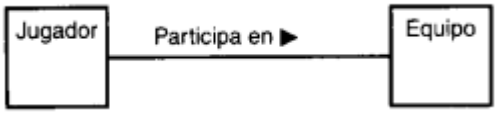
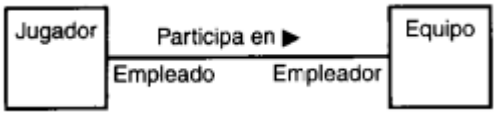
Diagramas de clase

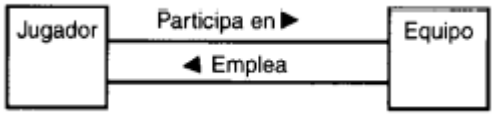
Asociaciones entre clases



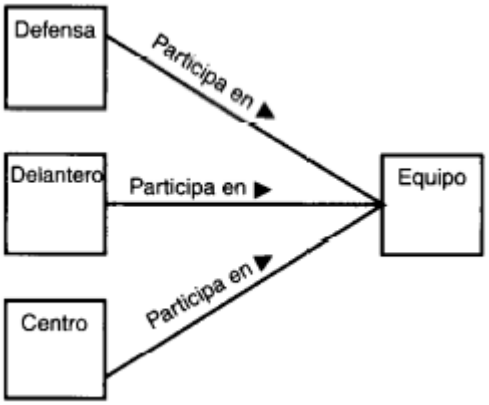
Se puede nombrar el rol o papel de cada clase en la relación



Puede ser bidireccional

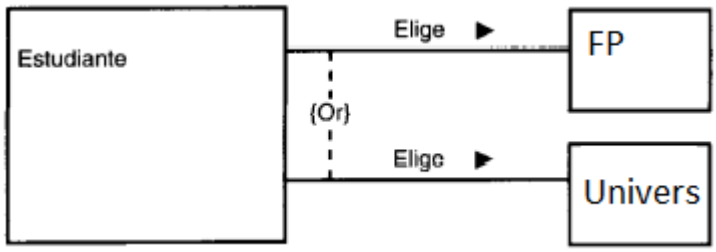


Pueden estar involucradas varias clases (esto NO es herencia)

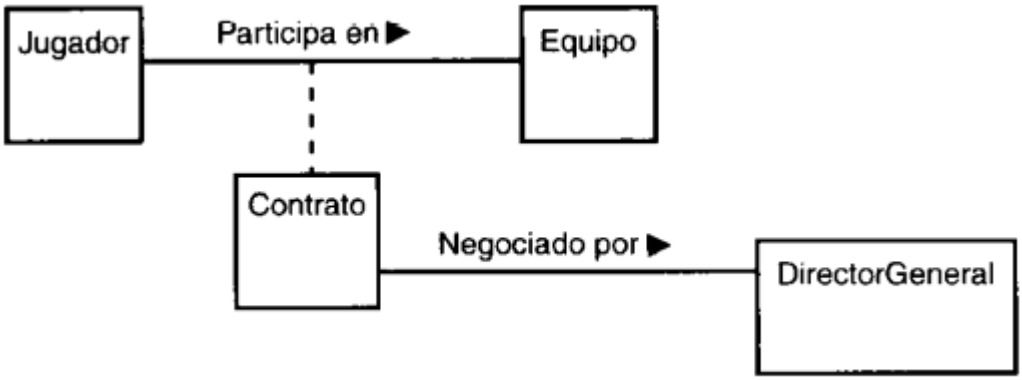


Se pueden poner restricciones indicando por ejemplo que los clientes deben ordenarse antes de ser atendidos

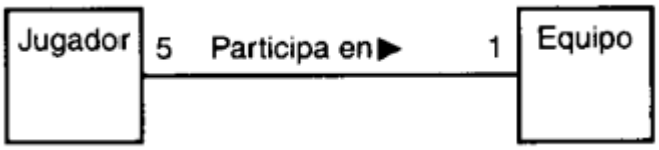
Se puede usar también para restringir relaciones



