UD 5: EL LENGUAJE UML(I): DIAGRAMAS DE CLASES

En esta unidad:

1 El Lenguaje UML

Ver presentación adjunta en el aula virtual

1.1 Conceptos básicos de UML

1.2 Tipos de diagramas y usos

2 La orientación a objetos y el modelado de los mismos

2.1 El objeto y las clases

Un objeto es una entidad identificable del mundo real. Puede tener una existencia física (un caballo, un libro) o no tenerla (un texto de ley). *Identificable* significa que el objeto se puede designar, por ejemplo:

Mi yegua Rosalinda

Mi libro sobre UML

El artículo 203B del Código Civil

En UML todo objeto posee un conjunto de atributos (estructura) y un conjunto de métodos (comportamiento). Un atributo es una variable destinada a recibir un valor. Un método es un conjunto de instrucciones que toman unos valores de entrada y modifican los valores de los atributos o producen un resultado.

Incluso los objetos del mundo real son percibidos siempre como dinámicos. Así, en UML, un libro se percibe como un objeto capaz de abrirse él mismo en una página determinada.

Todo sistema concebido en UML está compuesto por objetos que interactúan entre sí realizan operaciones propias de su comportamiento.

Un grupo de alumnos es un sistema de objetos que interactúan entre sí, cada objeto posee su propio comportamiento.

De esta forma, el comportamiento global de un sistema se reparte entre los diferentes objetos.

2.1.1 La abstracción

La abstracción es un principio muy importante en el modelado. Consiste en tener en cuenta únicamente las propiedades pertinentes de un objeto para un problema concreto. Los objetos utilizados en UML son abstracciones del mundo real.

Si nos interesamos por las personas en su actividad de profesional, atenderíamos a las propiedades profesión, salario, antigüedad, puesto o aptitud.

Si nos interesamos por las personas en sus relaciones sociales, atenderíamos a las propiedades edad, padres, hijos, amigos o pertenencia a asociaciones.

La abstracción es una simplificación indispensable para el proceso de modelado. Un objeto UML es una abstracción de un objeto del mundo real de acuerdo con las necesidades del sistema de la cual sólo se tienen en cuenta los elementos esenciales.

2.1.2 Clases de objetos

Un conjunto de objetos similares, es decir, con la misma estructura y comportamiento, y constituidos por los mismo atributos y métodos, forma una clase de objetos. La estructura y el comportamiento pueden entonces definirse en común en el ámbito de la clase.

Todos los objetos de una clase, llamada también instancia de clase, se distinguen por tener una identidad propia y sus atributos les confieren valores específicos.

El conjunto de personas constituye la clase Persona, que posee la estructura y el comportamiento descritos en la Ilustración 1.

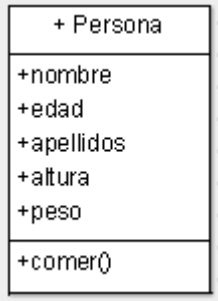

Ilustración 1 La clase Persona

*Recordemos que el **nombre de las clases** se escribe en **singular** y está siempre formado por un **nombre común** precedido o seguido de uno o varios adjetivos que lo califican. Dicho*

nombre es representativo del conjunto de objetos que forman la clase y representa la naturaleza de las instancias de una clase.

En UML las clases son representadas mediante una caja dividida en tres partes: nombre de la clase, atributos y operaciones o métodos.

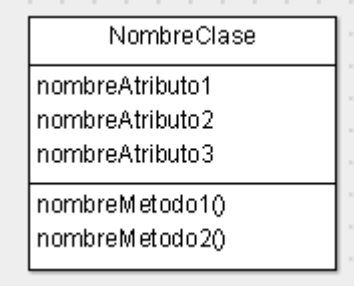

Ilustración 2 Representación simplificada de una clase en UML

Los atributos contienen información de los objetos. El conjunto de atributos forma la estructura del objeto.

Los métodos se corresponden con los servicios ofrecidos por el objeto y pueden modificar el valor de los atributos. El conjunto de métodos forma el comportamiento del objeto.

El número de atributos y métodos varía de acuerdo con la clase. No obstante, se desaconseja emplear un número elevado de ellos ya que, en general, éste refleja una mala concepción de la clase.

Esta es la forma más simple de representación de las clases, porque no hace aparecer las características de los atributos y de los métodos, a excepción de su nombre. Se utiliza a menudo en las primeras fases del modelado.

2.1.3 Encapsulación

La encapsulación consiste en ocultar los atributos y métodos del objeto a otros objetos. En efecto, algunos atributos y métodos tienen como único objetivo tratamientos internos del objeto y no deben estar expuestos a los objetos exteriores. Una vez encapsulados, pasan a denominarse atributos y métodos privados del objeto.

La encapsulación es una abstracción, ya que se simplifica la representación del objeto con relación a los objetos externos. Esta representación simplificada está formada por atributos y métodos públicos del objeto.

Al andar, las personas efectúan diferentes movimientos como pueden ser levantar las piernas, balancear los brazos o mover la cabeza. Esos movimientos son internos al funcionamiento de la persona y no tiene por qué ser conocidos en el exterior. Son métodos privados. Las operaciones acceden a una parte interna de la persona: sus músculos, su cerebro y su vista. La parte interna se representa en forma de atributos privados.

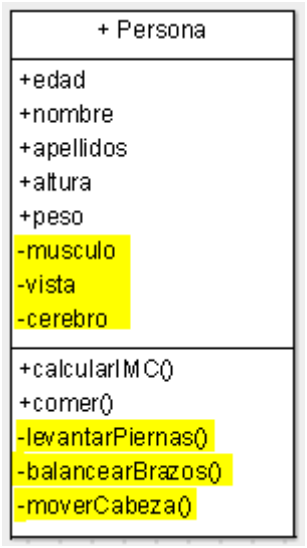

Ilustración 3 La clase Persona detallada. En amarillo se muestran los elementos privados

UML, al igual que la mayoría de lenguajes modernos orientados a objetos, introduce tres posibilidades de encapsulación:

- El atributo o el método privado: la propiedad no se expone fuera de la clase, ni tampoco dentro de sus subclases.
- El atributo o el método protegido: la propiedad sólo se expone en las instancias de la clase y sus subclases
- La encapsulación de empaquetado: la propiedad sólo se exponen en las instancias de clases del mismo empaquetado.

