RELACIONES ENTRE CLASES

Las clases casi nunca se encuentran aisladas. Por lo general la mayoría de ellas colaboran con otras de varias maneras.

Por tanto, al modelar un sistema también hay que modelar la forma en que las clases se relacionan.

ASOCIACIÓN

Las asociaciones son relaciones que especifican que los objetos de una clase están conectados con los objetos de otro.
Hay cuatro "detalles" que se le pueden poner a estas relaciones: nombre, rol, multiplicidad y agregación.

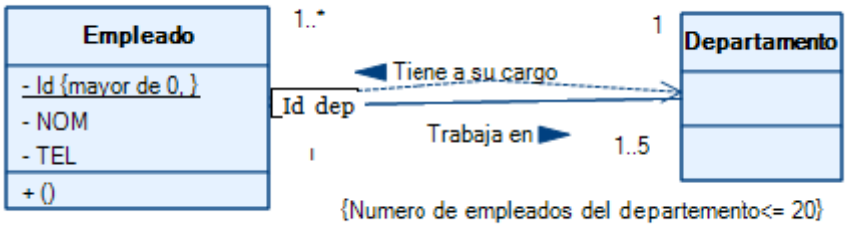
Nombre: Una asociación puede tener un nombre, que se utiliza para describir la naturaleza de la relación. Para evitar ambigüedades, se puede indicar una dirección al nombre, es decir, la dirección en que se debe leer el nombre.

Rol: Un rol es la cara que la clase de un extremo de la asociación presenta a la clase del otro extremo. Es el rol que juega la clase en la asociación.

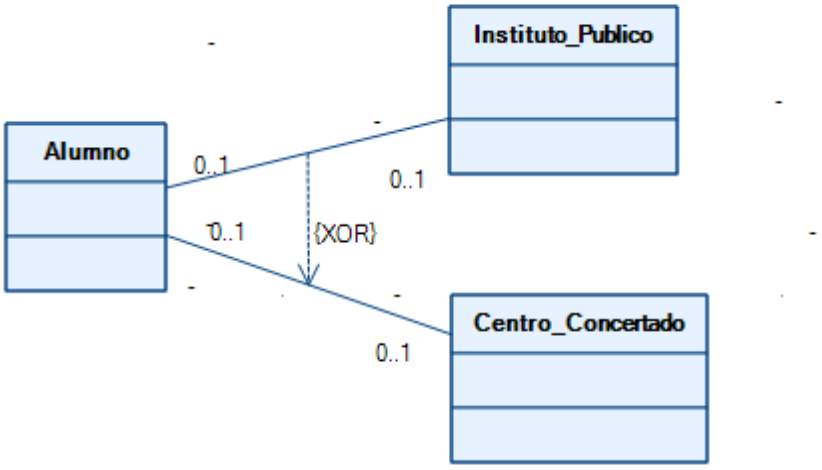
Multiplicidad: Representa el número de objetos que pueden conectarse a través de una relación de asociación. Se puede indicar una multiplicidad de exactamente uno (1..1), cero o uno (0..1), de 0 a muchos (0..*), o uno o muchos (1..*). También se puede indicar un valor exacto (por ejemplo 3), o varios valores (por ejemplo 3,5,7), o intervalos muy concretos (1..12)

Calificador: Es un detalle más, si es una multiplicidad de hasta muchas posibilidades (*). Se puede especificar de qué forma se distinguen unas instancias de otras. Se representa con un rectángulo y dentro el atributo o atributos que distinguen una instancia de la otra en el lado del buscador, no de la clase buscada. Se suele especificar cuando sobre esa asociación se realizará una búsqueda en el sistema. En el ejemplo Empleado tendría un calificador con los Id del departamento. Es similar a un array indexado.

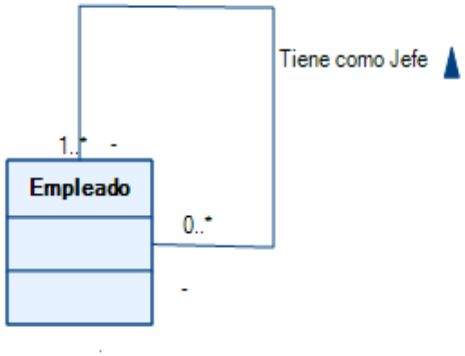
RESTRICCIONES DE UNA ASOCIACIÓN (constraint)
Se establecen entre llaves y junto a la clase que conlleva la restricción.



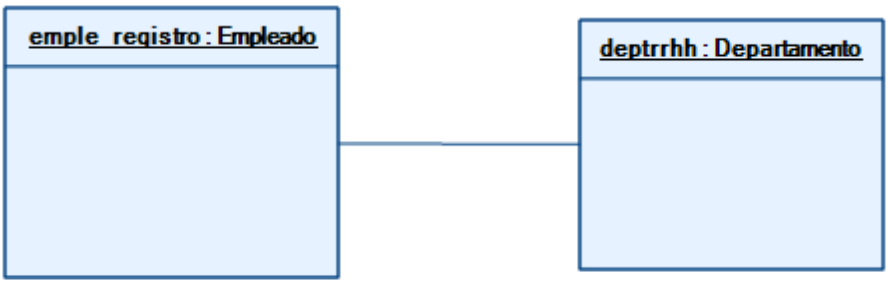
Restricción OR y XOR: Podemos establecer una restricción entre relaciones, de forma que sean excluyentes entre si.



