Nociones de UML

Antonio Espín Herranz

UML

- **UML** = Unified Modeling Language.
  - Lenguaje de Modelado Unificado.

- **Un lenguaje estándar para escribir los planos del software**, de la misma forma que los arquitectos hacen primero los planos de la casa. **ESTÁ PENSADO PARA ENTRAR POR LOS OJOS**.

- UML puede utilizarse para:
  - **Visualizar**: es un lenguaje gráfico, unificado y facilita la comunicación.
  - **Especificar** → construir modelo precisos, no ambiguos y completos.
  - **Construir**: **NO** es un lenguaje de programación Visual pero se corresponde con lenguajes como Java, C++, Visual Basic, etc.
  - **Documentar**: Requisitos, Diseño, Arquitectura, etc.

- **Pretende establecer una notación estándar**.
- Actualmente va por la versión 2.5.x.
- Está pensado para sistemas con gran cantidad de Sotfware.

Bloques básicos en UML

- **Elementos**:
  - Son **abstracciones** que constituyen los ciudadanos de primera clase en un modelo.
    - En nuestro caso podría ser una **CLASE**.
- **Relaciones**:
  - Ligan los elementos entre sí.
- **Diagramas**:
  - Agrupan colecciones interesantes de elementos.

Elementos de UML

- **Estructurales**:
  - Son **partes estáticas** de un modelo, representan conceptos o cosas materiales. En general se denominan **clasificadores**.
    - **Son clases, interfaces**, un caso de uso …
    - Una clase se representa mediante un rectángulo indicando su nombre, atributos y operaciones.
      - La clase Persona, Empleado, etc.

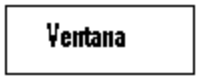
- **Comportamiento**:
  - Son las **partes dinámicas** del modelo, LOS VERBOS.
    - Interacción: comprende un conjunto de mensajes entre objetos.
      - dibujar, grabar, etc.
    - Máquinas de estados.

- **Agrupación**:
  - Son las partes organizativas de los modelos UML. Son las cajas en las que se puede descomponer un modelo.
    - **Los paquetes**: organizan el propio diseño.

- **Anotación**:
  - Son partes explicativas de los modelos de UML. Son comentarios llamados **notas**.

Elementos de UML (Gráficos)

- **Estructurales**:



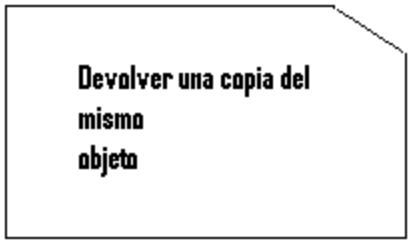
Cuando un sistema tenga muchas clases se pueden poner en forma abreviada.

Indicando solo el nombre de la Clase.

- **Agrupación**:

- **Comportamiento**:

- **Anotación**:

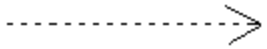


Relaciones en UML

- **Dependencia:**
  - Relación semántica entre dos elementos que un cambio en uno puede afectar al otro.
- **Asociación**:
  - Relación estructural entre clases (un todo y sus partes), son conexiones entre objetos.
    - La **agregación** es un tipo de asociación.
- **Generalización**:
  - También llamada especialización, la relación de herencia que se da entre una clase padre y la clase hija.
- **Realización**:
  - Relación Semántica entre Clasificadores. Un clasificador especifica un **contrato** que otro clasificador garantiza que cumplirá. Se pueden encontrar relaciones de realización: entre interfaces y las clases

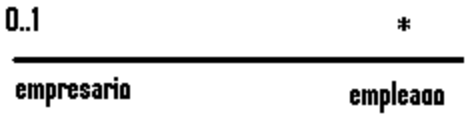
Relaciones en UML (Gráficos)

- **Dependencia**:

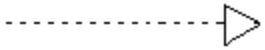


- **Generalización**:

- **Asociación**:



- **Realización**:



Diagramas en UML

- Un diagrama es la representación gráfica de un conjunto de elementos.

- Se visualiza como un grafo conexo de nodos (**Elementos**) y los arcos (**Relaciones**).

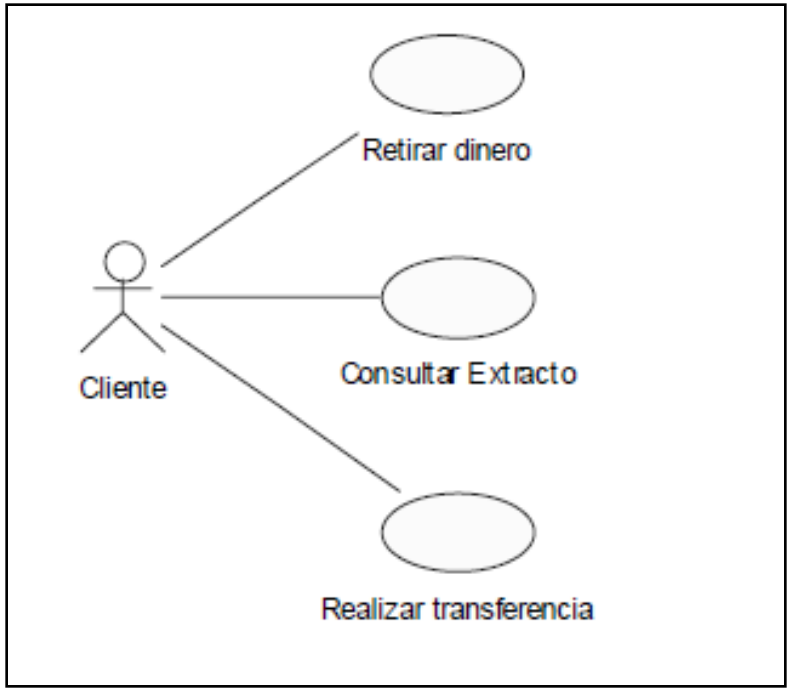
Tipos de diagramas en UML

- **DIAGRAMAS:**
  - **De Clases:** Muestra un conjunto de clases, interfaces y colaboraciones así como sus relaciones.
    - Son los mas comunes en el modelado de sistemas orientados a objetos. **VISTA ESTÁTICA DEL SISTEMA**.

