UML II

Diagramas

Diagrama de casos de uso.

Diagrama de clases.

Diagrama de objetos.

Diagrama de secuencia.

Diagrama de colaboración.

Diagrama de estados.

Diagrama de actividades.

Diagrama de componentes.

Diagrama de despliegue.

Diagrama de casos de uso

El uso de los diagramas de casos de uso será, por lo general, parte de un documento de diseño que el cliente y el equipo de diseño tomarán como referencia de las acciones inmediatas que puede realizar el usuario final sobre el sistema o aplicación desarrollado.

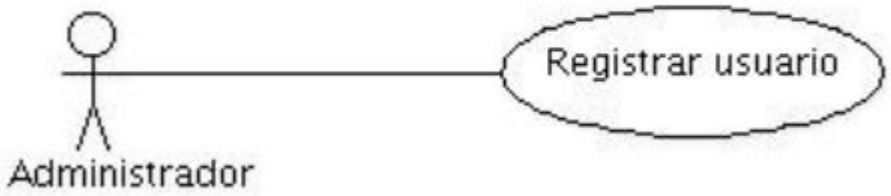
De esta manera se pueden explicar rápidamente conceptos que ayudaran a un analista a comprender como un sistema deberá comportarse. Para ello es necesario aprender a visualizar los conceptos involucrados dentro de los casos de uso para así poder colocar todos los actores en el escenario correspondiente, incluyendo las actividades que pueden producir en dicha situación.

Como se menciona anteriormente los diagramas de casos de uso se emplean para modelar la vista de casos de uso estática de un sistema. Esta vista cubre principalmente el comportamiento del sistema (los servicios visibles externamente que proporciona el sistema en el contexto de su entorno).

Actor

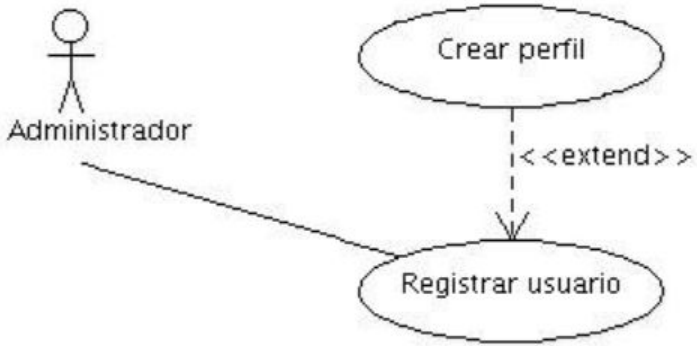
Un actor es una entidad externa al sistema que necesita intercambiar información con el sistema. La notación utilizada en UML para representar un actor es una figura humana con el nombre del actor.

Caso de uso



A diferencia de las técnicas de análisis estructurado, el diagrama de casos de uso no persigue modelar un flujo de datos. Un caso de uso puede definirse como un flujo completo de eventos en el que se especifica la interacción entre el actor y el sistema o como un negocio trabaja actualmente. La ejecución de un caso de uso termina cuando el actor genere un evento que requiera un caso de uso nuevo.

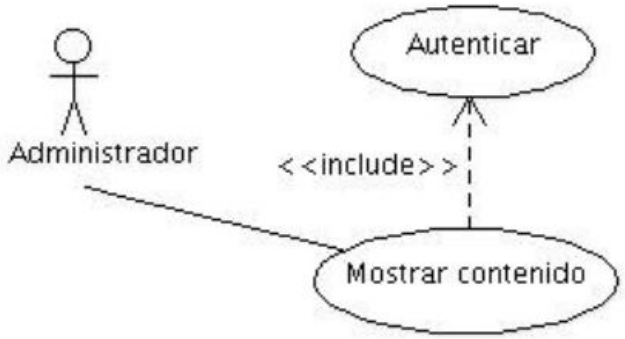
Extensión



Una extensión indica que la realización del caso de uso que extiende no es obligatoria durante la realización del caso de uso que está siendo extendido.

En el ejemplo que se muestra, el caso de uso Crear perfil puede o no ser realizado al ejecutarse el caso de uso Registrar usuario.

Inclusión



Una inclusión indica que un caso de uso se realizará incondicionalmente durante la realización del caso de uso que lo referencia.

En el ejemplo que se muestra, el caso de uso Autenticar debe ser realizado para que el caso de uso Mostrar contenido sea realizado correctamente.

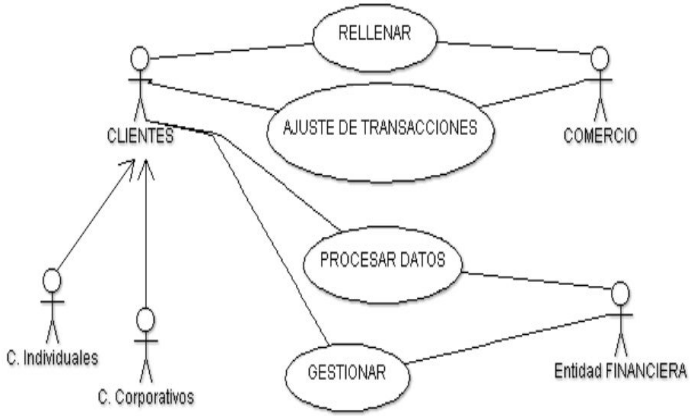
Funcionamiento

Cuando se modela la vista de casos de uso estática de un sistema, normalmente se emplearán los diagramas de casos de uso de una de las dos formas siguientes:

1) Para modelar el contexto de un sistema: Modelar el contexto de un sistema implica dibujar una línea alrededor de todo el sistema y asegurar qué actores quedan fuera del sistema e interactúan con él. Aquí, se emplearán los diagramas de casos de uso para especificar los actores y significado de sus roles.
2) Para modelar los requisitos de un sistema: El modelado de los requisitos de un sistema implica especificar qué debería hacer el sistema (desde un punto de vista externo), independientemente de cómo se haga. Aquí se emplearán los diagramas de casos de uso, para especificar el comportamiento deseado del sistema. De esta forma, un diagrama de casos de uso permite ver el sistema entero como una caja negra; se puede ver qué hay fuera del sistema y cómo reacciona a los elementos externos, pero no se puede ver cómo funciona por dentro.