ASOCIACIÓN REFLEXIVA.
Cuando los objetos de una clases estén conectados con objetos de la misma clase se denomina asociación **REFLEXIVA.**



VÍNCULOS
La representación de una asociación entre objetos se denomina vínculo (link)

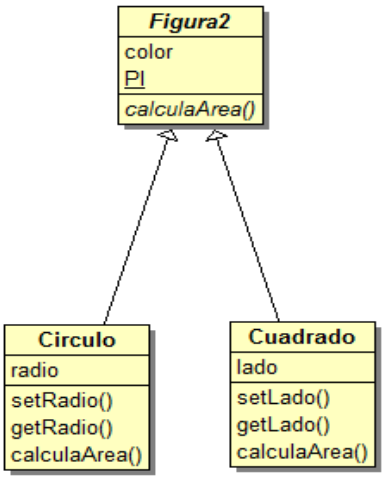


CLASES ASOCIACIÓN

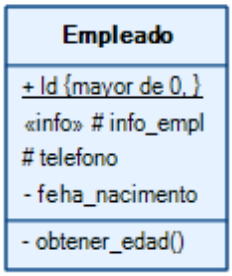
Una asociación puede dar lugar a una clase, en especial cuando la multiplicidad es en los dos casos con límite de muchos (*).
Entonces la relación se representa como una clase nueva con sus propios atributos y operaciones.
Este tipo de clase es muy importante para poder crear los diagramas de entidad relación para el modeladoy diseño de la base de datos que veremos más adelante.

GENERALIZACIÓN/ESPECIALIZACIÓN (HERENCIA)

- La herencia o generalización es, como hemos visto, un concepto fundamental en la metodología orientada a objetos.
- Una clase hereda todos los atributos y operaciones de la clase de la que es heredera, llamada clase madre o superclase. Puede entonces la clase hija o subclase, añadir atributos y métodos nuevos, así como modificar algunos de ellos.
- La modificación de una operación por varias clases hijas conlleva un polimorfismo, es decir, que un mismo método se va a comportar de forma diferente en una clase hija que en otra.
- En UML, una asociación de *generalización* entre dos clases, coloca a estas en una jerarquía que representa el concepto de herencia de una clase derivada de la clase base.
- En UML, las generalizaciones se representan por medio de una línea que conecta las dos clases, con una flecha en el lado de la clase base.
- No es necesario ponerle nombre a la relación.



VISIBILIDAD



- Toda clase encapsula unos elementos (atributos y operaciones) que disponen de ciertos criterios de visibilidad y manipulación para otras clases.
- Los elementos públicos(+) pueden ser usados por cualquier otra clase.: public
- Los elementos privados(-) pueden ser usados sólo por la clase propietaria.: private
- Los elementos protegidos(#) pueden ser usados por la subclases (y por la clases del mismo paquete en el caso de Java): protected

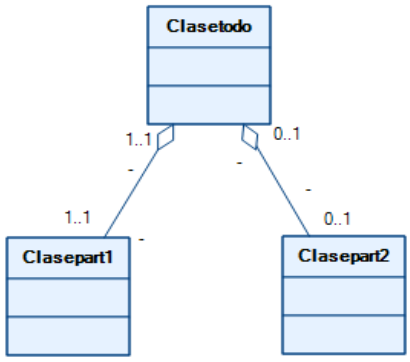
- La notación UML especifica que todo atributo y operación de una clase ha de disponer de un indicador de visibilidad.

CLASES ABSTRACTAS

- Una clase abstracta es aquella superclase de la que no vamos a obtener instancias /objetos.
- Sus operaciones serán definidas en las clases hijas, en ella están vacías, sin implementación.
- Esto es debido a que no tiene sentido trabajar directamente con objetos de esta clase, sino siempre con derivados.
- En UML su nombre se pone en cursiva para distinguirla de las clases no abstractas.

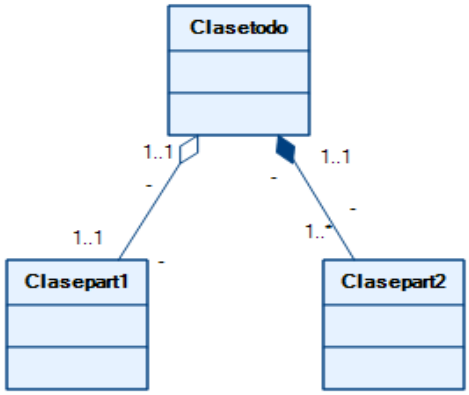
AGREGACIÓN

- Una clase guarda una relación con otras clases respecto a que la primera está formada por las segunda, es decir, está particionada en la clases agregadas.
- Es una relación todo-parte. La clase acumuladora es el todo y las clases componentes son las partes que la forman.
- No es necesario ponerle nombre a la relación (todo--- tiene una---- parte), y se representa con un rombo blanco en el lado del todo.
- Es preciso especificar la multiplicidad, es decir cuantos todos y partes colaboran en la relación.