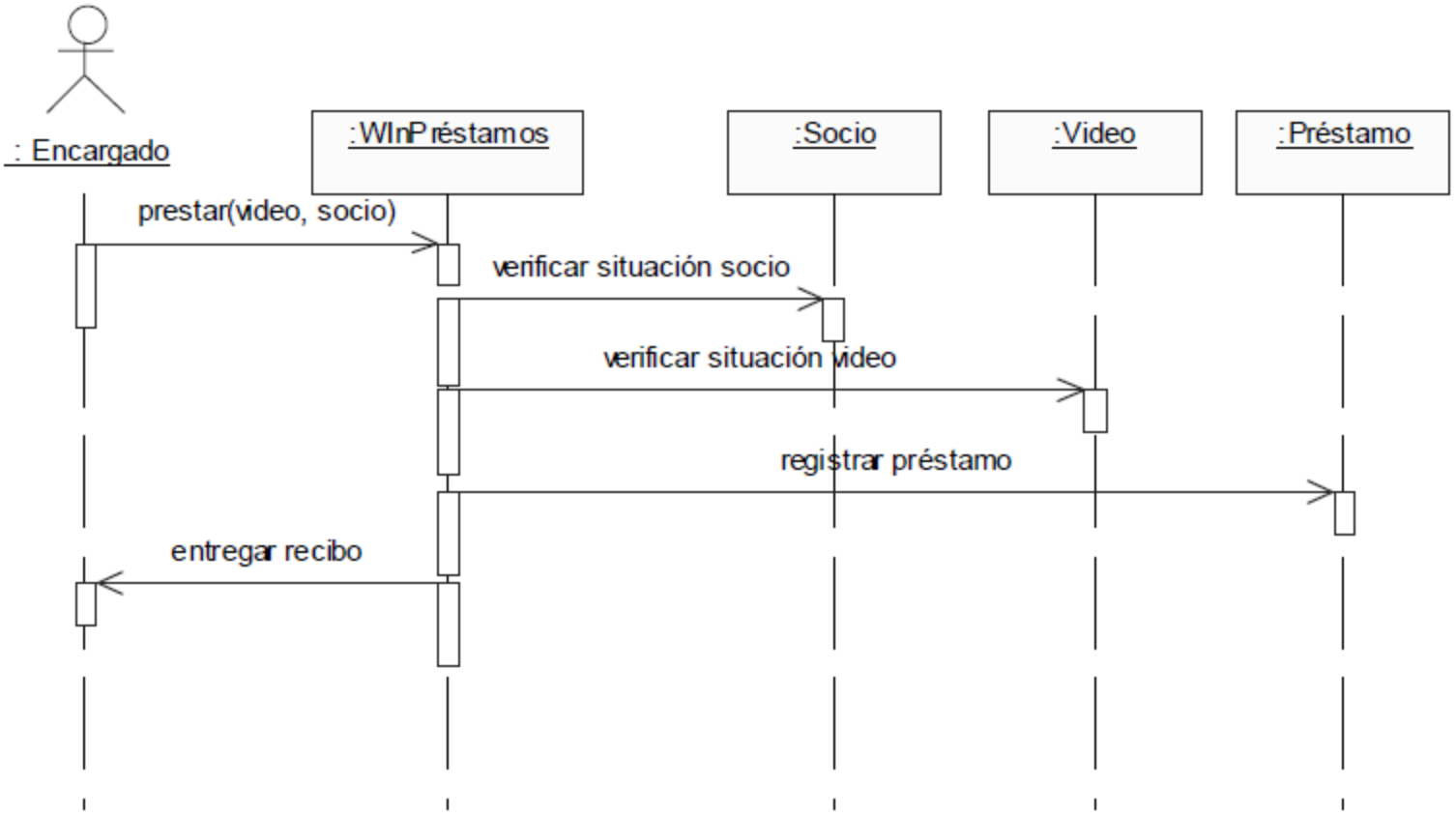
- **Otros diagramas de:**
  - Objetos, Paquetes, Componentes, Estructura compuesta, Casos de Uso, Secuencia, Comunicación, Estados, Actividades, Despliegue, Tiempos, Visión Global.