Ejemplo del contexto de un sistema

La siguiente figura muestra el contexto de un sistema de validación de tarjetas de crédito, destacando los actores en torno al sistema. Se puede ver que existen Clientes, de los cuales hay dos categorías (Cliente individual y Cliente corporativo). Estos actores representan los roles que juegan las personas que interactúan con el sistema. En este contexto, también hay actores que representan a otras instituciones tales como Comercio (que es donde los Clientes realizan una transacción con tarjeta para comprar un artículo o servicio) y Entidad Financiera (que presta servicio como sucursal bancaria para la cuenta de la tarjeta de crédito). En el mundo real estos dos actores probablemente sean sistemas con gran cantidad de software.



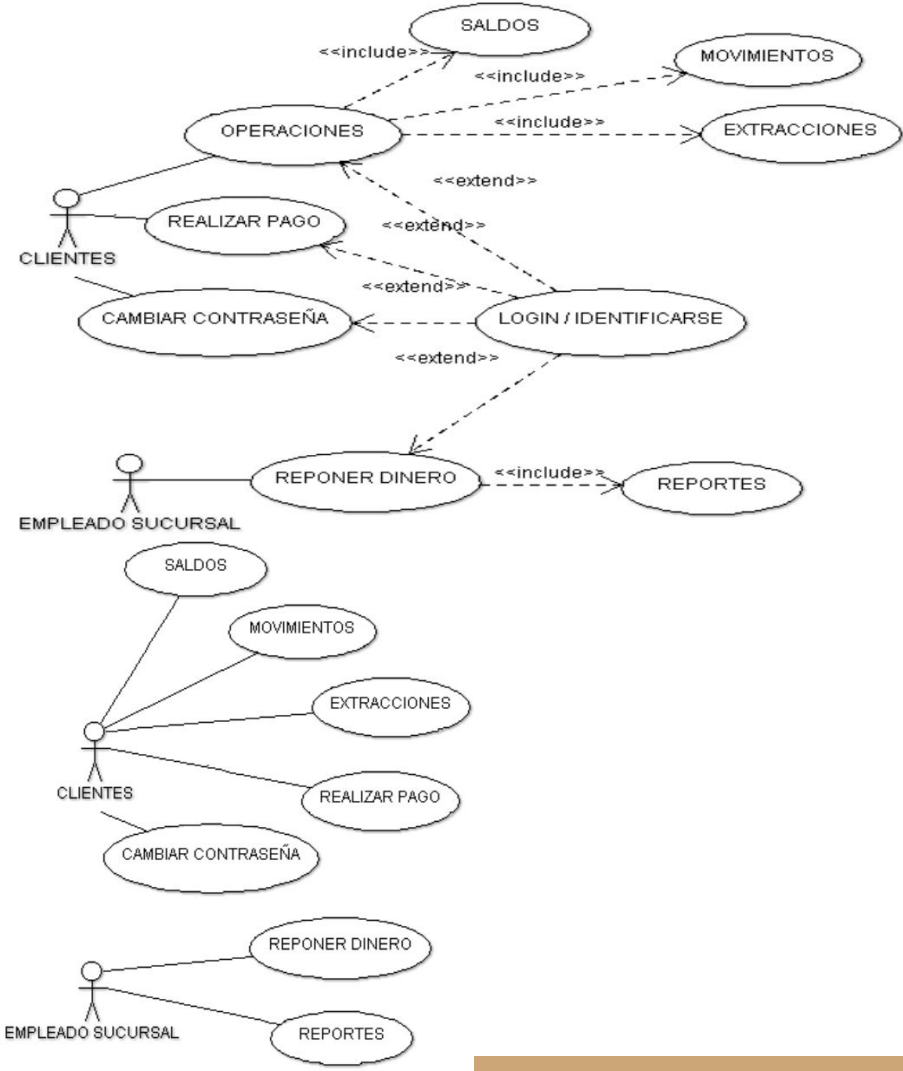
Ejemplo de los requisitos de un sistema

Continuando con el ejemplo anterior, en este caso, aunque omite las relaciones entre los actores y los casos de uso, añade casos de uso adicionales que son invisibles para el cliente normal, aunque son comportamientos fundamentales del sistema. Este diagrama es valioso porque ofrece un punto de vista común para los usuarios finales, los expertos del dominio y los desarrolladores para visualizar, especificar, construir y documentar sus decisiones sobre los requisitos funcionales del sistema.

Por ejemplo, Detectar fraude de tarjeta es un comportamiento importante tanto para el Comercio como para la Entidad Financiera. Análogamente, Informe estado de Cuentas, es otro comportamiento requerido del sistema por varias entidades.

Más ejemplos

Las figura de arriba muestra el caso de uso desplegado y como lo vería un desarrollador y la otra muestra el primer escenario de caso de uso que suele ser utilizado para instruir al usuario regular, cuando se realiza una exhibición de facultades del producto, como es el funcionamiento y las posibilidades que brinda la aplicación.