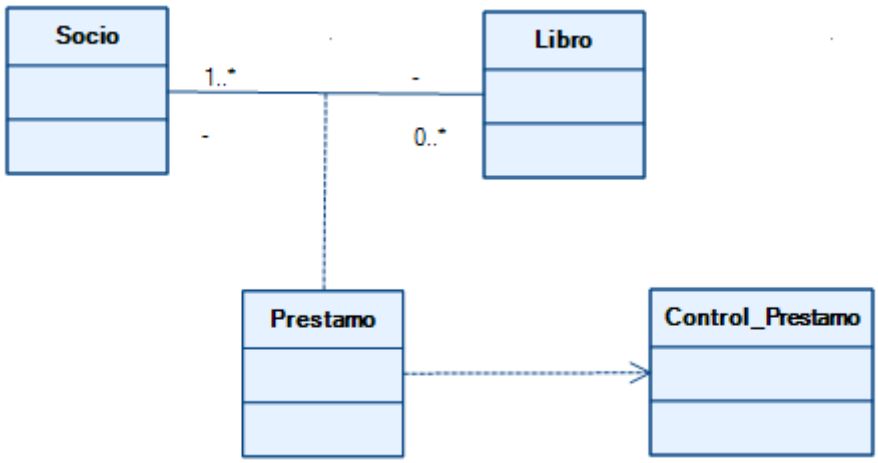
COMPOSICIÓN:

- Es una agregación con una relación muy fuerte. Esto quiere decir que la "parte" implicada sólo puede existir si pertenece a ese "todo".
- Si la clase "todo" desaparece, esa clase "parte" también desaparece del sistema.
- La multiplicidad de la relación parte pertenece a (0-1 o de 1-1) todo.
- La representación es una agregación pero con el rombo de color negro.



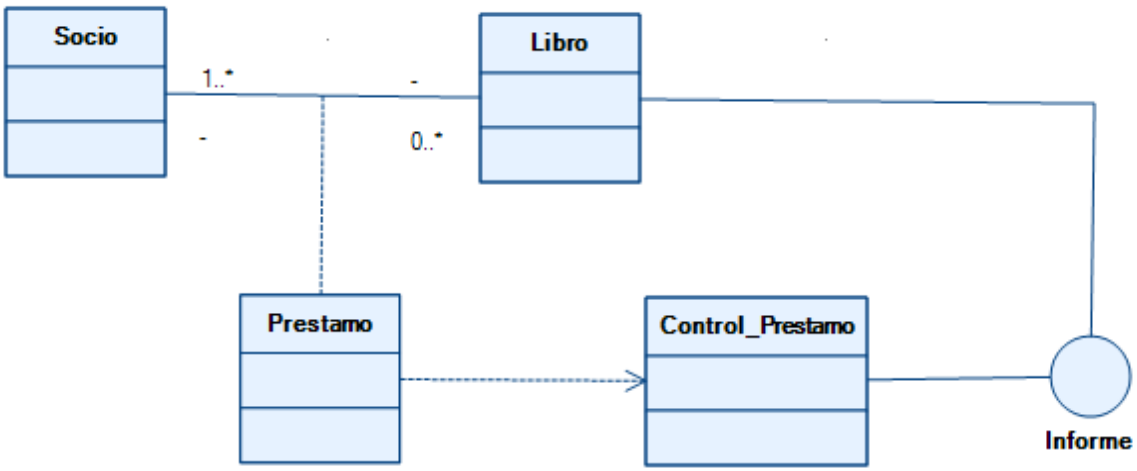
DEPENDECIA

- Es el tercer tipo de relación. La clase destinataria depende de una operación de la clase origen. Por tanto existe un mensaje entre clases.
- La clase destinataria sólo existe porque la clase origen tiene una instancia y existe una operación que se ejecuta (mensaje). Se denomina relación semántica o de uso, ya que afecta a la propia creación o uso de la clase dependiente.
- Se representa con una flecha discontinua.



REALIZACIÓN

- Es una relación que es mezcla de una dependencia y de una generalización.
- La clase dependiente precisa que la origen se instancie y en algunos casos ejecute una operación concreta(imprimir_informe()).
- La clase dependiente se denomina INTERFACE y representa la forma en que puede actuar la clase.
- Además esta clase interface está formada sólo por operaciones que han sido recogidas de otras clases para su re utilización, no tiene atributos.



- Se representa con una flecha discontinua con la punta blanca. La clase interface se representa con la palabra <<<interfaz>> antes del nombre.
- Otra forma más común, es una linea continua que une la clase un circulo que representa el interface.