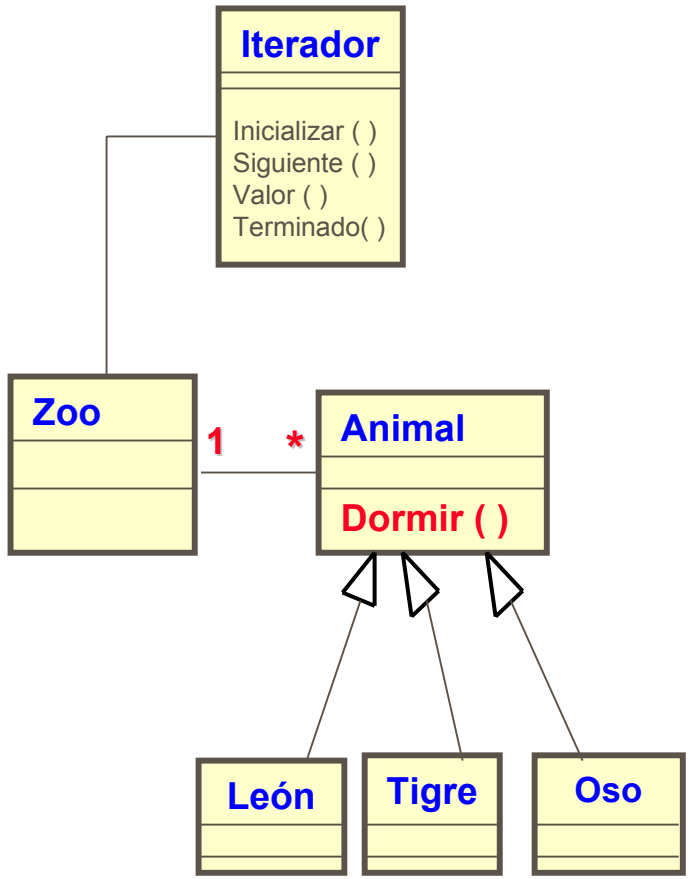
Objeto Iterador

- El diagrama siguiente representa una colección polimorfa. Todos los animales saben dormir, pero c/raza tiene su forma particular.
- Los mecanismos generales escritos según la especificación del ZOO no necesitan conocer los gustos particulares de cada raza para invocar la operación Dormir ( ).
- Un iterador es un objeto asociado a una colección que permite visitar todos sus elementos sin desvelar su estructura interna.



Operaciones del Iterador

- El iterador es activo cuando el control de la iteración se deja al usuario por medio de las 4 operaciones siguientes :
  - Inicializar, permite tener en cuenta los elementos presentes en un momento dado en la colección.
  - Siguiente, permite el paso al elemento siguiente.
  - Valor, que devuelve el elemento actual.
  - Terminado, que es verdad cuando todos los elementos han sido visitados.



Ejemplo de código utilizando Iterador

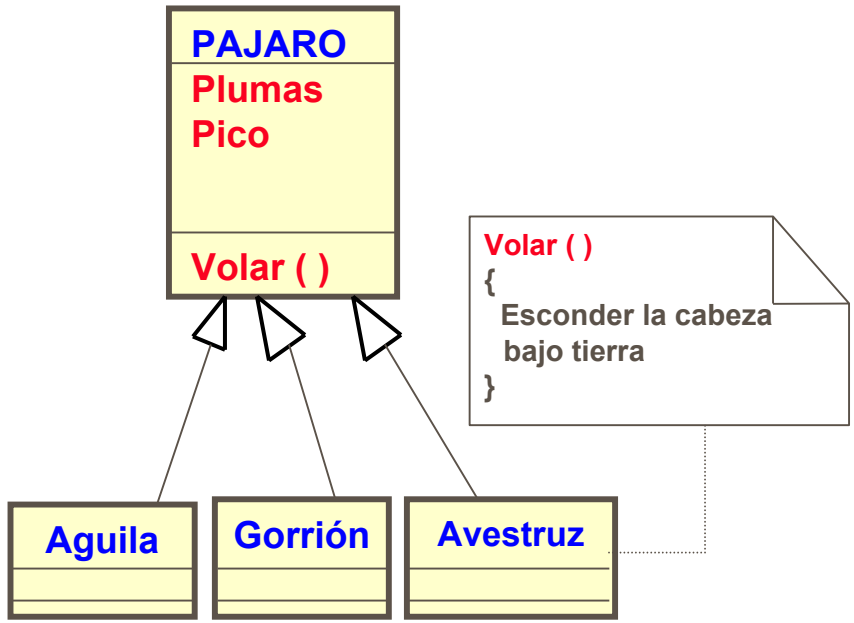
```
Visita: iterador;
UnAnimal: Animal; -- variable polimorfa
...
Visita.Inicializar (elZoo);
While not Visita.Terminado( )
loop
      UnAnimal:= Visita.Valor( );
      UnAnimal.Dormir( );
      Visita.Siguiente( );
end loop;
```

La variable **UnAnimal** es polimorfa: puede contener cualquier animal devuelto por la función **Visita.Valor( )**.

El envio del mensaje **Dormir( )** al animal contenido por la variable **UnAnimal** desencadena una manera específica de dormir que depende de la subclase del animal. El mecanismo que duerme a los animales del **Zoo** es independiente de los animales que se encuentran realmente en el **Zoo** en un momento dado

Violación del principio Sustitución

- El ejemplo precedente sólo funciona si cada Animal comprende el mensaje Dormir( ). Los mecanismos que implementan el polimorfismo manipulan objetos a través de las especificaciones de sus superclases.
- Para que el polimorfismo funcione efectivamente, es necesario que se verifique el principio de sustitución.



Lo mostrado no tiene consecuencias mientras nadie escriba un mecanismo general para manipular los pájaros.