Ejemplo de modelado de requisitos (más allá del software)

El diagrama de casos de uso se puede utilizar no solo para representar esquemas informáticos, sino que también sirven para demostrar el modelado de requisitos de un sistema.

Ejemplo:

Te encargan realizar una aplicación para la compra-venta de cuadros. En cuanto a la compra de cuadros, una vez que el agente introduce unos datos básicos sobre el cuadro, el sistema debe proporcionar el precio recomendado que el agente de la galería debería pagar. Si el vendedor del cuadro acepta la oferta, entonces el agente de la galería introduce más detalles (sobre el vendedor del cuadro y la venta).

Los datos básicos incluyen el nombre y apellidos del artista, el título y fecha de la obra, sus dimensiones, la técnica (óleo, acuarela u otras técnicas), el tema (retrato, naturaleza muerta, paisaje, otro) y la clasificación (obra maestra, obra representativa, otro tipo). Si es obra maestra, el precio recomendado se calcula comparando el cuadro introducido con los que hay en el registro de cuadros, tomando el más parecido y aplicando un algoritmo que tiene en cuenta la coincidencia de tema, la técnica y las dimensiones del cuadro. El sistema debe utilizar información de subasta de todo el mundo que ahora la galería recibe en un CD de manera mensual. Para una obra representativa, el precio recomendado se calcula como si fuera una obra maestra y luego se aplica una corrección. Para una obra de otro tipo, se calcula utilizando el área del cuadro y un coeficiente de moda para el artista. Si no hay coeficiente de moda para un artista, el agente tiene por norma no comprar el cuadro. El coeficiente de moda varia de mes a mes...

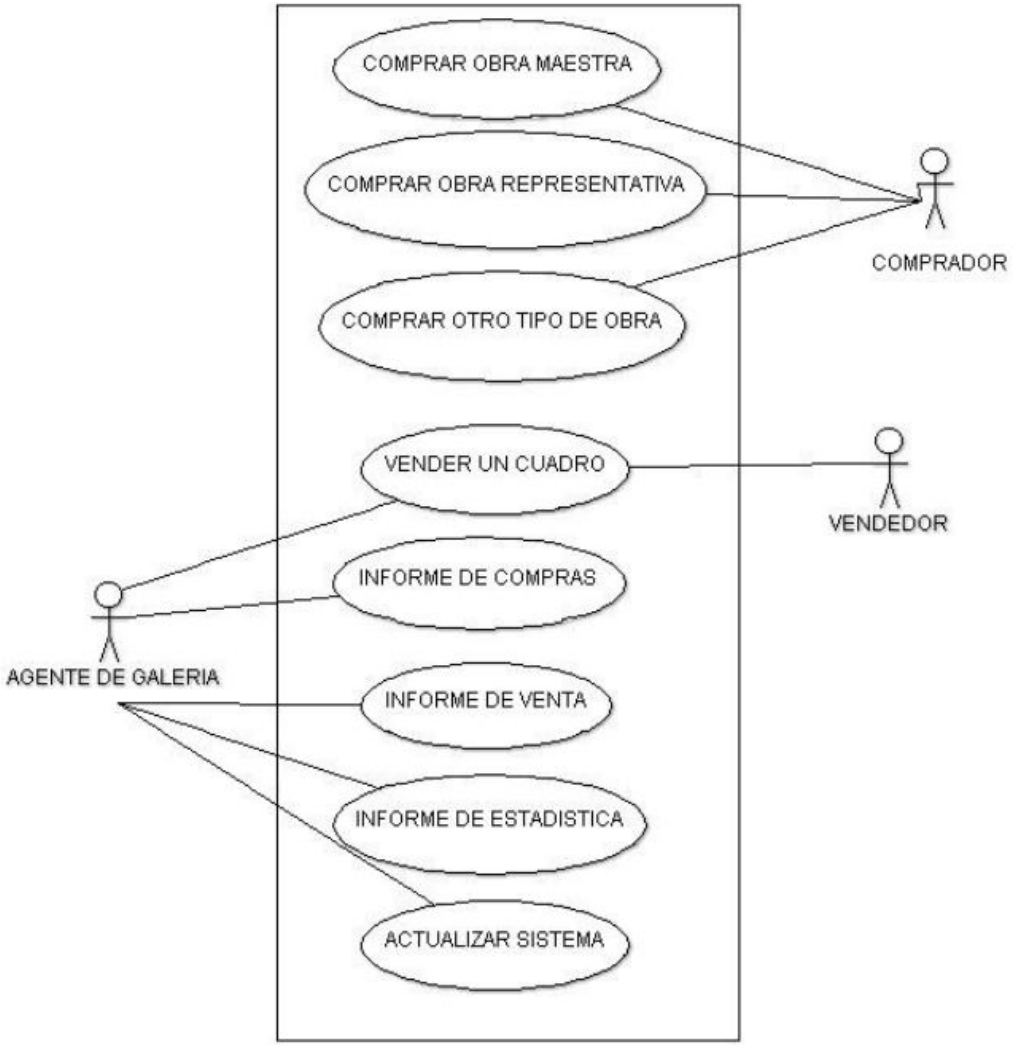
... sigue el ejemplo

Si el cuadro finalmente se compra, se introducen datos adicionales.

En cuanto a la venta de cuadros por parte de la galería, el sistema simplemente registra la fecha de venta, el nombre y dirección del comprador y el precio de venta real.