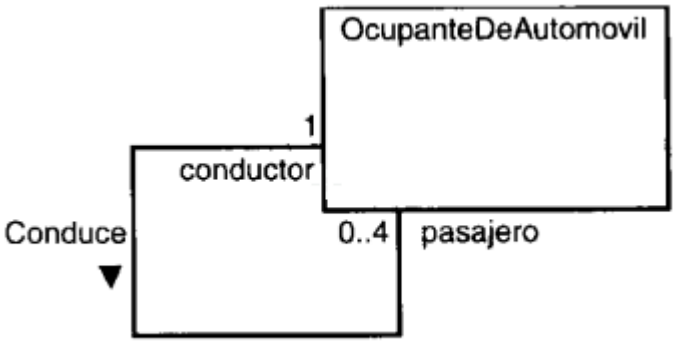
Las relaciones pueden requerir atributos o métodos, en ese caso se incorpora su propia clase, que a su vez puede relacionarse con otras



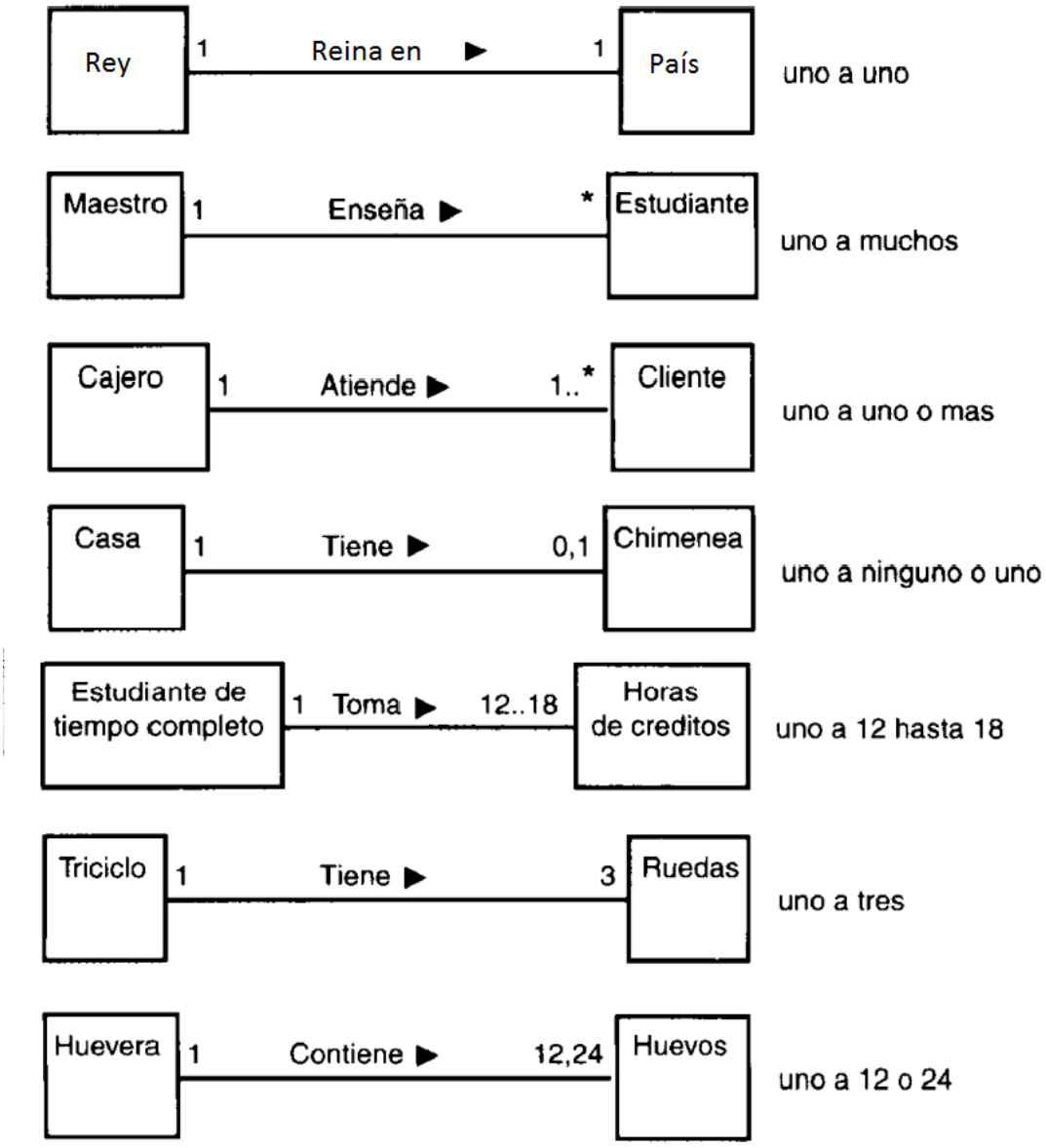
Es frecuente expresar la multiplicidad en las relaciones