Ejemplo: Casos de Uso

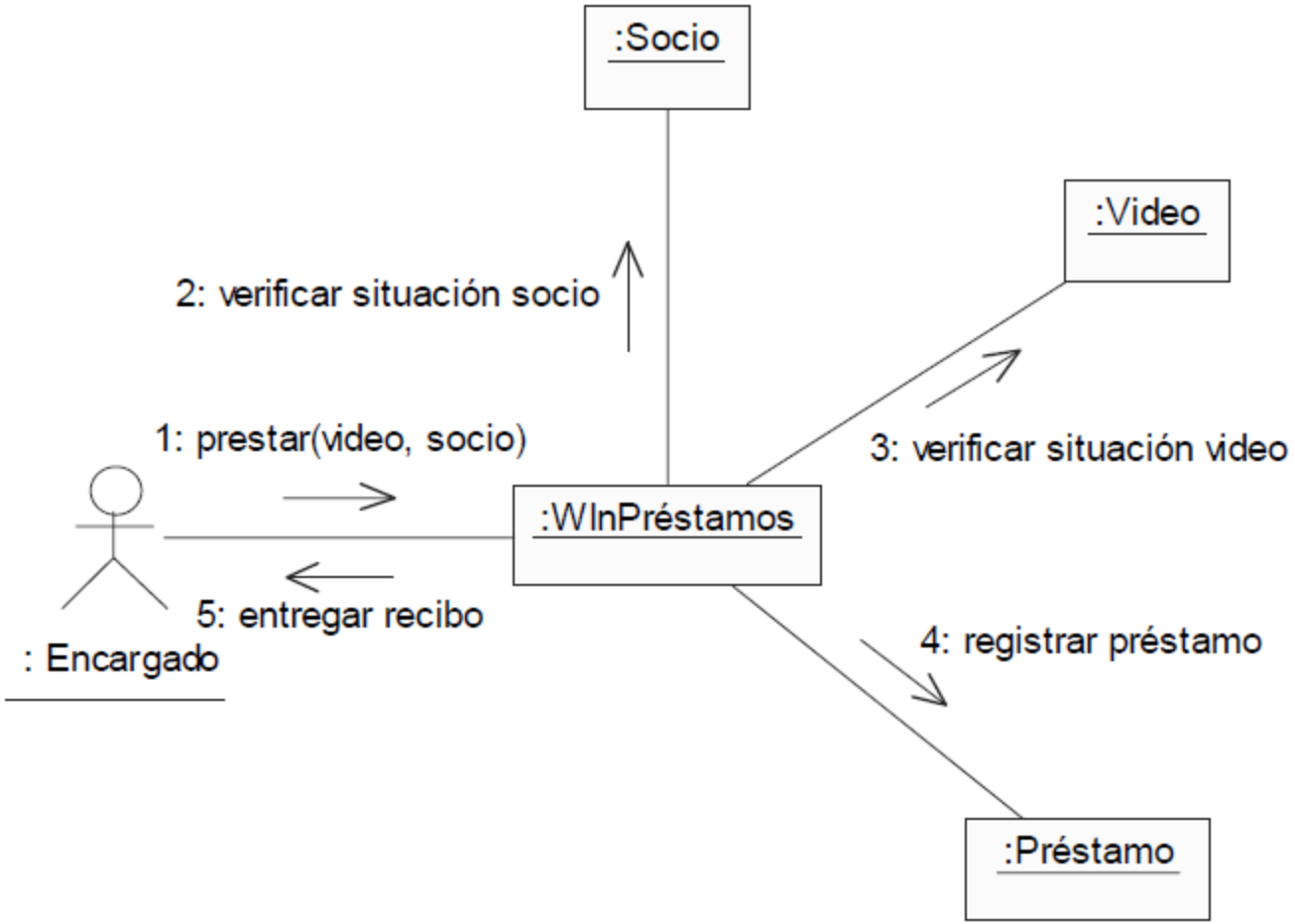
- Es una técnica para capturar información respecto de los servicios que un sistema proporciona a su entorno.
- No pertenece estrictamente al enfoque orientado a objeto, es una técnica para la captura y especificación de requisitos.



Ejemplo: Diagrama de Secuencia



Ejemplo: Diagrama de Colaboración



Diagramas de Clases

- El Diagrama de Clases es el diagrama principal para el análisis y diseño del sistema.
- Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia.
- La definición de clase incluye definiciones para atributos y operaciones.

Clasificación

- El mundo real puede ser visto desde abstracciones diferentes (subjetividad).

- Mecanismos de abstracción:
  - Clasificación / Instanciación
  - Composición / Descomposición
  - Agrupación / Individualización
  - Especialización / Generalización

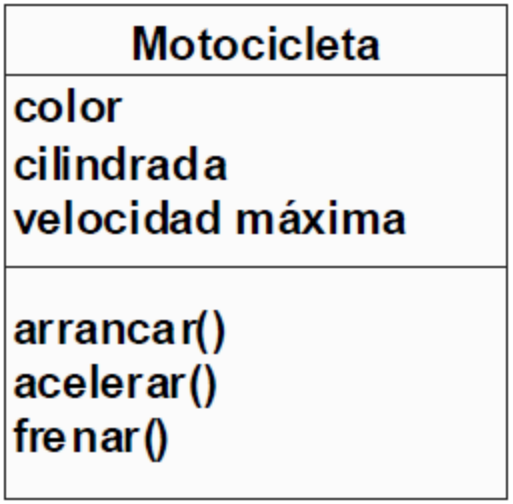
- La clasificación es uno de los mecanismos de abstracción más utilizados.

Clases

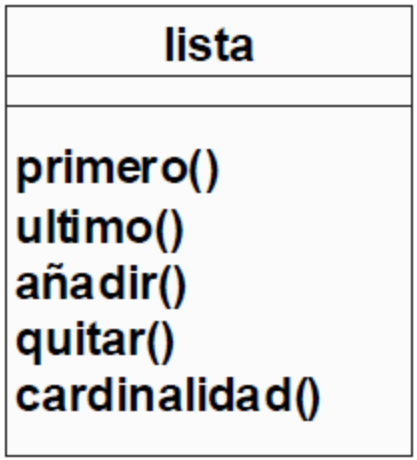
- La clase define el ámbito de definición de un conjunto de objetos.
- Cada objeto pertenece a una clase.
- Los objetos se crean por instanciación de las clases.

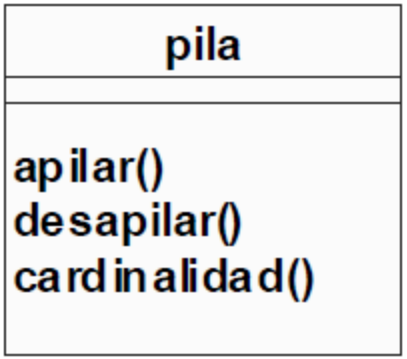
Clase: Notación Gráfica

- Cada clase se representa en un rectángulo con tres compartimientos:
  - nombre de la clase.
  - atributos de la clase.
  - operaciones de la clase.



Mas ejemplos





Según el número de clases que tenga el diagrama se pueden especificar mas o menos detalles:

Lo mínimo sería el nombre de la clase en un cuadro.