La noción de propiedad privada conduce a establecer una diferencia entre las instancias de una clase y las de sus subclases. Esta diferencia está vinculada a aspectos bastante sutiles de la programación con objetos. La encapsulación de empaquetado, por su parte, procede del lenguaje Java.

Cada uno de los tipos de encapsulado tiene un signo que se coloca delante del nombre de la propiedad en el diagrama de clases:

Encapsulado	Signo	Descripción
Público	+	Visible para todos
Protegido	#	Visible en las subclases de la clase
Privado	-	Visible sólo en la clase
Empaquetado	~	Visible sólo en las clases del mismo empaquetado

2.1.4 La noción de tipo

De manera general, llamamos variable a cualquier elemento que pueda tomar un valor, es decir, a cualquier atributo, parámetro o incluso a los valores de retorno de un método

El tipo es una especificación aplicada a una variable. Consiste en fijar el conjunto de valores posibles que la variable puede tomar. Dicho conjunto puede ser:

- una clase, en cuyo caso la variable debe contener una referencia a una instancia de la misma
- un tipo estándar, como el conjunto de enteros, cadenas de caracteres, boleanos o reales.

En UML, los tipos estándar son:

- *Integer*, para el tipo de los enteros;
- *String*, para el tipo de las cadenas de caracteres;
- *Boolean*, para el tipo de los boleanos;
- *Real*, para el tipo de los reales.

***Ejemplo**: 1 o 3 o 10 son ejemplos de valores de entero. "Coche" es un ejemplo de cadena de caracteres en la cual se ha optado por las comillas como separadores. False y True son los dos únicos valores posibles del tipo Boolean. 3.1415, donde el punto hace la función de separador de decimales, es un ejemplo bien conocido de número real.*

Veremos que, en general, el tipo de un atributo sólo recurre a una clase si ésta es una clase de una biblioteca externa al sistema modelado o una interfaz como las que veremos más adelante. Aun así, no es aconsejable utilizar clases del sistema para dar un tipo a un atributo. En esos casos, como veremos a continuación, vale más recurrir a las asociaciones entre objetos.

Por el contrario, el tipo de un parámetro o del retorno de un método puede ser un tipo estándar o una clase, pertenezca o no al sistema.

El tipo de un atributo, de un parámetro y del valor de retorno de un método se especifica en la representación de la clase.

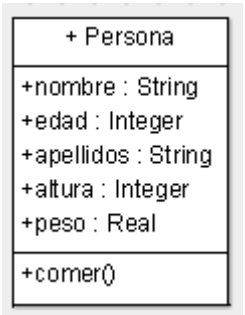

Ilustración 4 La clase Persona con el tipo de los atributos

2.1.5 Firma de los métodos

Un método de una clase puede tomar parámetros y devolver un resultado. Los parámetros son valores transmitidos:

- En la ida, al enviar un mensaje que llama a un método
- O en el retorno de llamada del método.

El resultado es un valor transmitido al objeto que efectúa la llamada cuando ésta se devuelve.

Como hemos visto anteriormente, tanto los parámetros como el resultado pueden tener tipos. El conjunto constituido por el nombre del método, los parámetros con su nombre y su tipo, así como el tipo de resultado, se conoce como firma del método. Una firma adopta la siguiente forma

<nombreMétodo> (<dirección><nombreParámetro>:<tipo>, ...) : <tipoResultado>

Es posible indicar la dirección en la cual el parámetro se transmite colocando delante del nombre del parámetro una palabra cable. Las tres palabras claves posibles son:

- *in:* el valor del parámetro sólo se transmite al efectuar la llamada
- *out:* el valor del parámetro sólo se transmite en el retorno a la llamada del método
- *inout:* el valor del parámetro se transmite en la llamada y en el retorno.

Si no se especifica ninguna palabra clave, el valor del parámetro sólo se transmite en la llamada.

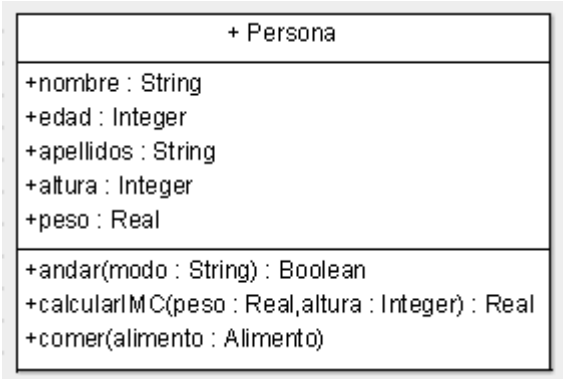

Ilustración 5 La clase Persona con la firma de los métodos

2.1.6 La forma completa de representación de las clases

La representación completa de las clases muestra los atributos con las características de encapsulación, el tipo y los métodos con la firma completa.

También es posible dar valores predeterminados a los atributos y a los parámetros de un método. El valor predeterminado de un atributo es el que se le atribuye al crear un nuevo objeto. El valor predeterminado de un parámetro se utiliza cuando aquél que llama a un método no proporciona explícitamente el valor del parámetro en el momento de la llamada.

La representación completa de una clase puede llegar a ser demasiado compleja, por lo que es posible escoger una representación intermedia entre la representación simplificada y la completa.

2.1.7 Los atributos y los métodos de clase

Las instancias de una clase contienen un valor específico para cada uno de sus atributos. Este valor, por tanto, no se comparte con el conjunto de instancias. En algunos casos, es preciso utilizar atributos cuyo valor es común a todos los objetos de una clase. Estos atributos son conocidos como *atributos de clase*, porque están vinculados a la clase y no a los objetos de la misma.