A veces pueden surgir relaciones reflexivas

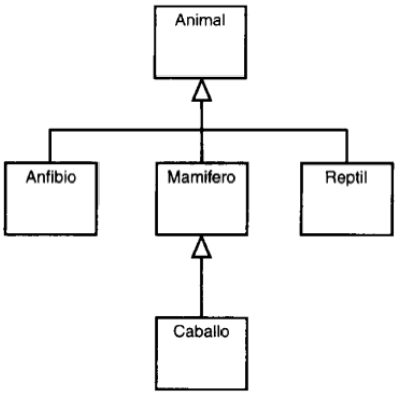


Ejemplos de multiplicidad



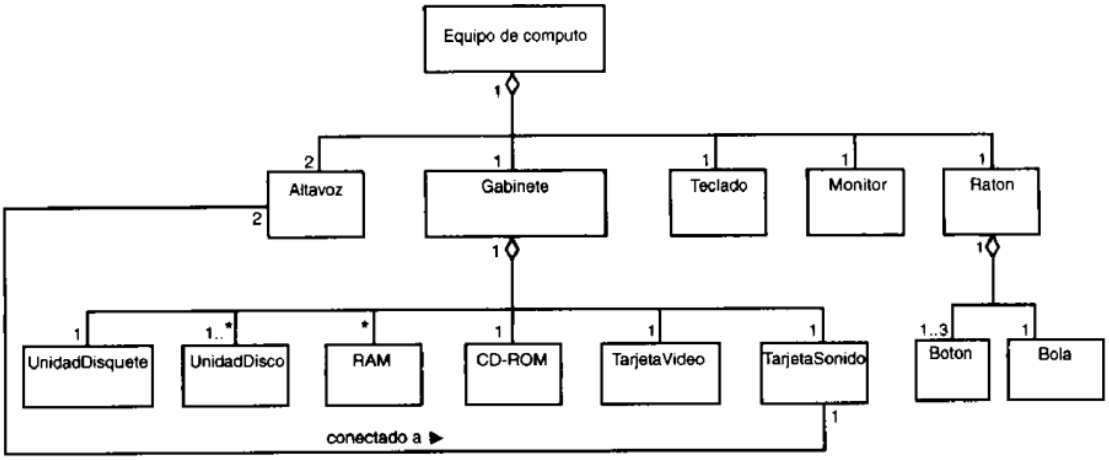
Dependencia: sucede cuando un método de una clase utiliza un objeto de otra, ya sea instanciándolo o recibiéndolo como parámetro

Herencia o generalización

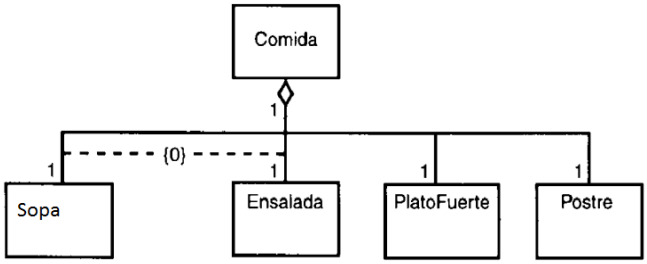


Una clase abstracta (herencia forzada) se distingue poniendo su nombre en cursiva.

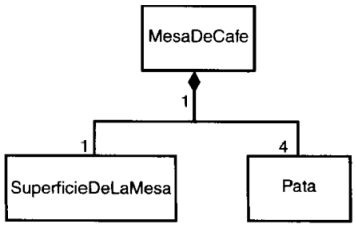
Agregación:



Puede haber restricciones



Composición: agregación más restrictiva. Podemos razonarlo así: si se borra un objeto de una clase, ¿se borraría también el resto?



Resumiendo: ¿Cuándo usar cada tipo de relación?

- **Herencia**: es el caso más reconocible, cuando una o varias clases son variaciones de otra.
- **Composición**: la vida del objeto contenido depende de la vida del objeto contenedor.
- **Agregación / Asociación** : es el caso más difícil de distinguir. En la práctica se implementan de igual forma, pero la agregación tiene un matiz de relación “contiene”, mientras que la asociación no. En la duda, los autores de UML recomiendan no usar agregación.
- **Dependencia**: se trata de la relación más débil, en la que un objeto necesita otra clase de forma puntual. Suele corresponder a parámetros.