O se puede incluso especificar el nivel de acceso de los atributos y de los métodos.

Encapsulación

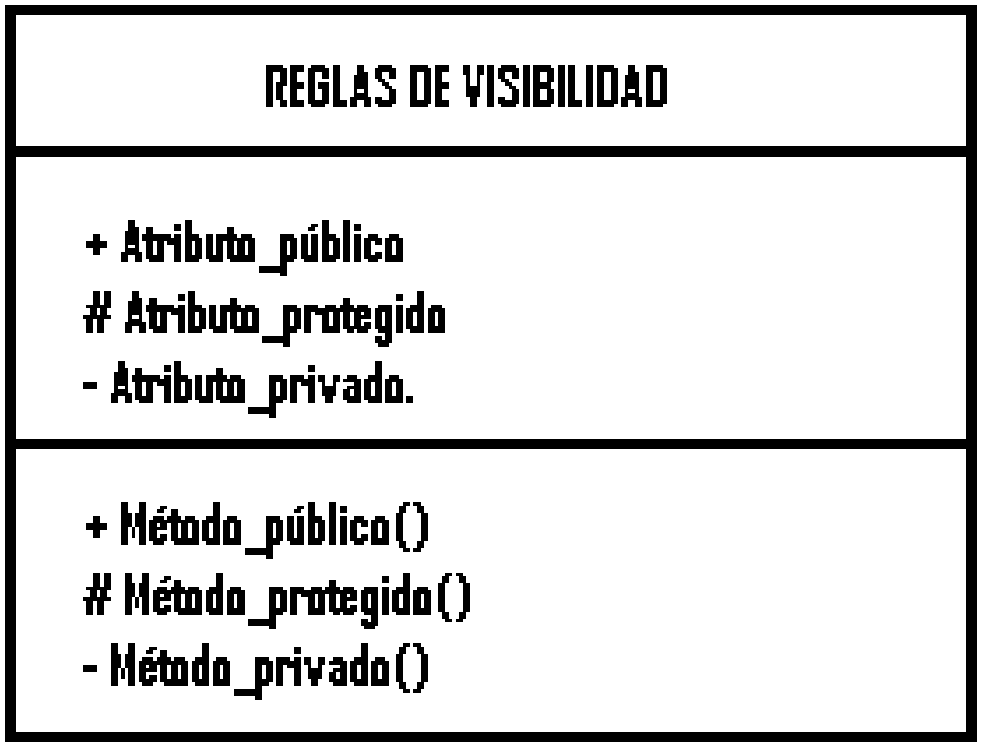
- La encapsulación presenta dos ventajas básicas:
  - Se protegen los datos de accesos indebidos.
  - El acoplamiento entre las clases se disminuye.
  - Favorece la modularidad y el mantenimiento.

- Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos.

Niveles de Encapsulación

- Los niveles de encapsulación están heredados de los niveles de C++:
  - **(-) Privado** : es el más fuerte. Esta parte es totalmente invisible (excepto para clases friends en terminología C++)
  - **(#)** Los atributos/operaciones **protegidos. E**stán visibles para las clases friends y para las clases derivadas de la original.
  - **(+)** Los atributos/operaciones **públicos** son visibles a otras clases (cuando se trata de atributos se está transgrediendo el principio de encapsulación)

Gráficamente



Relaciones entre Clases

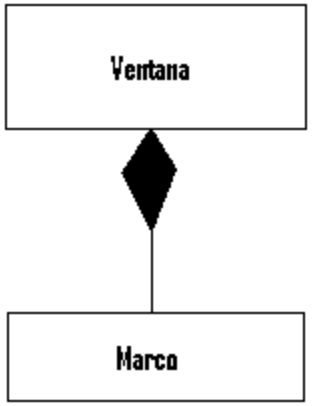
- Los enlaces entre de objetos pueden representarse entre las respectivas clases.

- Formas de relación entre clases:
  - Asociación y Agregación (vista como un caso particular de asociación).
  - Generalización/Especialización.
  - Dependencia.

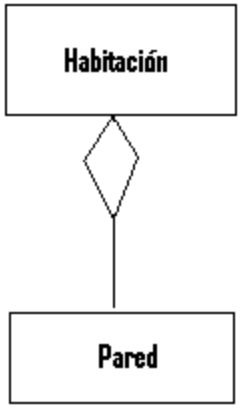
- Las relaciones de Agregación y Generalización forman jerarquías de clases.

Agregación

- La agregación simple → Composición.
  - Ejemplo, el marco de una ventana pertenece exactamente a una ventana. La destrucción de la ventana → la destrucción del marco.
  - Se representa por un rombo relleno de negro:



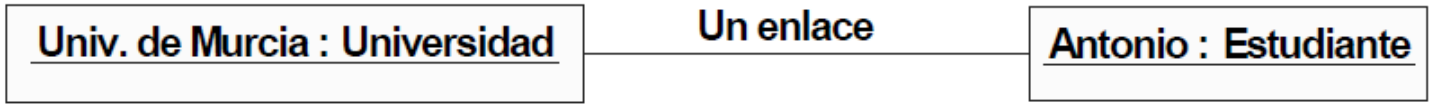
- La agregación compuesta:
  - Por ejemplo, una pared puede formar parte de una habitación pero a su vez puede formar parte de otras habitaciones. Se representa por medio de un Rombo vacío.

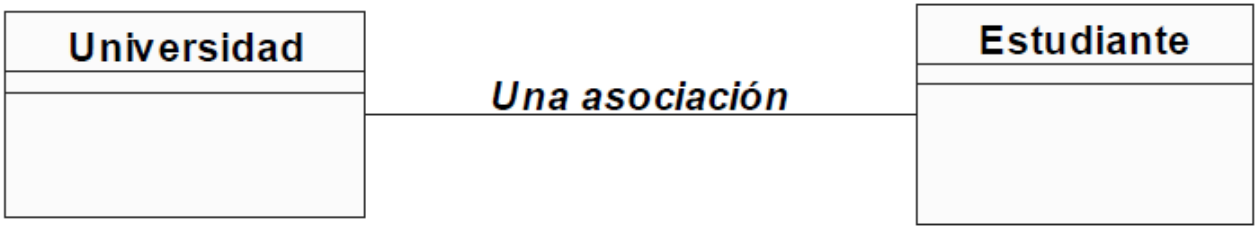


Estas relaciones representan un todo de la parte. El rombo siempre indica el TODO.