*En muchas herramientas UML, no se utilizan los términos "atributo y método de clase". Estas herramientas dan preferencia a la denominación "**atributos o métodos estáticos**", denominación propia de lenguajes como Java o C++.*

*En ArgoUML estos atributos y métodos son conocidos como "**ownerScope**"*

Los atributos de clase se representan mediante un nombre subrayado. Pueden estar encapsulados y poseer un tipo. Además, se recomienda vivamente asignarles un valor predeterminado.

***Ejemplo**: Estudiamos el sistema de una carnicería, por lo que introducimos una clase PorcionCarne. La venta de este producto está sujeta a una tasa de IVA cuyo montante es el mismo para todas las porciones. El atributo se destaca y se protege, ya que sirve para calcular el precio con el IVA incluido y se expresa en porcentajes, de ahí que su tipo sea Integer. El valor predeterminado es 10, es decir 10%, que es la tasa de de IVA que se aplica en España a los productos alimenticios.*

Dentro de una clase, también puede existir uno o varios métodos de clase vinculados a la misma. Para llamar a un método de clase, hay que enviar un mensaje a la propia clase y no a una de sus instancias. Estos métodos sólo manipulan los atributos de clase.

La clase PorcionCarne incorpora un método de clase que sirve para fijar la tasa de IVA. En efecto, la ley es la que fija dicha tasa y, como ya experimentamos hace poco, la ley puede modificarse.

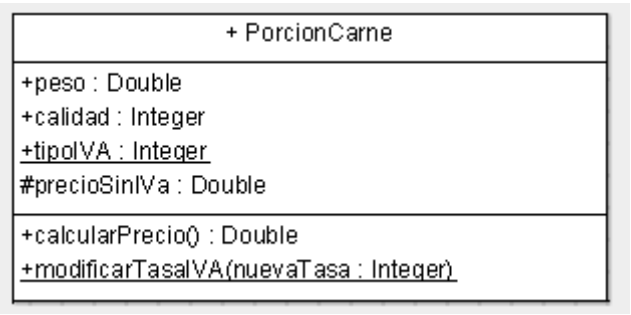

Ilustración 6 Atributos y métodos de clase

Los atributos y métodos estáticos NO SE HEREDAN. La herencia se aplica a la descripción de las instancias, calculada a través de la unión de la estructura y del comportamiento de la clase y de sus superclases. Una subclase puede acceder a un atributo o a un método estático de una de sus superclases, pero no los hereda. De haber herencia, tendríamos tantos ejemplares de atributos o métodos como subclases poseyera la clase que los introdujo.

Recordamos también que una subclase puede acceder a un atributo o a un método de clase de una de sus superclases, a condición de que no se haya declarado como privado.

2.2 Especialización y generalización

Hasta el momento, cada clase de objetos se introduce de forma separada a las demás clases. Las clases pueden definirse también como subconjuntos de otras clases, con las que comparten propiedades.

Hablamos entonces de subclases de otras clases que, por tanto, constituyen especializaciones de esas otras clases.

La clase de los coches es una subclase de la clase de los vehículos.

La generalización es la relación inversa a la especialización. Si una clase es una especialización de otra clase, ésta última es una generalización de la primera. Es su superclase.

La clase de los vehículos es una superclase de la clase de los coches.

La relación de especialización puede aplicarse a varios niveles, dando lugar a la jerarquía de clases.

La clase de los coches es una subclase de la clase de los vehículos rodados, y esta es una subclase de la clase de los vehículos. La clase de las bicicletas es otra subclase de la clase de los vehículos rodados, mientras que la clase de los aviones es una subclase de los vehículos. Ilustración 7

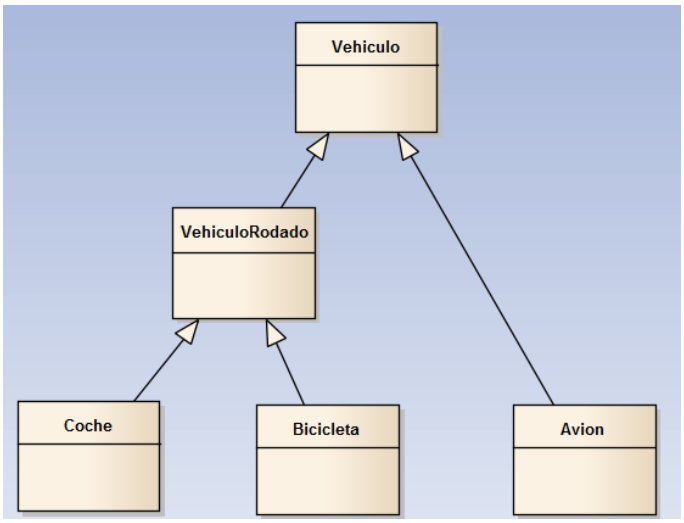

Ilustración 7 Jerarquía de clases

La forma de representar este tipo de relaciones en UML es mediante una flecha, que va siempre desde la subclase hacia la superclase, cuya punta es triangular.

2.3 Herencia

La herencia es la propiedad que hace que una subclase se beneficie de la estructura y del comportamiento de su superclase. La herencia deriva del hecho de que las subclases son subconjuntos de las superclases. Sus instancias son asimismo instancias de la superclase y, por

consiguiente, además de la estructura y del comportamiento introducidos en la subclase, se benefician también de la estructura y comportamiento definidos por la subclase.

Tomamos un sistema en el que la clase Coche es una subclase directa de la clase Vehículo. El Coche se describe entonces mediante la combinación de la estructura y del comportamiento derivados de las clases Coche y Vehículo.

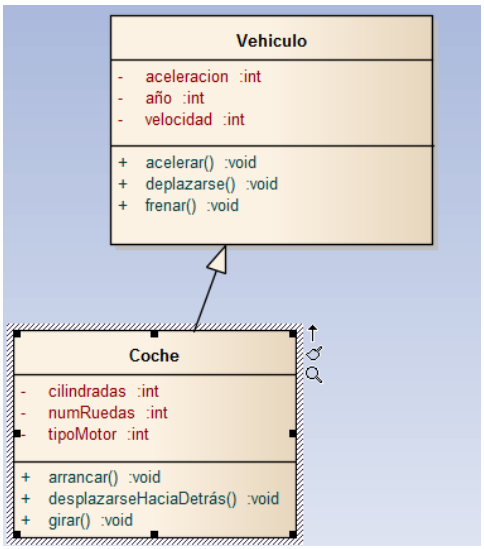

Ilustración 8 Herencia de atributos y operaciones

La herencia es una consecuencia de la especialización. Sin embargo, los informáticos emplean mucho más a menudo el término *hereda* que *especializa* para designar la relación entre una subclase y su superclase.