El sistema también deberá detectar nuevas tendencias en el mercado de arte tan pronto como sea posible. La idea es detectar secuencias de compras por valores mayores que los esperados por la obra de un artista determinado, de tal manera que tu cliente pueda comprar cuadros de ese artista antes de que otros detecten la tendencia. Con el objetivo de detectar cuándo el precio de venta es mayor que el precio esperado cuando tu cliente compró el cuadro, se debe mantener un registro de todas las compras y todas las ventas.

Se quieren generar tres informes: compras y ventas realizadas durante un año, y artistas de moda....



Se puede visualizar en el Diagrama de caso de uso de requisitos sobre las diferentes actividades en las cuales se ven relacionados los diferentes actores.
En tanto el agente galería se encuentra afectado por todos los casos de uso, su relación con el comprador solamente es la de "vender un cuadro", mientras que con el vendedor existen 3 tipos de "compras" posibles, según los detalles diagnosticados en el ejercicio.
Por lo detallado en el esquema se puede visualizar rápidamente dos aspectos fundamentales - 1) Los actores están en lados opuestos que por lo general significa cómo actúan sobre el sistema, uno realiza modificaciones internas (agente) mientras que los otros dos tienen actuaciones externas (vendedor, comprador). 2) Los tipos de compras que puede realizar el comprador están especificadas y no unificadas, esto se debe a que en cada una de las transacciones hay en específico una opción esencial, como por ejemplo si se llegase a vender una obra maestra, por los costos y valor intrínseco de la misma se deberán rellenar unos formularios especiales que requieran de donde provienen los fondos de dicha compra, quien es la persona o razón social afectada en esta transacción, etc.
Anteriormente habíamos declarado que en el caso en particular de un Primer escenario debemos ya que no utilizamos los extend o include debemos identificar los datos y acciones relacionadas en el diagrama por tal motivo se desarrolla el siguiente modelo de especificaciones.

Actores:

Agente Galería, vendedor, comprador.

Interesados y Objetivos:

- Agente: quiere obtener una recomendación lo más acertada posible del precio máximo recomendado de manera rápida.
- Vendedor: quiere vender el cuadro a un precio razonable de manera rápida.
- Comprador: quiere comprar los cuadros al mejor precio posible.

Precondiciones:

El agente ha entrado en la aplicación para una negociación, pero eso significa que los precios de los cuadros se pueden modificar.

Garantía de éxito (postcondiciones):

Se registra la venta. Se requiere la identificación de las partes ya sea compra(agente) venta(vendedor) u venta(agente) compra(comprador).

Escenario Principal de Éxito:

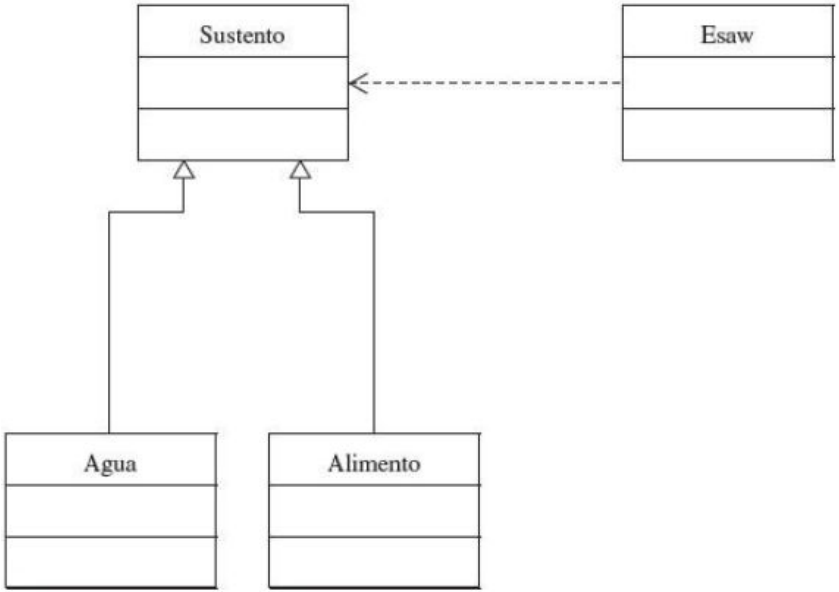
1. El agente introduce la descripción del cuadro para lo que se requiere identificación.
2. El sistema busca el cuadro más parecido del mismo autor.
3. El sistema presenta el precio recomendado, pero el agente es quien dispone del precio de la obra de arte.
4. El agente hace una propuesta por debajo del precio recomendado, y el vendedor acepta la oferta, ambos identificados.
5. El agente introduce información de la venta para lo cual deberá estar identificado.
6. El comprador, compra el cuadro teniendo en cuenta el precio del mismo puede funcionar en modo invitado, pero para comprar se debe identificar.

Diagrama de clase

Los diagramas de clases se usan para mostrar las clases de un sistema y las relaciones entre ellas.

Una sola clase puede mostrarse en más de un diagrama de clases y no es necesario mostrar todas las clases en un solo diagrama monolítico de clases. El mayor valor es mostrar las clases y sus relaciones desde varias perspectivas, de una manera que ayudará a transmitir la comprensión más útil.

Los diagramas de clases muestran una vista estática del sistema; no describen los comportamientos o cómo interactúan los ejemplos de las clases.