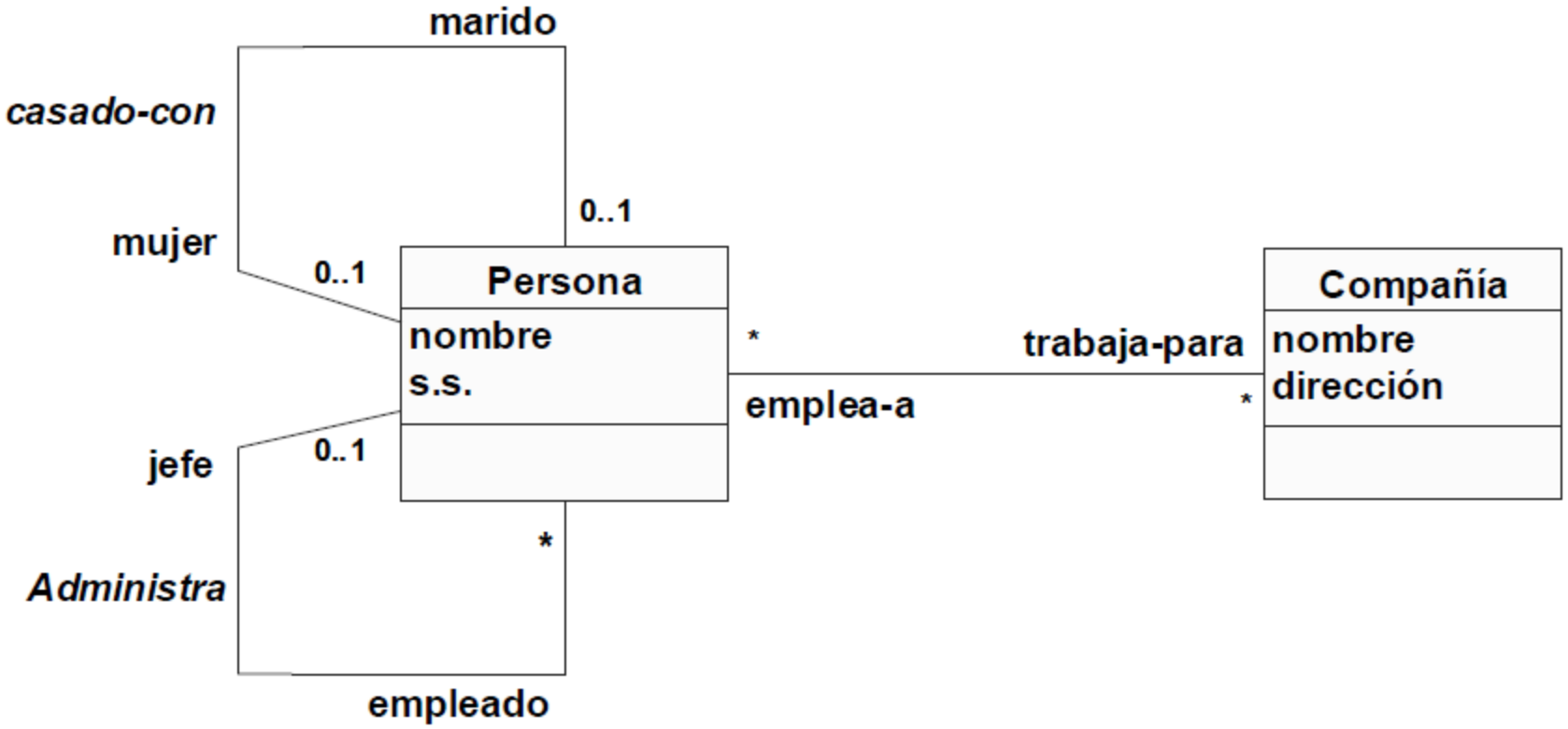
Asociación

- La asociación expresa una conexión bidireccional entre objetos.
- Una asociación es una abstracción de la relación existente en los enlaces entre los objetos.





Ejemplo de Asociación



Asociación

- Especificación de **multiplicidad** (mínima...máxima)

  1 Uno y sólo uno.

  0..1 Cero o uno.

  M..N Desde M hasta N (enteros naturales).

  ***** Cero o muchos.

  0..* Cero o muchos.

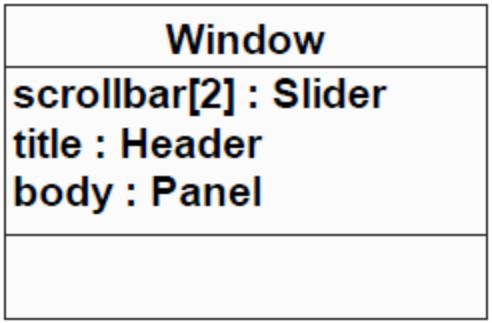
  1..* Uno o muchos (al menos uno).

