POO - Relaciones

Recordemos que dijimos que los Objetos:

"...se los dotará de **características** y de **responsabilidades** para que puedan operar y responder a la necesidad particular que nos interese.
Estos **Objetos** podrán **interactuar** entre ellos."

Características	→	Atributos ✅
Responsabilidades	→	Métodos ✅
¿y que podrán interactuar?	→	**Relaciones**

POO - Relaciones

Habíamos comentado que una de las bases y características principales de la POO es que los objetos “sabrán” hacer cosas. Tendrán un **estado** y un **comportamiento**.

Así, si necesito conocer el **saldo()** de una **Cuenta**, no podré verlo directamente (el atributo saldo estará encapsulado), tendré que preguntarle a la Cuenta, y ella sabrá si me lo puede dar o no. Por ejemplo verificando que yo sea el titular.

POO - Relaciones

Además los objetos podrán comunicarse entre sí estableciendo “relaciones”.

Por ejemplo:

una Persona	**tiene**	un Perro
una CuentaBancaria	**tiene**	un Titular
un ReproductorDeDVD	**usa**	un DVD
un Gato	**es**	un AnimalDomestico

POO - Relaciones

Así, si una CuentaBancaria **tiene** un Titular, entonces a la CuentaBancaria le podré preguntar cuál es el nombre de su Titular:

unaCuentaBancaria->nombreDeTitular()

Y me devolverá por ejemplo “Willie Tanner”

Puede darme el nombre de su titular, porque **tiene** a su Titular. Es decir que uno de sus atributos será :

- Titular titular

POO - Relaciones

Pensando esto en código.
La CuentaBancaria le consultará al Titular (que **tiene)**, para informar su nombre:
Es decir, al hacer: unaCuentaBancaria->**nombreDeTitular();** ...

...en el código del método **nombreDeTitular()**, recurriremos al atributo **Titular titular,** y le consultaremos: **titular->getNombre();** …

Y así obtenemos el nombre del Titular a traves de la CuentaBancaria.

UML - Relaciones

¿Cómo se diagraman estas Relaciones en el **Diagrama de Clases** de **UML**?

Esencialmente, con flechas.

UML - Relaciones

Relación de composición

una Persona **tiene** un Perro

Esta relación se diagrama con una **línea sólida** y punta de flecha **rellena**.

En estas relaciones, aparece la clase relacionada como un atributo.

UML - Relaciones

Además de la flecha: ¿viste el rombo que dibujamos al principio? Hablemos de esto. Vamos a usar dos rombos, uno relleno y otro vacío.

◆ El rombo relleno denotará una relación fuerte. Es decir que uno de los componentes depende del otro, y “moriría” si no existiera el otro.

La mesa tiene patas. Y si se rompiera la relación, las patas dejarían de existir. Porque son patas, siempre y cuando haya una mesa.

La empresa tiene uno o varios empleados.
Un empleado, lo es de una empresa, y si la empresa no existiera, dejaría de ser empleado.

UML - Relaciones

◇ El rombo vacío denotará una relación débil. Es decir que los componente son más fuertes que la relación. Y “sobrevivirían” si la relación se rompiese.

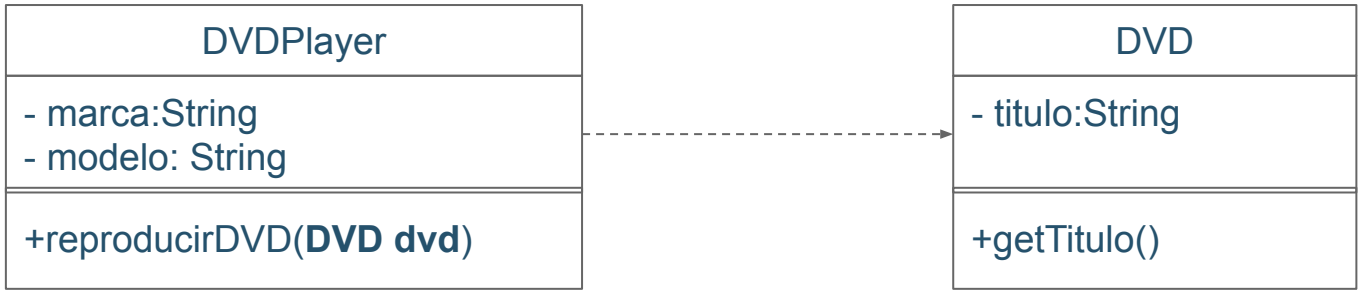
La persona podrá comprar cero, uno o más productos.
Y el producto podrá relacionarse con cero, una, o más personas.
Pero ambos existirán más allá de esta relación.