2.4 Clases abstractas y concretas

Si examinamos la jerarquía de presentada en la Ilustración 7 vemos que en ella existen dos tipos de clases:

- Las clases que poseen instancias, es decir, las clases Coche y Bicicleta, llamadas clases concretas.
- Las clases que no poseen directamente instancias, como la clase Vehículo. En efecto, si bien en el mundo real existen coches, aviones, etc., el concepto de vehículo propiamente dicho continúa siendo abstracto. No basta para definir completamente un vehículo. La clase Vehículo se llama clase abstracta.

La finalidad de las clases abstractas es poseer subclases concretas, éstas sirven para factorizar atributos y métodos comunes a las subclases. De esta manera, la clase abstracta fija los atributos y operaciones mínimas que debe tener cualquier clase que la concrete.

En UML cualquier elemento (clase, atributo u operación) que sea abstracto se representa con letra cursiva.

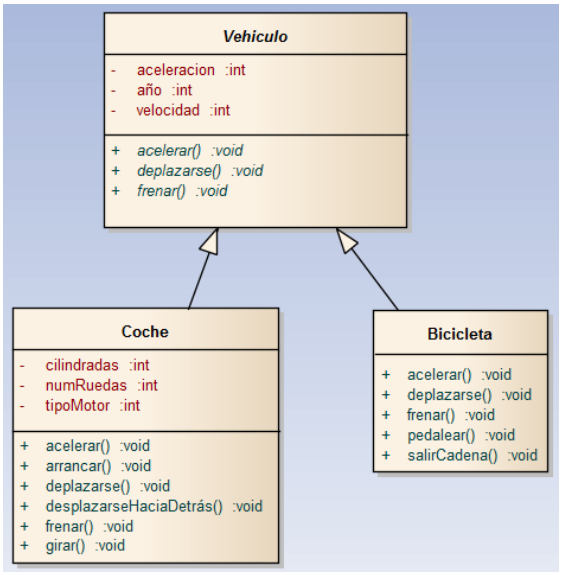

Ilustración 9 En UML, los nombres de las clases abstractas se escriben en cursiva

2.5 Polimorfismo

El polimorfismo significa que una clase (generalmente abstracta) representa un conjunto formado por objetos diferentes, ya que éstos son instancias de subclases diferentes. Cuando se llama a un método del mismo nombre, esta diferencia se traduce en comportamientos distintos (excepto en los casos en los que el método es común y las subclases lo han heredado de la superclase).

Tomemos la jerarquía de clases ilustrada en la Ilustración 9. El método acelerar() tiene un comportamiento diferente según si el vehículo es una instancia de Coche o de Bicicleta.

Si consideramos la clase Vehículo en su totalidad, tenemos un conjunto de vehículos que reaccionan de distinta manera al activarse el método acelerar().

3 Las asociaciones entre objetos

3.1 Los vínculos entre objetos

En el mundo real, muchos objetos están vinculados entre sí. Dichos vínculos corresponden a una asociación existente entre los objetos

Ejemplos:

El vínculo existente entre una persona y su padre;

El vínculo existente entre una persona y su madre;

El vínculo existente entre un trabajador y la empresa a la que pertenece;

El vínculo existente entre una empresa y su propietario

En UML, estos vínculos se describen mediante asociaciones, de igual modo que los objetos se describen mediante clases. Un vínculo es un elemento de una asociación. Por tanto, una asociación vincula a las clases y a sus instancias.

Las asociaciones tienen un nombre y, como ocurre con las clases, éste es un reflejo de los elementos de la asociación.

Ejemplos:

La asociación 'padre' entre la clase Descendiente y la clase Padre;

La asociación 'madre' entre la clase Descendiente y la clase Madre;

La asociación 'trabaja entre la clase Trabajador y la clase Empresa;

La asociación 'propietario' entre la clase Empresa y la clase Persona.

Las asociaciones que hemos estudiado hasta el momento en forma de ejemplos establecen un vínculo entre dos clases, por lo que reciben el nombre de ***asociaciones binarias.***

Las asociaciones que vinculan tres clases se denominan asociaciones ***ternarias*** *y, en general, aquellas que vinculan n clases reciben el nombre de asociaciones* ***n-arias****.*

En la práctica, la gran mayoría de asociaciones son binarias y las cuaternarias y superiores apenas se utilizan.

3.2 Representación de las asociaciones entre clases

La representación gráfica de una asociación binaria consiste en una línea continua que une las dos clases cuyas instancias se vinculan. Las clases se sitúan en los extremos de la asociación. El nombre de la asociación se indica encima de la línea.

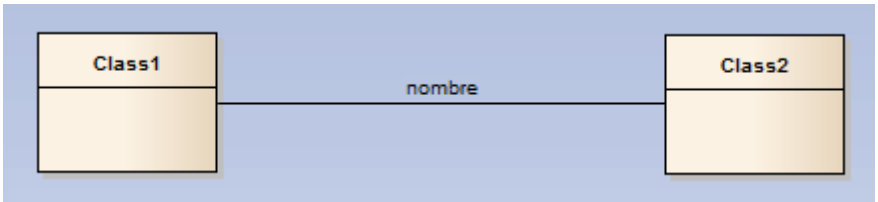

Ilustración 10 Asociación binaria entre clases

Para señalar el sentido de lectura del nombre de la asociación con respecto al nombre de las clases, la línea puede contener puntas de flecha.

Los extremos de una asociación pueden recibir un nombre. Dicho nombre es representativo de la función que desempeñan en la asociación las instancias de la clase correspondiente. Así, a la hora de traducir un diagrama a código, una clase tendrá un atributo cuyo tipo será la clase situada en el otro extremo de la asociación. Por consiguiente, puede ser pública o estar encapsulada de alguna de las maneras ya vistas. Cuando se especifican las funciones, muchas veces no es preciso indicar el nombre de la asociación, ya que éste suele ser el mismo que el de una de las funciones.

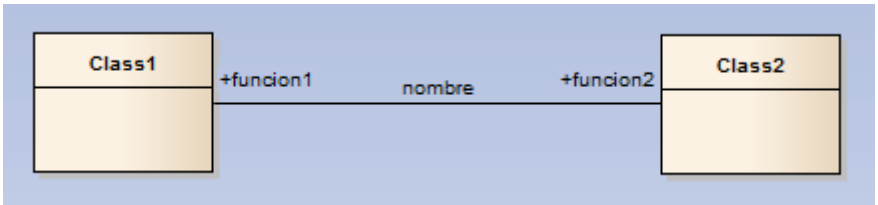

Ilustración 11 Funciones de una asociación binaria

La Ilustración 12 muestra la representación gráfica de las asociaciones introducidas en el ejemplo del apartado 3.1

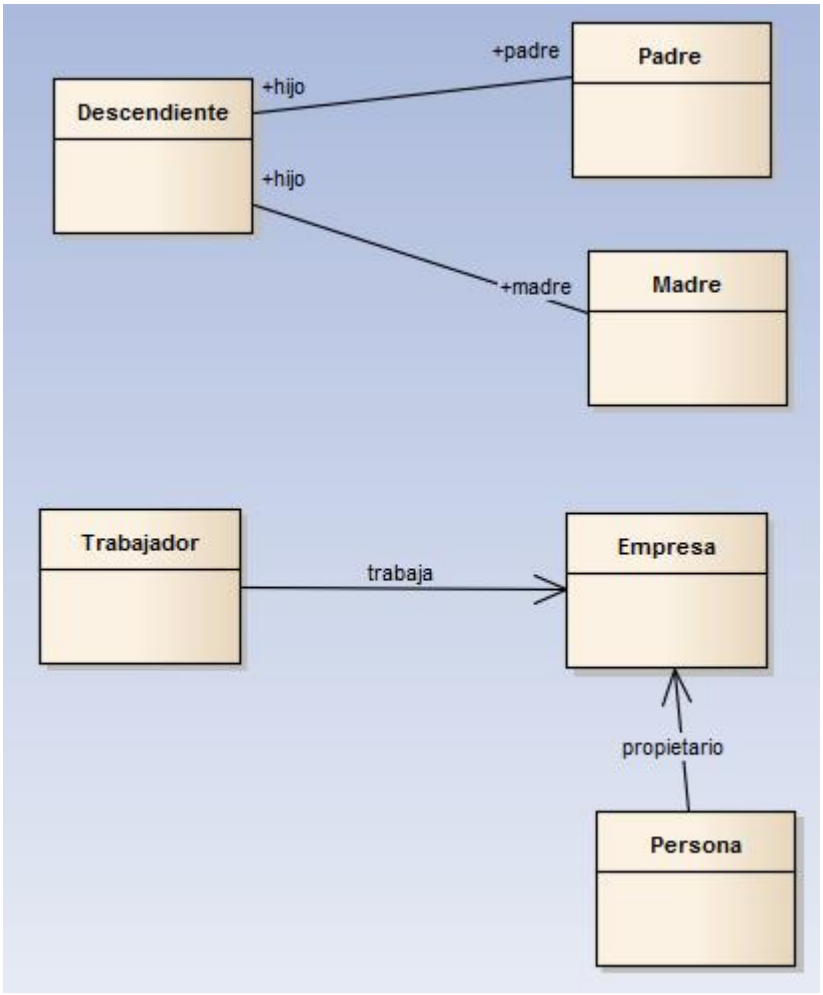

Ilustración 12 Ejemplos de asociaciones binarias

La representación gráfica de una asociación ternaria y superiores consiste en un rombo que une las diferentes clases.

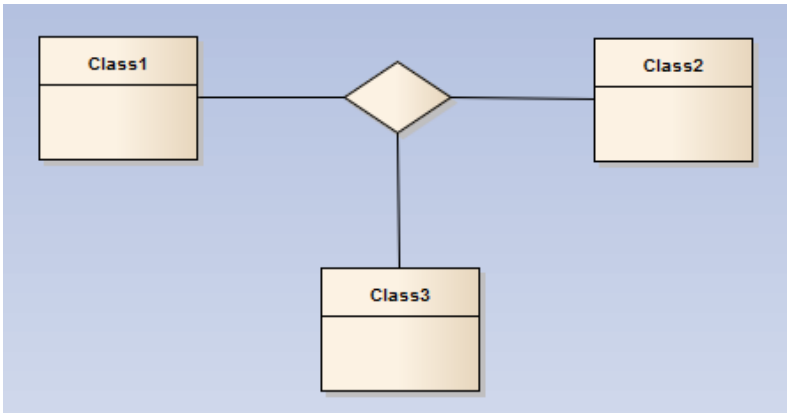

Ilustración 13 Asociación ternaria

En la asociación 'familia' que vincula las clases Padre, Madre y Descendiente, cada uno de los elementos constituye un triplete (madre, padre, hijo).

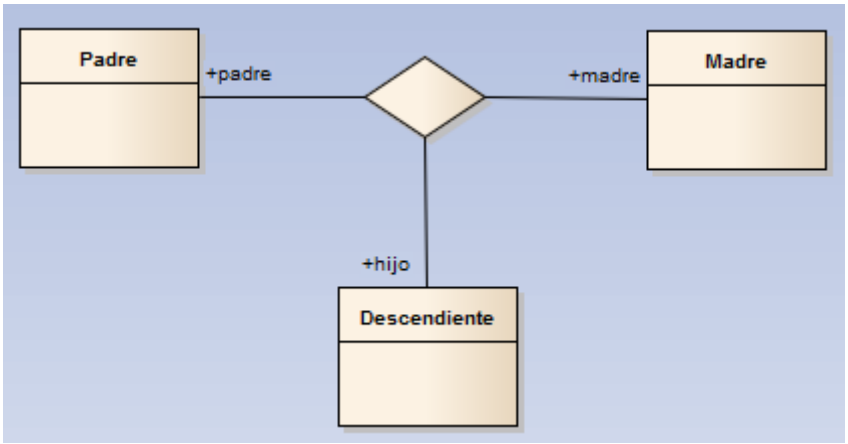

Ilustración 14 La asociación ternaria 'familia'

3.3 La cardinalidad o multiplicidad de las asociaciones

La cardinalidad situada en un extremo de una asociación indica a cuántas instancias de la clase situada en ese mismo extremo está vinculada una instancia de la clase situada en el extremo opuesto.

En uno de los extremos opuestos de la asociación, es posible especificar la cardinalidad mínima y la máxima, con el fin de indicar el intervalo de valores al que deberá pertenecer siempre la cardinalidad.

La siguiente tabla describe la sintaxis de especificación de las cardinalidades:

Especificación	Cardinalidad
0..1	Cero o una vez
1	Únicamente una vez
0..*	De cero a varias veces
1..*	De una a varias veces
M..N	Entre M y N veces
N	N veces

De no existir una especificación explícita de las cardinalidades mínimas y máximas, éstas valen 1.

Ejemplo: *La Ilustración 15 retoma los ejemplos, a los que se les agrega las multiplicidades de cada asociación. Una empresa puede tener varios propietarios, y una persona puede ser propietaria de varias empresas.*

Como ejemplo, la primera asociación debe leerse del siguiente modo: "un descendiente posee un solo padre. Un padre puede tener de cero a varios hijos".

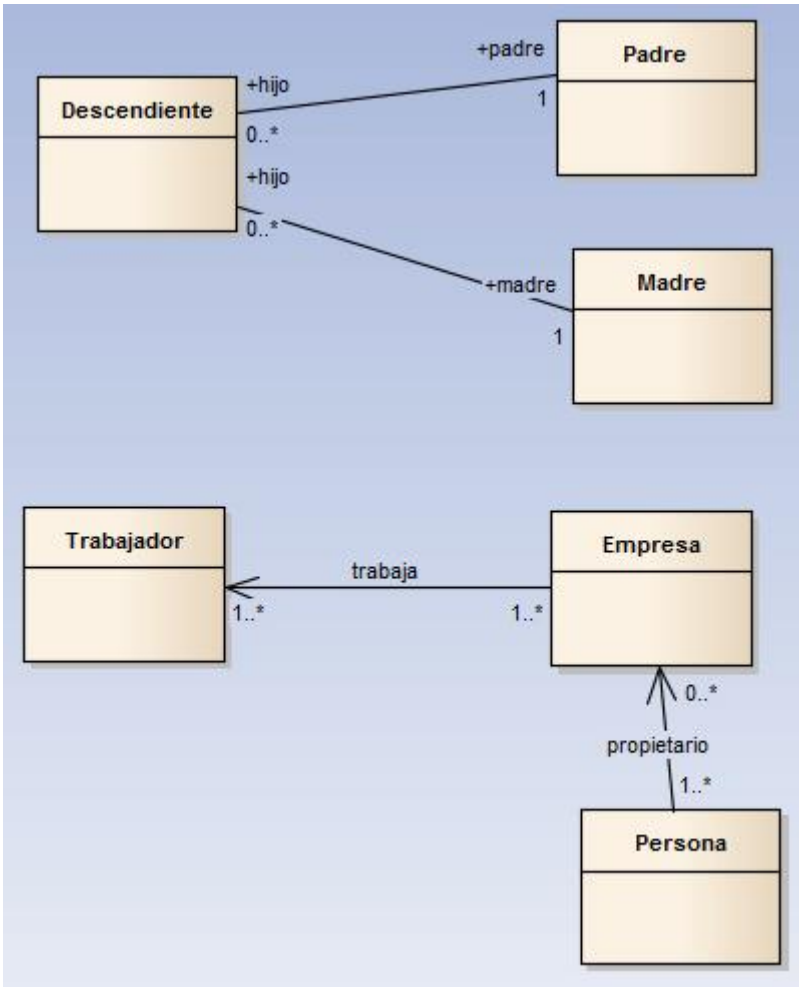

Ilustración 15 Ejemplos de asociaciones con cardinalidades

3.4 Navegación

Por defecto, las asociaciones tienen una navegación bidireccional, es decir, es posible determinar los vínculos de la asociación desde una instancia de cada clase de origen. Las navegaciones bidireccionales resultan más complejas de realizar y, en la medida de lo posible, conviene evitarlas.

Para especificar el único sentido útil de navegación se dibuja la asociación en forma de flecha.

Ejemplo: *en el caso de la empresa, resulta útil conocer los trabajadores que posee dicha empresa, pero lo contrario es innecesario.*

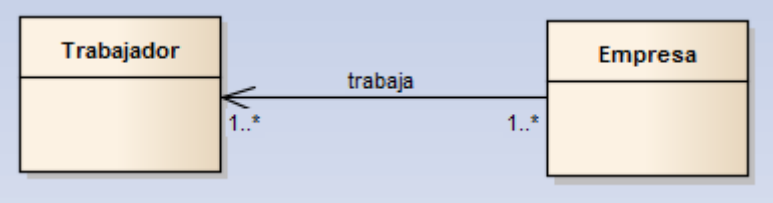

Ilustración 16 Ejemplo de navegación

3.5 Asociar una clase a sí misma

Cuando encontramos una misma clase en los dos extremos de una asociación, hablamos de asociaciones reflexivas que unen entre sí las instancias de una misma clase.

En estos casos, resulta preferible asignar un nombre a la función desempeñada por la clase en cada extremo de la asociación.

Las asociaciones reflexivas sirven principalmente para describir dentro del conjunto de instancias de una clase:

- Grupos de instancias (una asociación que representa una relación de equivalencia)
- Una jerarquía dentro de las instancias (una relación de orden).

Ejemplo*: Para poder participar en una carrera, es preciso que los corredores hayan participado en otras carreras previas. Podemos crear, por consiguiente, una asociación entre una carrera y las carreras celebradas previamente.*

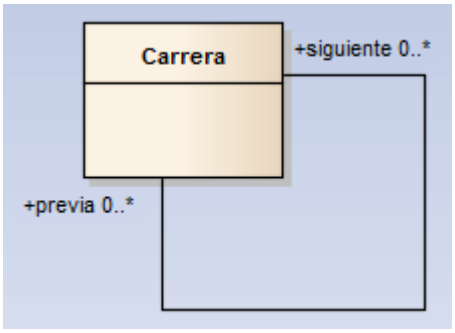

Ilustración 17 Asociación reflexiva entre carreras

Ejemplo: *La Ilustración 18 muestra la asociación "ascendente/descendiente directo" entre las personas. Esta asociación crea una jerarquía dentro de las personas.*

La cardinalidad para los ascendentes es 2, ya que cualquier persona tiene un padre y una madre.

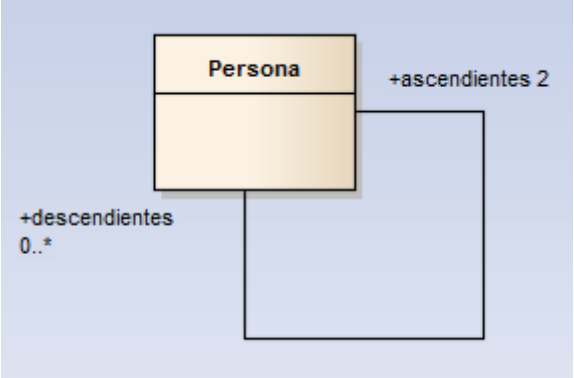

Ilustración 18 Asociación "ascendente/descendiente directo" entre personas

3.6 Las clases-asociaciones

Los vínculos entre las instancias de las clases pueden llevar informaciones. Éstas son específicas a cada vínculo.

En esos casos, la asociación que describe los vínculos recibe el estatus de clase y sus instancias son elementos de la asociación.

Al igual que el resto de clases, pueden estar dotadas de atributos y operaciones, así como estar vinculadas a otras clases a través de asociaciones.

Estas clases asociación se representan gráficamente uniéndolas a la línea de la asociación por medio de una línea discontinua.

Ejemplo: *Cuando un cliente compra productos en una conviene especificar la cantidad de productos adquiridos mediante una clase-asociación, aquí llamada clase Compra.*

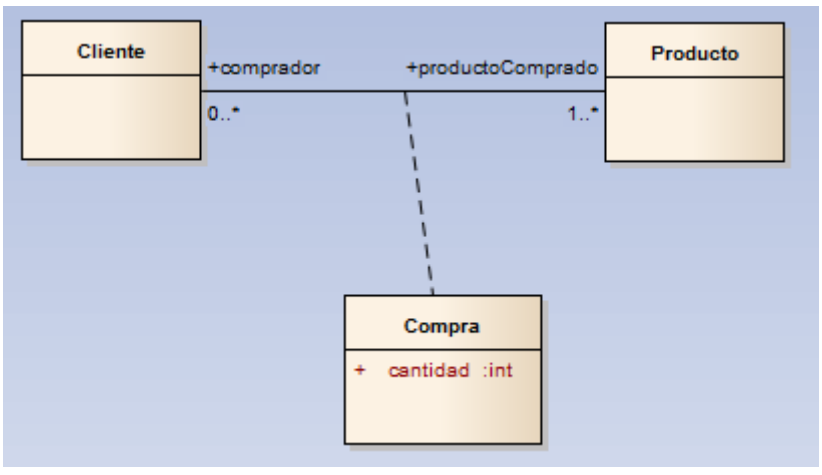

Ilustración 19 Clase-asociación Compra

3.7 Los objetos compuestos

Un objeto puede ser complejo y estar compuesto por otros objetos. La asociación que une a estos objetos es la composición, que se define a nivel de sus clases, pero cuyos vínculos se establecen entre las instancias de las clases. Los objetos que forman el objeto compuesto se denominan *componentes*.

Un caballo es un ejemplo de objeto complejo. Está formado por diferentes órganos (patas, cabeza, etc.).

En estos casos, nos encontramos ante una asociación entre objetos de un caso particular llamada *asociación de composición*. Ésta asocia un objeto complejo con los objetos que lo constituyen, es decir, sus componentes.

Existen dos formas de composición, fuerte o débil, que pasamos a examinar a continuación.

3.7.1 La composición fuerte o composición

La composición fuerte es una forma de composición en la que un objeto (el TODO) está formado por varios componentes, cuya vida está ligada a la del todo. De esta forma, los componentes no pueden ser compartidos por varios objetos compuestos. Por tanto, la cardinalidad máxima del objeto compuesto es obligatoriamente uno.

La desaparición del objeto compuesto comporta la desaparición de los componentes.

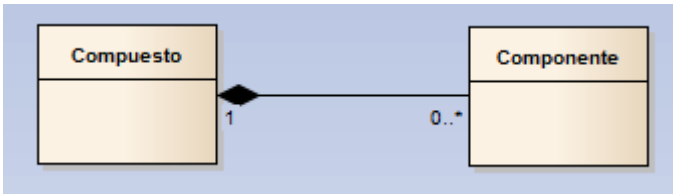

Ilustración 20 Asociación de composición fuerte

En adelante, la asociación de composición fuerte será denominada simplemente ***composición.***

Ejemplo*: Una persona está compuesta, entre otras cosas, por un cerebro. El cerebro no se comparte. La muerte de la persona comporta la muerte del cerebro.*

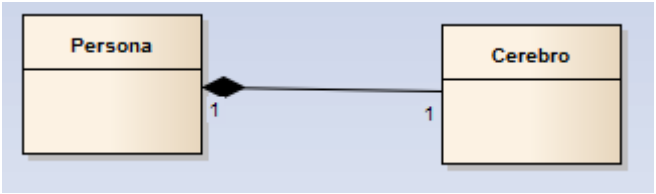

Ilustración 21 Asociación de composición entre una persona y su cerebro

Ejemplo: *Una carrera ciclista otorga premios. Los premios no se comparten con otras carreras, ya que son específicos de una carrera. Si la carrera no se organiza, los premios no se reparten y desaparecen, por lo que se trata de una relación de composición.*

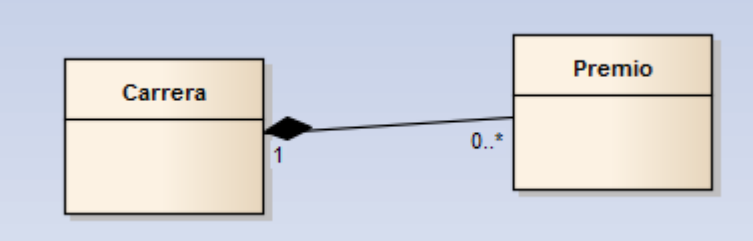

Ilustración 22 Asociación de composición entre una carrera ciclista y sus premios

3.7.1.1 *La composición débil o agregación*

La composición débil, llamada habitualmente agregación, impone muchas menos especificaciones a los componentes que la composición fuerte. En el caso de la agregación, los componentes pueden ser compartidos por varios compuestos (de la misma asociación de agregación o de varias asociaciones de agregación distintas) y la destrucción del compuesto no conduce a la destrucción de los componentes.

La agregación se da con mayor frecuencia que la composición. En las primeras fases del modelado es posible utilizar sólo la agregación y determinar más adelante qué asociaciones de agregación son en realidad de composición.

***Ejemplo**: un puerto está compuesto, entre otras cosas, por muelles, que a su vez están compuestos por pantalanes en los que atracan los barcos. Esta composición es una muestra de agregación. En efecto, la pérdida del puerto no acarrea la pérdida de los objetos y la pérdida de los pantalanes no acarrea la desaparición de los barcos.*

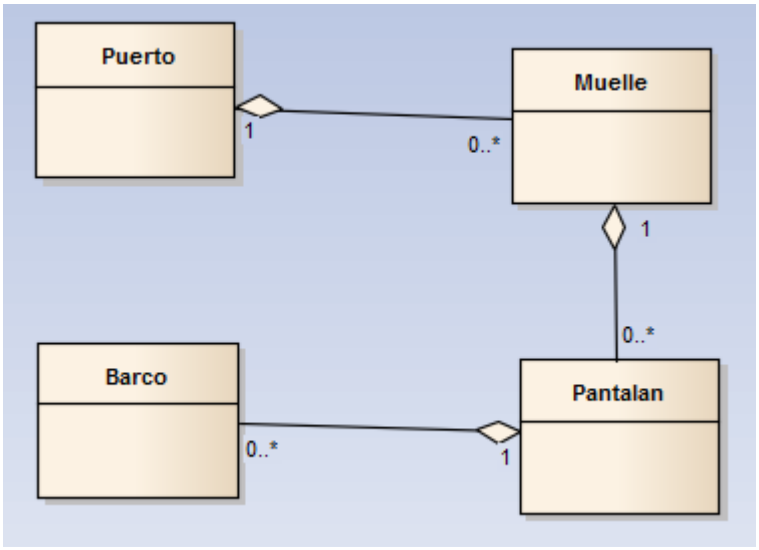

Ilustración 23 Asociación de agregación entre un puerto y su equipamiento y los barcos atracados en él.

***Ejemplo**: un propietario posee una colección de coches. Un coche pertenece a una sola colección y puede ser simultáneamente ser confiado a un cuidador o no serlo. Por tanto, puede ser componente de dos agregaciones.*

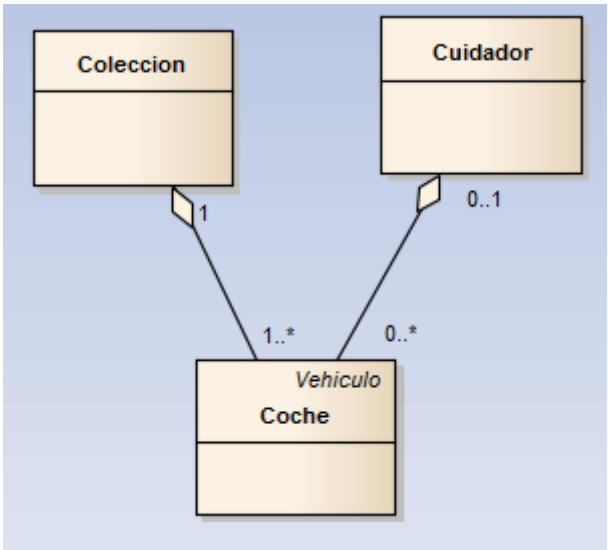

Ilustración 24 Componente compartido por varias agregaciones

***Ejemplo**: hilando más fino, un coche puede pertenecer a varios propietarios. En ese caso, la cardinalidad en la colección del propietario ya no es 1, sino 1..*. Por tanto, el coche puede ser compartido varias veces en una misma agregación.*

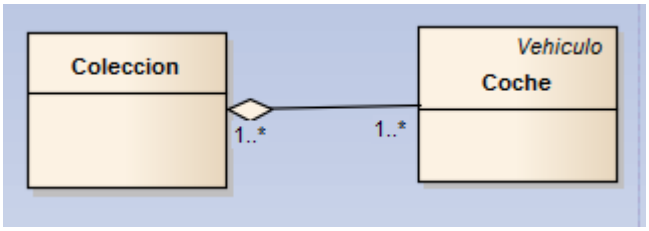

Ilustración 25 Componente compartido varias veces en una misma agregación

3.7.1.2 *Diferencias entre composición y agregación*

La siguiente tabla resume las diferencias entre ambos tipos de composición.

	Agregación	Composición
Representación	Rombo transparente	Rombo negro
Varias asociaciones comparten los componentes	Sí	No
Destrucción de los componentes al destruir el compuesto	No	Sí
Cardinalidad del compuesto	Cualquiera	0..1 ó 1

4 Elaboración de diagramas de clases

4.1 Herramientas

4.2 Ingeniería directa

4.3 Ingeniería inversa