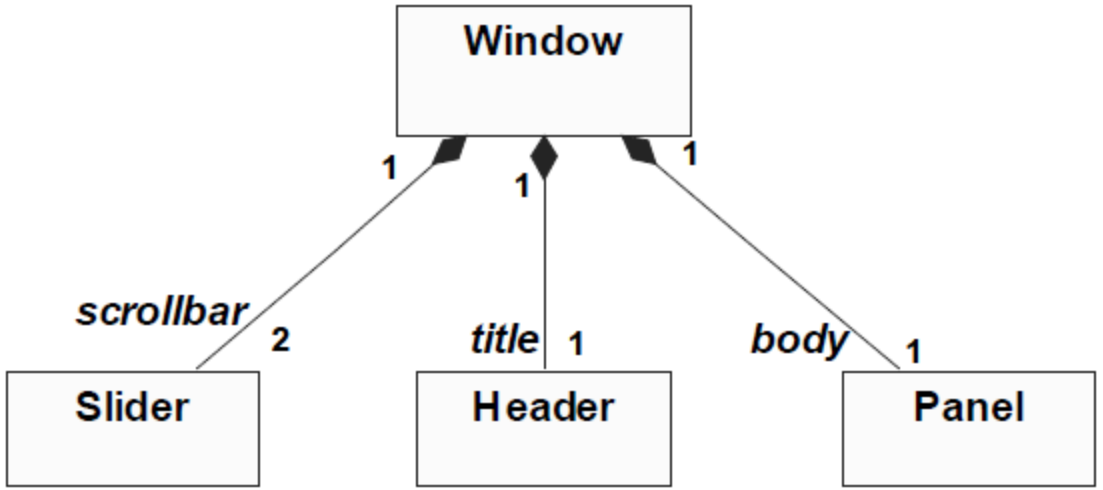
- La multiplicidad mínima >= 1 establece una restricción de existencia

Agregación

- La agregación representa una relación parte_de entre objetos.
- En UML se proporciona una escasa caracterización de la agregación.
- Puede ser caracterizada con precisión determinando las relaciones de comportamiento y estructura que existen entre el objeto agregado y cada uno de sus objetos componentes.

Ejemplo


Window
scrollbar[2] : Slider
title : Header
body : Panel


Window
1
1
1
scrollbar
2
title
1
body
1
Slider
Header
Panel

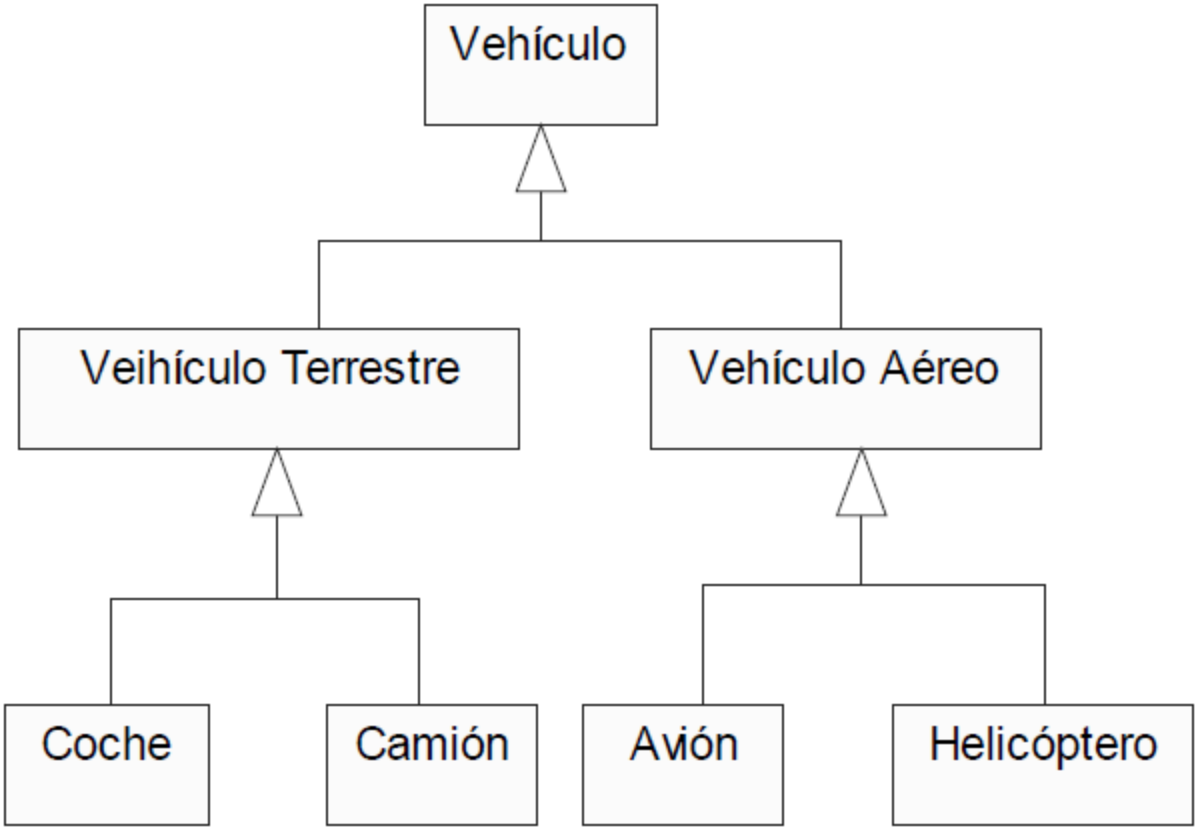
Generalización

- Permite gestionar la complejidad mediante un ordenamiento taxonómico de clases.
- Se obtiene usando los mecanismos de abstracción de Generalización y/o Especialización.
- La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general.

Generalización

- Nombres usados: clase padre - clase hija.
- Otros nombres: superclase - subclase, clase base - clase derivada.
- Las subclases heredan propiedades de sus clases padre, es decir, atributos y operaciones (y asociaciones) de la clase padre están disponibles en sus clases hijas.