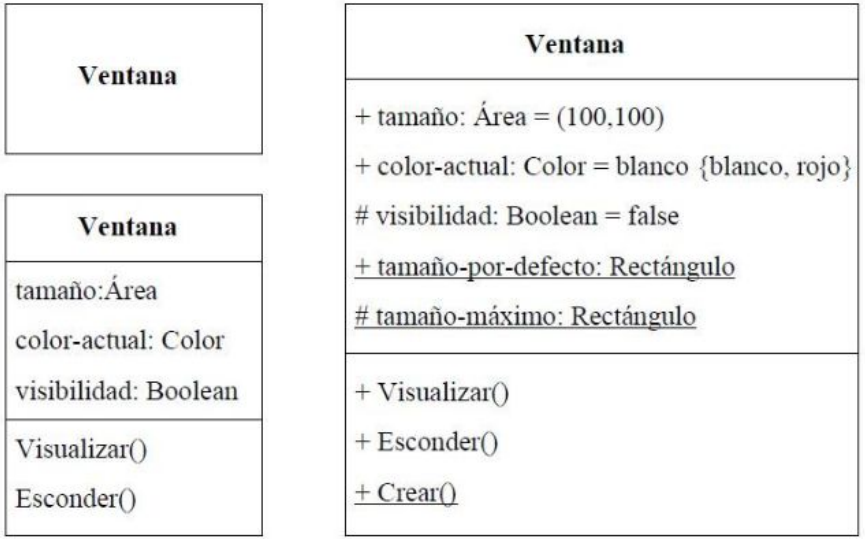
Un diagrama sencillo de clases, quizás uno de muchos, que transmite una faceta del sistema que se está diseñando.

Clases

Las clases describen un conjunto de objetos con características y comportamiento idénticos, es decir, objetos que comparten los mismos atributos, operaciones y relaciones.

Las clases se representan gráficamente por medio de un rectángulo con tres divisiones internas. Los tres compartimentos alojan el nombre de la clase, sus atributos y sus operaciones, respectivamente.

En muchos diagramas se omiten los dos compartimentos inferiores. Incluso cuando están presentes, no muestran todos los atributos y todas las operaciones. Por ejemplo, en la figura de la derecha viene representada la clase Ventana de tres formas diferentes: sin detalles, detalles en el ámbito de análisis y detalles en el ámbito de implementación.

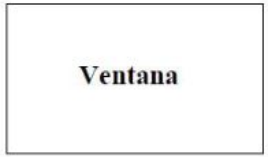


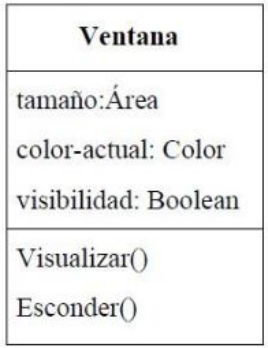
CLASE: Ventana con y sin detalles.

Compartimento del nombre

Cada clase debe tener un nombre que la distinga de las otras clases. Dicho nombre puede ser un nombre simple (nombre de la clase) o un nombre compuesto (nombre del paquete que contiene a la clase: nombre de la clase).

En este ejemplo, Ventana es un nombre simple de clase; pero si estuviese contenida en el paquete Gráficos, entonces el nombre compuesto sería Gráficos: Ventana. Otra forma de representar dicho nombre compuesto es escribir dentro de este compartimento primero el nombre del paquete (en este caso, «Gráficos») y debajo el nombre de la clase contenida en él.





Compartimento de la lista de atributos

Los atributos describen las características propias de los objetos de una clase.

La sintaxis completa de un atributo es:

[visibilidad] nombre [multiplicidad] [: tipo] [= valor-inicial] [{lista-propiedades}]

Componentes de los atributos (definición):

visibilidad puede ser:

•+ (pública): que hace el atributo visible a todos los clientes de la clase.

•- (privada): que hace el atributo visible sólo para la clase.

(protegida): que hace el atributo visible a las subclases de la clase.

*Si un atributo no tiene asignada ninguna visibilidad, quiere decir que la visibilidad no está definida (no hay visibilidad por defecto).

Atributos

Nombre: es una cadena de texto que representa el nombre del atributo.

Tipo: es un tipo de atributo típico: string, boolean, integer, real, double, point, área y enumeration. Se llaman tipos primitivos. También pueden ser específicos de un cierto lenguaje de programación, aunque se puede usar cualquier tipo.

Multiplicidad: es un indicador de la multiplicidad del atributo, que va encerrado entre corchetes. La ausencia de multiplicidad significa que tiene exactamente valor 1, mientras que una multiplicidad de 0..1 proporciona la posibilidad de valores nulos (la ausencia de un valor).

Valor inicial: es una expresión con el valor inicial que va a contener el atributo cuando se crea de nuevo el objeto.

Lista propiedades: es una lista que contiene los valores permitidos en un atributo. Se utiliza para especificar tipos enumerados tales como color, estado, impresión, etc.

Nota: Los atributos tamaño-por-defecto y tamaño-máximo tienen por tipo la clase Rectángulo, mientras que el resto corresponde a tipos primitivos: Área, Color y Boolean. Asimismo, estos dos atributos, que aparecen subrayados, son atributos cuyo ámbito es la clase.

Operaciones

Ejemplos de operaciones: En el ejemplo de la Figura anterior la clase Ventana tiene las siguientes operaciones:

+ Visualizar()

+ Esconder()

+ Crear()

En estas operaciones no se ha especificado ninguno de los argumentos mencionados antes, excepto el de la visibilidad. La operación Crear es una operación cuyo ámbito es la clase, pues aparece subrayada.

Relaciones de los diagramas de clase

ASOCIACION:

Es una relación de estructura entre clases, es decir, una entidad se construye a partir de otra u otras. Aunque este tipo de relación es más fuerte que la Dependencia es más débil que la Agregación, ya que el tiempo de vida de un objeto no depende de otro.