La agenda podrá tener de 0 a muchas personas anotadas.
Una persona, podrá estar en 0 o muchas agendas anotada.
Y ambos existirán más allá de esta relación.

UML - Relaciones

Relación de uso

un DVDPlayer **usa** un DVD



Esta relación se diagrama con una **línea punteada** y punta de flecha **rellena**.

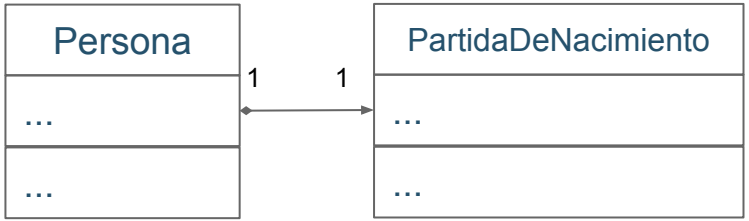
En las relaciones de uso, la clase relacionada aparece como parámetro de un método

UML - Relaciones

Cardinalidad

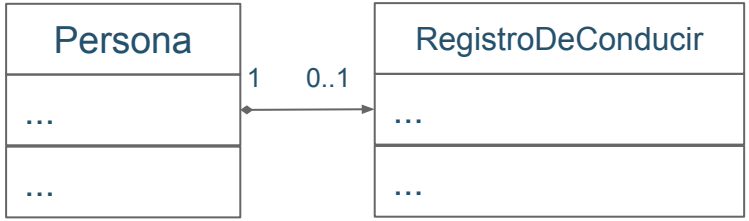
Estas relaciones que vamos mencionando, tienen asociada también una cardinalidad que se puede notar. Así:

Uno a uno



Una **persona** tiene una y solo una partida de nacimiento.
Una **partida** corresponde a una y sólo una personal.

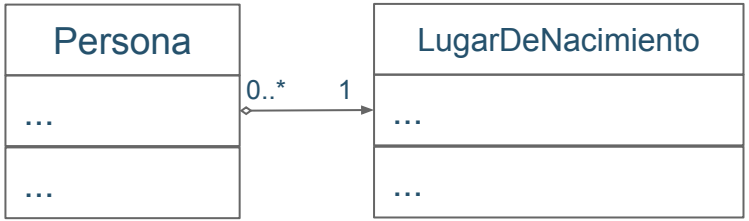
Uno a uno (un lado opcional)



Una **persona** tiene cero o un registro de conducir.
Un **registro** corresponde una y solo una persona.

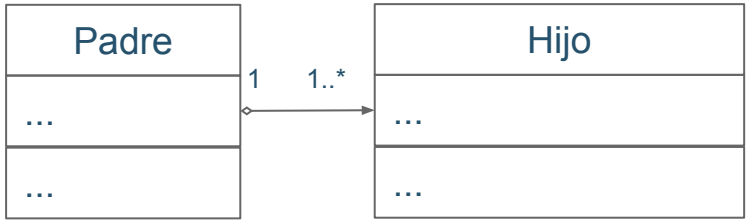
UML - Relaciones

Muchos a uno



Una **persona** tiene uno y solo un lugar de nacimiento.
Un **lugar** puede corresponder al lugar de nacimiento de ninguna, una o muchas personas.

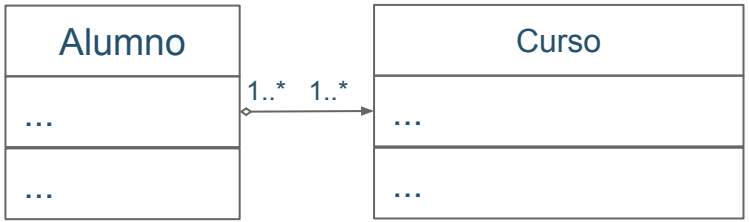
Uno a muchos



Un padre puede tener uno o muchos hijos.
Un hijo, corresponde a un solo padre.

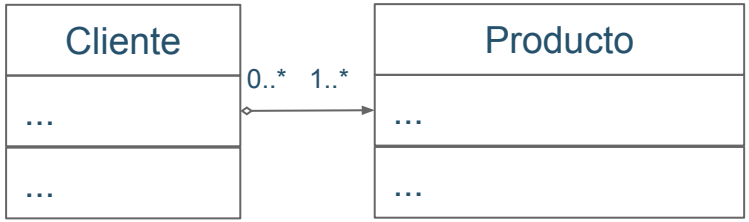
UML - Relaciones

Muchos a muchos



Un **alumnos** puede tener uno o más cursos.
Un **curso** puede corresponder a uno o más alumno

Muchos a muchos (un lado opcional)



Un **cliente** puede tener uno o muchos productos.
Un **producto**, puede ser comprado por 0, 1 o muchos clientes.