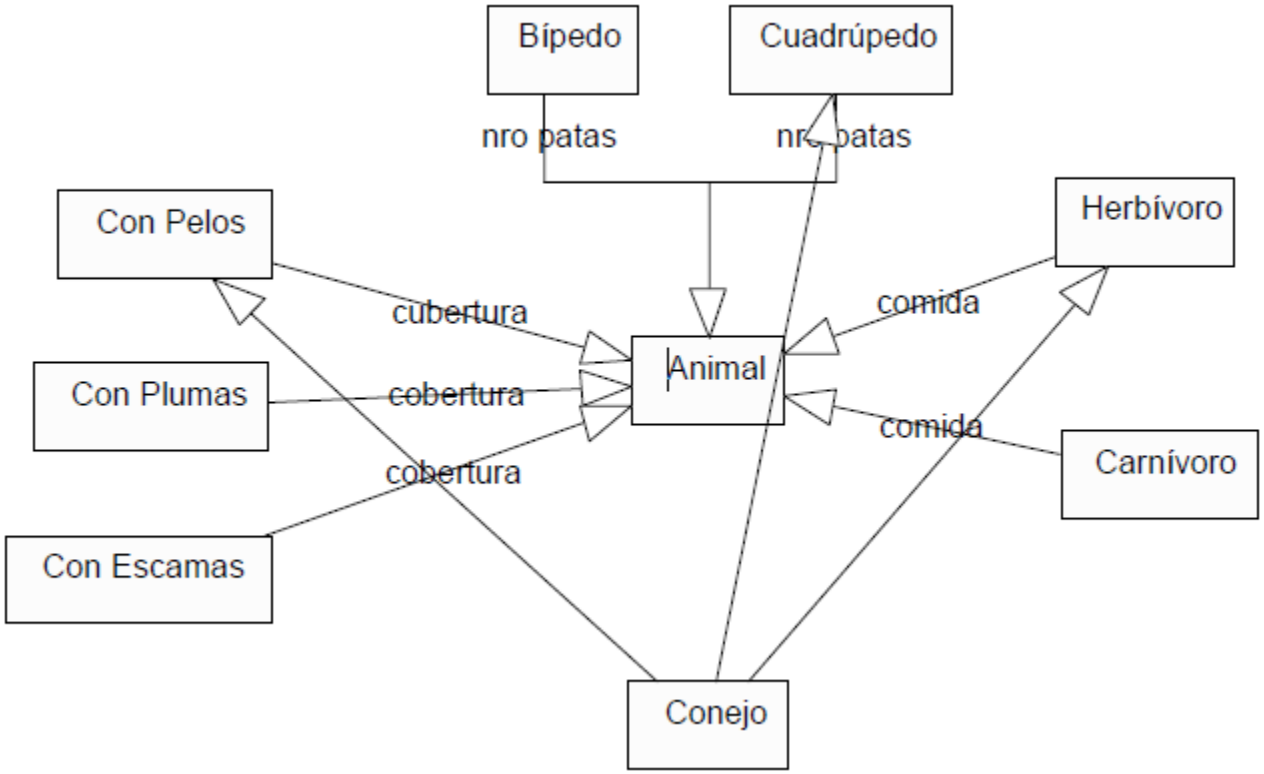
Ejemplo: Generalización



Herencia Múltiple

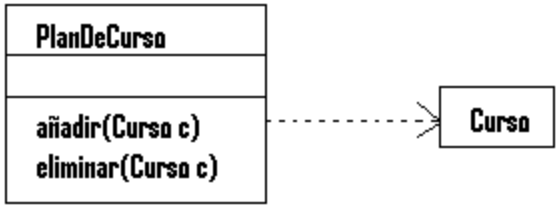
- Se presenta cuando una subclase tiene más de una superclase.
- La herencia múltiple debe manejarse con precaución. Algunos problemas son el conflicto de nombre y el conflicto de precedencia.
- Se recomienda un uso restringido y disciplinado de la herencia. Java y Ada 95 simplemente no ofrecen herencia múltiple.

Ejemplos

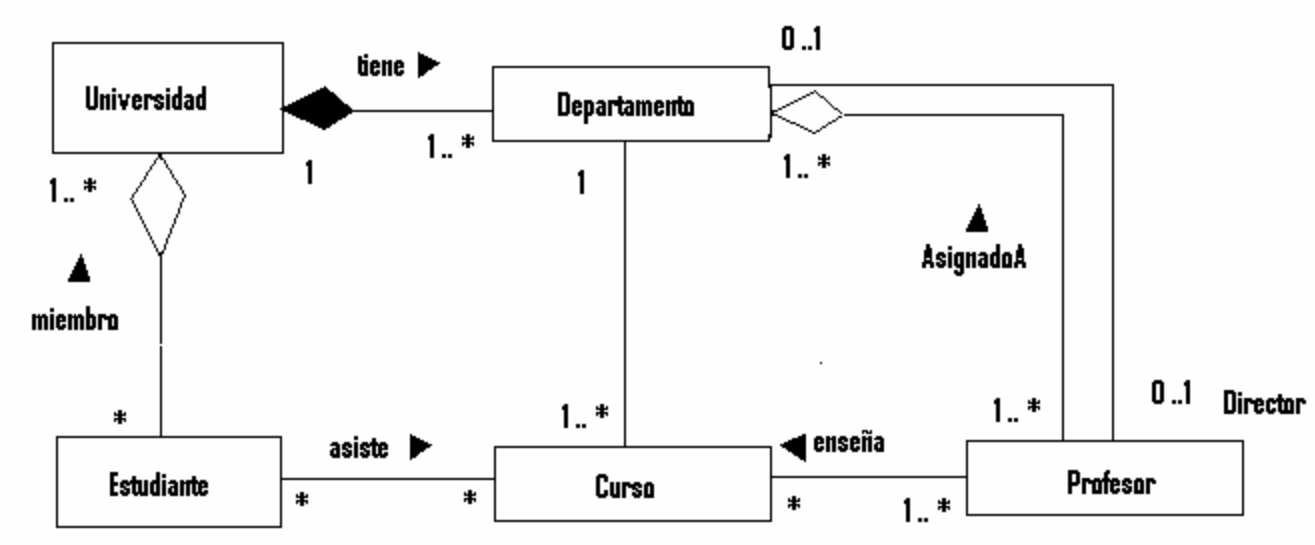

Bípedo
Cuadrúpedo
nro patas
nro patas
Con Pelos
Herbívoro
cubertura
comida
Animal
Con Plumas
cobertura
comida
Carnívoro
cobertura
Con Escamas
Conejo

Dependencia

- Es la conexión que se da entre una clase que utiliza a otra clase como parámetro de una operación.
- Se crea una dependencia que vaya desde la clase con la operación hasta la clase utilizada como parámetro de la operación.



Relaciones estructurales



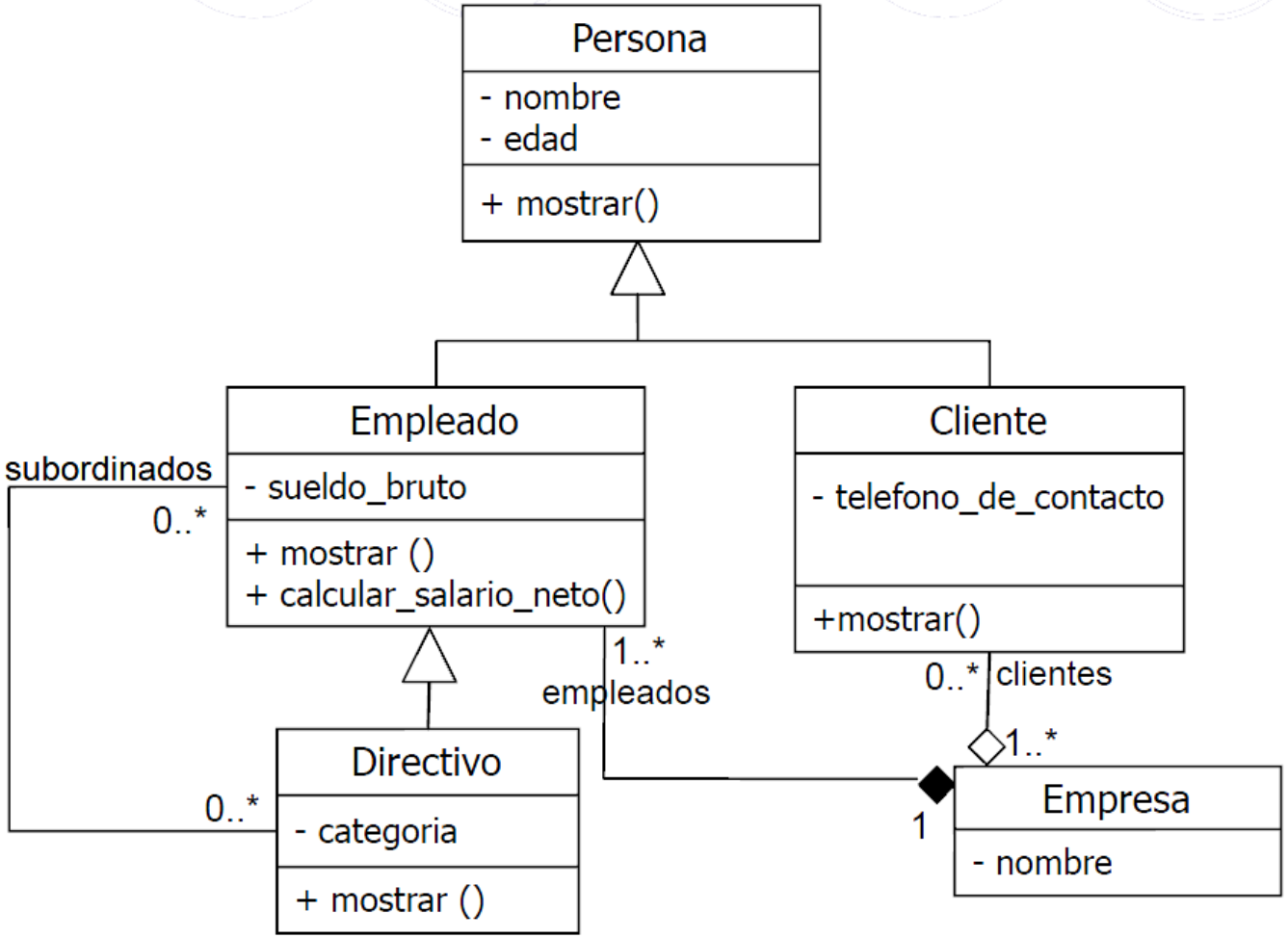
Dentro de las asociaciones tenemos 4 adornos:

- **Nombre**: El nombre de la relación. Por ejemplo: *asiste*, *enseña*. Se puede indicar el sentido con una flecha.
- **Rol**: Rol especifico de la clase que participa en la relación. Por ejemplo *Director* (de la Clase Profesor).
- **Multiplicidad**: Indica los objetos que se pueden conectar.
- **Agregación**: La relación estructural entre iguales. Cuando ambas clases están al mismo nivel.

Ejercicio

- Representar mediante un diagrama de clases la siguiente especificación:
  - Una aplicación necesita almacenar información sobre empresas, sus empleados y sus clientes.
  - Ambos se caracterizan por su nombre y edad.
  - Los empleados tienen un sueldo bruto, los empleados que son directivos tienen una categoría, así como un conjunto de empleados subordinados.
  - De los clientes además se necesita conocer su teléfono de contacto.
  - La aplicación necesita mostrar los datos de empleados y clientes.

Solución


Persona
- nombre
- edad
+ mostrar()
Empleado
- sueldo_bruto
+ mostrar ()
+ calcular_salario_neto()
Cliente
- telefono_de_contacto
+mostrar()
subordinados
0..*
1..*
empleados
0..*
clientes
1..*
Directivo
0..*
- categoria
+ mostrar ()
1
Empresa
- nombre