Representación UML:

Se representa con una flecha continua que parte desde una clase y apunta a otra. El sentido de la flecha nos indica la clase que se compone (base de la flecha) y sus componentes (punta de la flecha).

El nombre de la asociación es opcional y se muestra como un texto que está próximo a la línea. Se puede añadir un pequeño triángulo negro sólido que indique la dirección en la cual leer el nombre de la asociación. En el ejemplo de la Figura se puede leer la asociación como “director manda sobre Empleado”.

Relaciones de los diagramas de clase

MULTIPLICIDAD:

La multiplicidad es una restricción que se pone a una asociación, que limita el número de instancias de una clase que pueden tener esa asociación con una instancia de la otra clase. Indica cuántas instancias de una clase pueden surgir fruto de la relación, y se indican mediante:

1 --> sólo uno.

0..1 --> cero o uno.

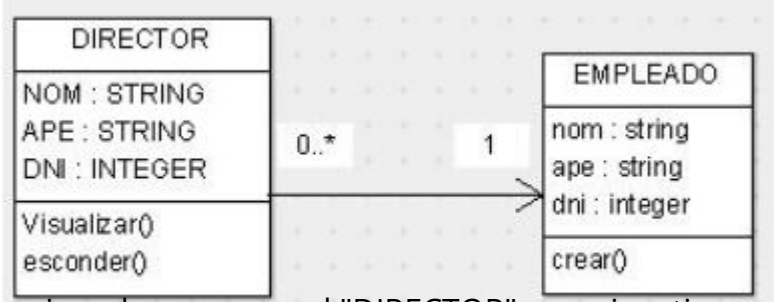
n --> mediante un número entero se indica cuántas relaciones pueden haber.

n..m --> varios a varios.

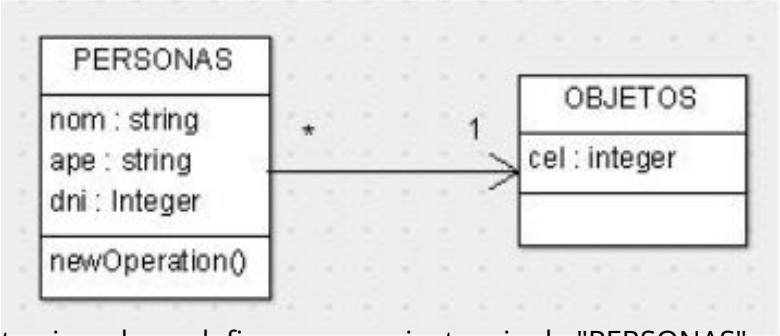
* --> cero o más.

0..* --> cero o más (lo mismo que el anterior).

1..* --> uno o más.

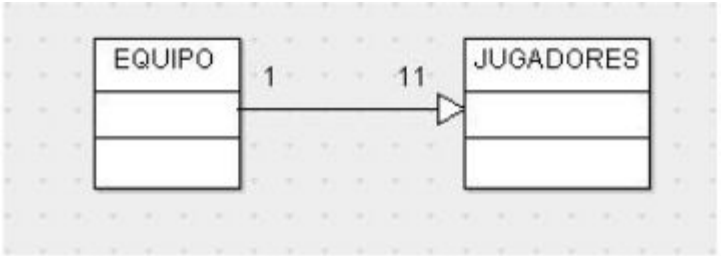


En este ejemplo se ve que el "DIRECTOR" maneja o tiene a su responsabilidad 0 o más empleados, en este caso por el lado de "EMPLEADO" vemos que hay 1 sola dependencia que es "DIRECTOR".



En este ejemplo se define que una instancia de "PERSONAS" puede estar relacionada sólo con 1 "OBJETOS", que a su vez puede estar relacionada con varias instancias de "PERSONAS".

En este tercer caso en particular, la instancia de "CLUB" puede estar relacionada con varias de "Personas", y cada una de ellas puede estar relacionada con varias de "CLUB".

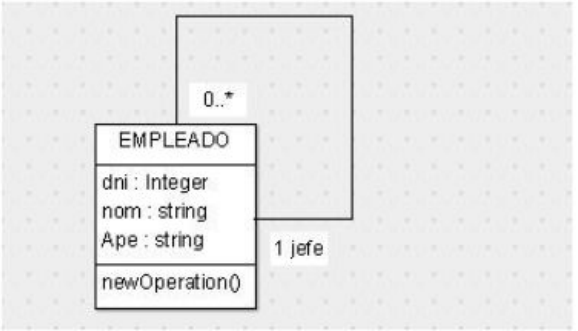


En el último caso, una instancia de "EQUIPO" puede estar relacionada con 11 instancias de "JUGADORES", y cada una de ellas a su vez con sólo una de "EQUIPO".

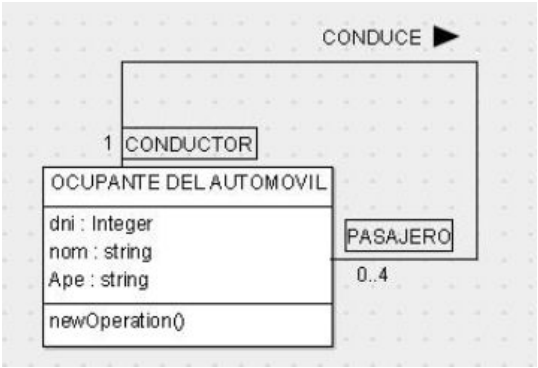
Relaciones de los diagramas de clase

ASOCIACION REFLEXIVA:

Por otro lado, si la asociación es entre objetos de la misma clase nos encontramos ante una asociación reflexiva. En la imagen se muestra que un empleado jefe está relacionado con ninguno o varios empleados, y que un empleado tiene sólo un jefe:



Imaginemos entonces por un momento un Automóvil que puede tener varios ocupantes, uno puede ser el "CONDUCTOR" y otro "PASAJERO". En el papel del conductor, el OCUPANTE_DE_AUTOMOVIL puede llevar ninguno o más clases del mismo objeto, por lo tanto representaremos con una asociación reflexiva como se asocian dichos roles u objetos en el mismo cuadrante.



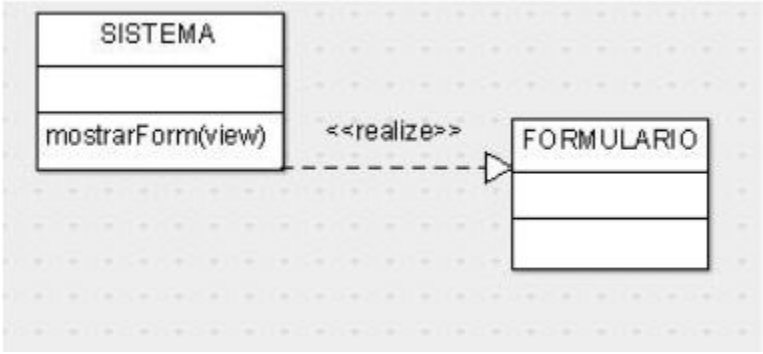
Relaciones de los diagramas de clase

DEPENDENCIA:

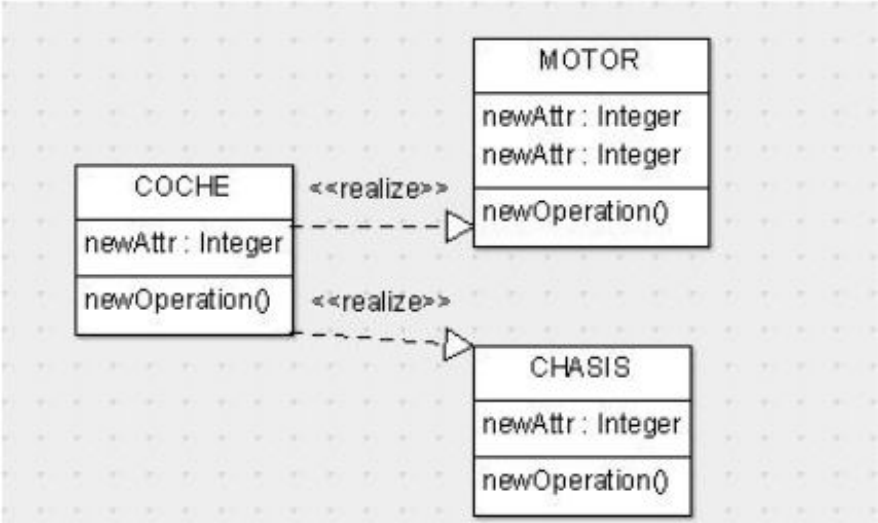
Es otro tipo de relación, una clase utiliza o requiere de otra, el uso más común de una dependencia es mostrar que la firma de la operación de una clase utiliza a otra clase.

Supongamos que un sistema tiene 1 o varios formularios corporativos para ser visualizados por pantalla y ser completados. En este tipo de diseños, tendremos una clase "SISTEMA" y una clase "FORMULARIO". Entre algunas de sus muchas operaciones, la clase "SISTEMA" va a permitir mostrarform. El "FORMULARIO" que el sistema será capaz de desplegar, dependerá directamente del tipo que seleccione el usuario.

La notación que utiliza UML para este tipo de elementos es una línea discontinua con una punta de flecha en forma de triángulo sin relleno que apunta a la clase de la que depende, como se muestra en la figura de acá a la derecha.



En la Dependencia entre dos clases uno de los elementos es dependiente ("COCHE" según la imagen de ejemplo) del otro, viéndose afectado por los cambios que se produzcan en él y no pudiendo funcionar correctamente si le falta.
Ello no implica que el elemento dependiente deba obligatoriamente contener al otro como uno de sus atributos (puede que lo necesite como parámetro o que lo utilice en un método, por ejemplo).
En lugar de la Dependencia es recomendable definir la relación como Asociación, a menos que sea importante representarla como tal debido a que un cambio en un elemento del que depende(n) otro(s) pueda afectar al funcionamiento del elemento dependiente.

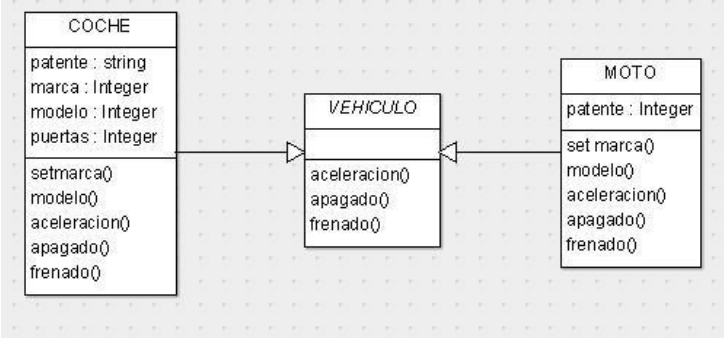


Relaciones de los diagramas de clase

GENERALIZACIÓN O HERENCIA:

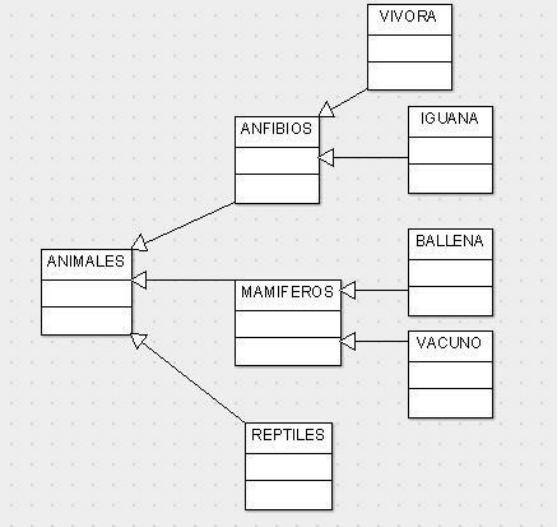
En cuanto a la Generalización, indica una relación de herencia entre dos clases u objetos (el elemento hijo comparte la misma estructura que el elemento madre).

Como se muestra en esta imagen "VEHICULO" hereda sus propiedades aceleración, apagado, frenado a sus dos clases hijos "COCHE" y "MOTO" ambos son vehículos con sus propiedades pero que están ampliamente vinculadas con las de la clase madre.



En el sentido amplio de la programación tener el conocimiento sobre un objeto cuya categoría de cosas, automáticamente sabrá qué elementos podrán ser transferidos a otra clase.

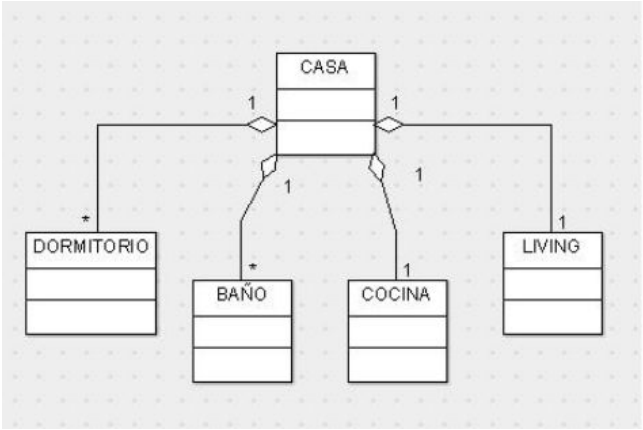
La jerarquía de la herencia no tiene que finalizar en dos niveles, es decir que una clase secundaria puede nacer de otra clase secundaria. Imaginemos el caso de la clase "ANIMALES" donde sus clases secundarias serian "ANFIBIOS" "MAMIFEROS" "REPTILES" luego de esta clase secundaria se genera una nueva clase dependiste de "REPTILES" llamada "VIVORA""IGUANA" y de la clase "MAMIFEROS" se desprenden "BALLENA" "VACUNO" etc.



Relaciones de los diagramas de clase

AGREGACIÓN:

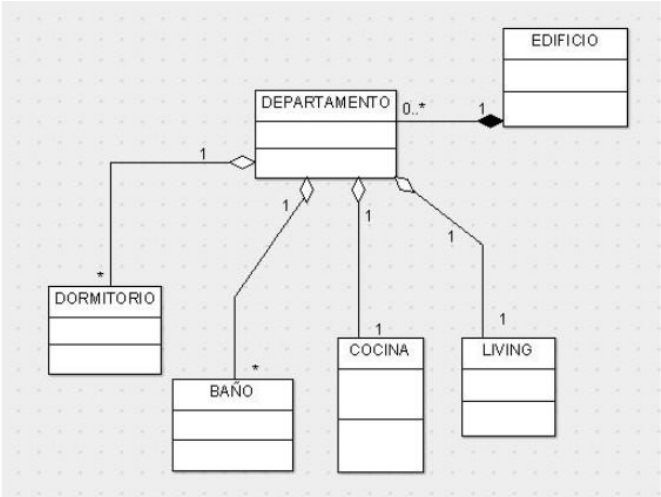
En ocasiones una clase consta de otras clases. Este es un tipo especial de relación conocida como agregación. Los componentes y la clase que constituyen son una asociación que conforma un bloque completo. En la Agregación el conjunto de elementos relacionados forma un "TODO", aunque los elementos relacionados podrían existir y "funcionar" independientemente de él. Se representan mediante una línea con un rombo sin relleno en un extremo.



Relaciones de los diagramas de clase

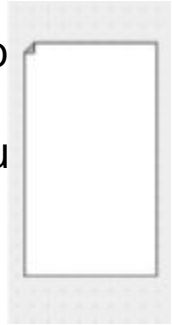
COMPOSICIÓN:

Una composición es un tipo muy representativo de una agregación. Cada componente dentro de una composición puede pertenecer tan solo a un todo. Por ejemplo, pensemos en una mesa de café, tanto la superficie de la tabla como las patas establecen una composición, no existe una mesa sin esos componentes juntos conformando en unidad un TODO, los elementos relacionados no podrían existir y "funcionar" independientemente si no formasen parte de dicho "TODO". El símbolo de una composición es el mismo que el de una agregación, excepto que el rombo está relleno es decir es de color negro.



Notas

Muchas veces vamos a requerir agregar algún texto co información a nuestros diagramas de clases y objetos, insertaremos el mismo en un rectángulo con una de su esquina doblada:



Las notas pueden estar en cualquier parte del diagrama sin mantener relación con ningún elemento en particular, aunque se utilizar una línea discontinua para especificar a qué elemento hacen referencia: