Técnico en Desarrollo de Software

Introducción a UML

Galileo UNIVERSIDAD | Instituto von Neumann

Introducción a UML

Unidad I

1. Introducción al UML

El UML (Lenguaje Unificado de Modelado) es un lenguaje de modelado, diseñado para la modelación de sistemas de software. Es una herramienta gráfica, con orientación a objetos, con la cual podemos construir, diseñar, especificar y visualizar sistemas, de un modo que sea fácil de entender y comunicar.

Una de las grandes ventajas del UML es su facilidad para representar una gran cantidad de situaciones, desde las definiciones de las clases que conforman nuestro sistema, las formas de interactuar con él, los componentes, hasta el detalle que lleva cada interacción, incluyendo mensajes, condiciones y quienes están involucrados.

UML nace del trabajo en conjunto de Grady Booch, James Rumbaugh e Ivar Jacobsor quienes habían desarrollado sus propias metodologías para el diseño orientado a objeto y que a mediados de los años noventa decidieron desarrollar su trabajo en conjunto.

UML ha pasado por varias versiones y se ha convertido en la especificación más aceptada para el análisis y diseño de sistemas.

Porque es importante el UML

Tomemos de referencia los sistemas informáticos de hoy en día, conforme avanza el mundo y los sistemas informáticos avanzan para estar presentes en muchos aspectos de nuestra vida, éstos se vuelven de una complejidad mayor. Hoy en día estos pueden involucrar elementos de hardware, software, comunicación por red, uso de servicios de la nube, grandes cantidades de información, etc.

Por ejemplo, las autoridades de una educación académica quieren mejorar el proceso de control de ingreso a salones dentro de su institución. Para esto las autoridades contactan a un analista de sistemas quien inicia un diálogo con las diferentes personas involucradas en el proceso a mejorar (Recolección de requerimientos).

El analista de sistemas debe asegurar haber captado las necesidades del cliente y de transmitir dichas necesidades al equipo de desarrollo de software encargado de construir dicho sistema (diseño y desarrollo).

Concluido el sistema un equipo de expertos en pruebas de sistemas debe ser capaz de interactuar con el nuevo software creado y verificar que cada una de las necesidades del cliente hayan sido alcanzadas. Y por último un equipo

de desarrollo (puede ser distinto al equipo que creó el sistema) estará a cargo del mantenimiento y mejora continua del sistema creado.

El caso mencionado anteriormente por muy sencillo que sea puede llegar a involucrar varios componentes de tecnología y personas en el proceso. Entre estos se encuentran:

Expertos del negocio:

- Directivos
- Profesores
- Alumnos
- Invitados

Proceso de desarrollo:

- Analista de sistemas.
- Grupo de desarrolladores.
- Diseñadores de infraestructura.
- Diseñadores de interacción de usuario.
- Diseñadores Gráficos.
- Expertos en pruebas.

Para poder crear un sistema con un nivel de complejidad considerable la clave es organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas comprendan y convengan con él. UML proporciona las herramientas para organizar un diseño y generar un modelo

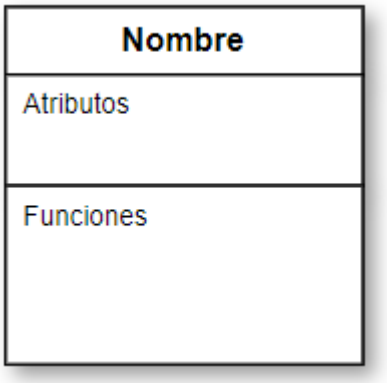
sólido del sistema que todos los involucrados en el proceso puedan comprender e incluso detectar con anticipación si la idea original no fue captada con precisión.

2. Diagramas del UML

Diagrama de Clases

Los diagramas de clase se usan para describir la estructura de un objeto

En este caso, un objeto es una entidad que pertenece al sistema, ya sea una persona, un computador, un componente, una interfaz, etc.



Una clase es representada como un rectángulo, puede tener hasta tres categorías de interés

Un Nombre que la identifica.

La central contiene atributos que describen la clase, cosas como nombre, número de identificación, descripción, etc.

La parte inferior contiene funciones, o métodos, que son las cosas que una clase puede hacer, cosas como enviar mensajes, recibir tareas, crear programas, etc,

Los métodos llevan al final un par de paréntesis

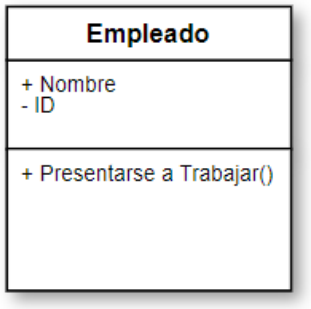
Ej. *escribir()*

Visibilidad de los Atributos y Métodos de una clase

Los atributos de una clase pueden ser visibles para sí mismo, para todas las clases, o para clases que estén relacionadas

- Un atributo que tenga un símbolo "+" es público, por lo que cualquier clase puede verlo
- Un atributo que tenga el símbolo "-" es privado, por lo que solo la clase misma puede verlo
- Un atributo que tenga el símbolo "#" es protegido, esto significa que solo la clase y las clases que hereden de esta pueden ver el atributo.
-

La misma simbología se usa para describir a los métodos



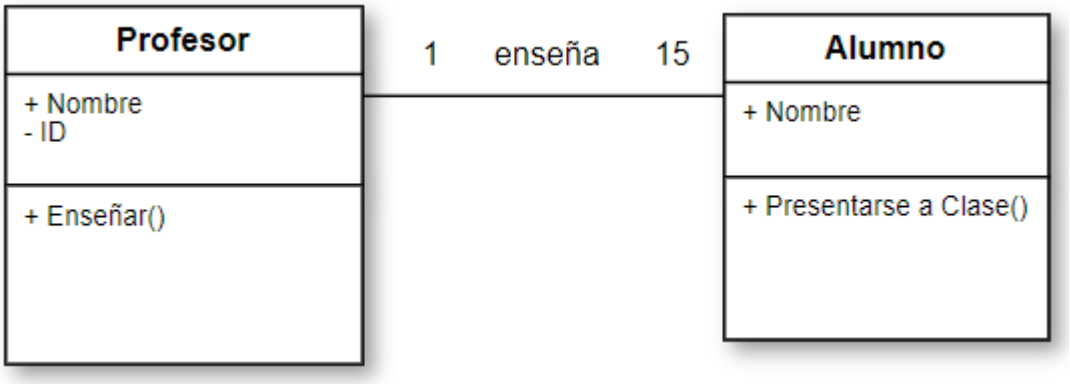
Una clase con un atributo y un método público, además de un atributo privado

Relaciones entre clases

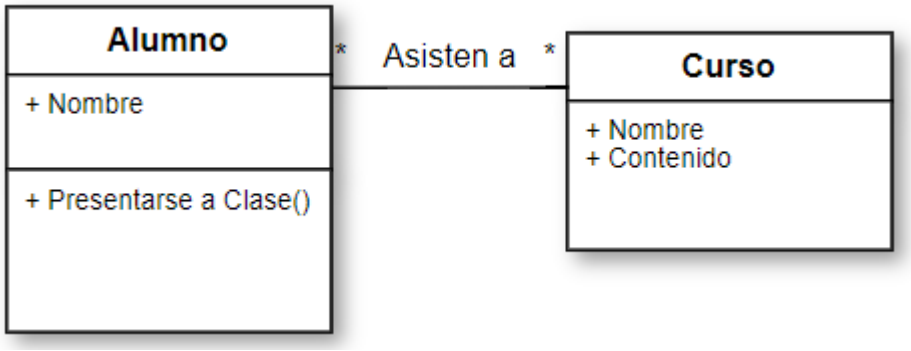
A veces las clases que creamos están relacionadas con otras clases, esto lo representamos por medio de distintos tipos de líneas

Multiplicidad

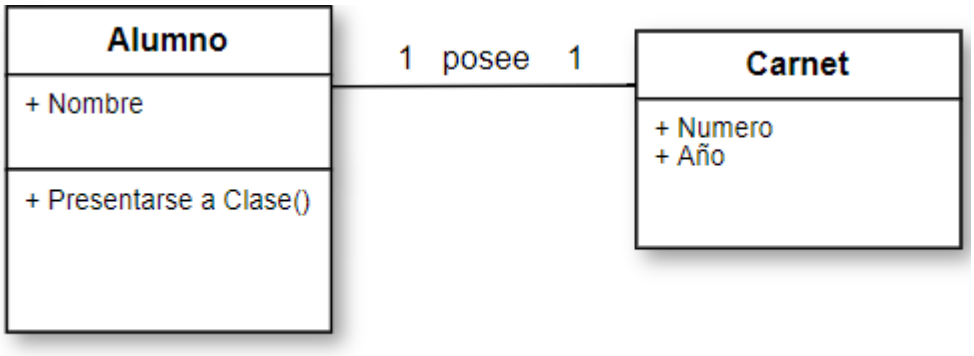
La multiplicidad en una relación nos indica cuántas instancias de la clase estarán presentes, o más bien nos indica que límites o restricciones tendremos en el diseño de nuestro proyecto. Existen varios tipos de multiplicidad.



Uno a varios



Varios a varios



Uno a uno

La multiplicidad de una clase puede ser también más específica, por ejemplo

- *"2..*"* indica que la clase puede ser dos a más, estableciendo un límite inferior
- *"2 ..5"* indica que solo pueden haber de 2 a 5 clases en esa relación

Los límites establecidos por la multiplicidad son de suma importancia ya que generalmente son restricciones del equipo o del proyecto en sí. Como en el ejemplo anterior, un alumno tendrá sólo un carnet, de otro modo podría ocasionar problemas, el segundo carnet podría ser usado por personas no autorizadas. Otro ejemplo, una clase puede tener de 20 a 30 alumnos, esto puede ser por el tamaño de la clase, que no permite más asientos.

Otros tipos de relaciones

Si bien las relaciones simples nos permiten diseñar muchos escenarios, en ocasiones nos interesa ser más específicos en el escenario que estamos

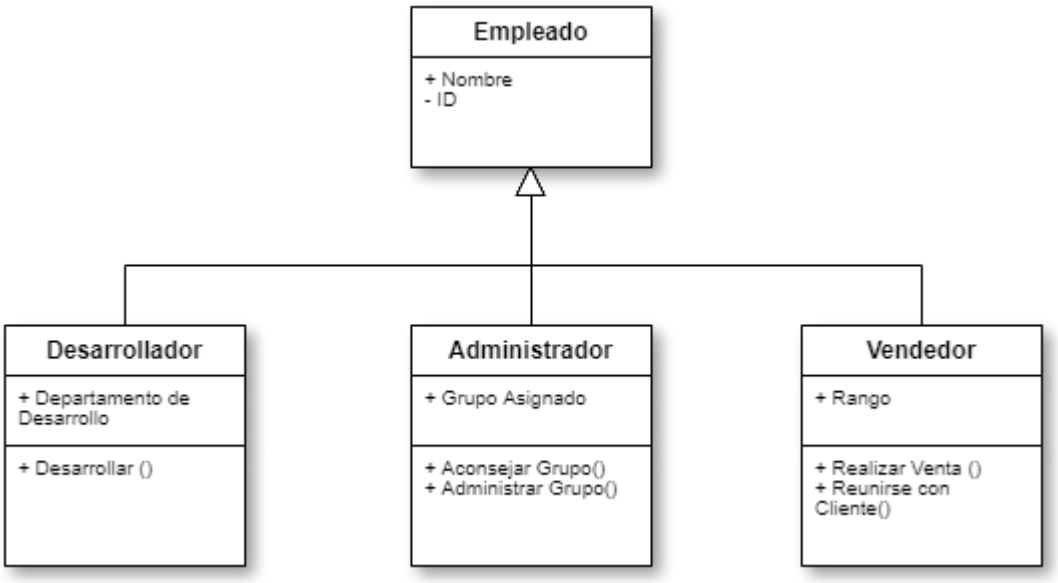
diseñando, veamos otros tipos de relaciones, en que casos podemos usarlos y que información extra nos proveen.

Generalización

También conocida como herencia o especialización. Una generalización surge cuando tenemos varias clases que cumplen roles similares, y comparten algunas características, entonces podemos crear una abstracción y crear una clase base, que contenga las características que comparten, y luego usando esa clase base como modelo generar las clases especializadas que tendrán únicamente los atributos y métodos que sean únicos para esa clase.

La generalización se reconoce por un triángulo blanco en el extremo de la relación que esta unido a la clase base. Una generalización no tendrá multiplicidad.

Veamos un ejemplo



En este caso, tenemos varias clases de empleados, todos tendrán un ID y el nombre del empleado, sin embargo sabemos que diferentes empleados tienen diferentes responsabilidades, áreas de trabajo, y actividades, por eso vamos a realizar una generalización, crear una base, y así poder tratar a todos como empleados, pero sabiendo que cada uno tendrá un comportamiento diferente.

Generalización es el nombre que se le da cuando el proceso se realiza de la manera que se acaba de explicar, cuando se genera una clase base al ver que varias clases comparten muchos atributos y métodos. En cambio, la especialización es el proceso opuesto, vemos que tenemos una clase, pero queremos que tenga varios comportamientos distintos. Entonces la separamos en varias clases, que aún comparten algunos atributos básicos, pero cada uno tiene métodos y atributos únicos, en este caso se creó de una base de varias especializaciones.

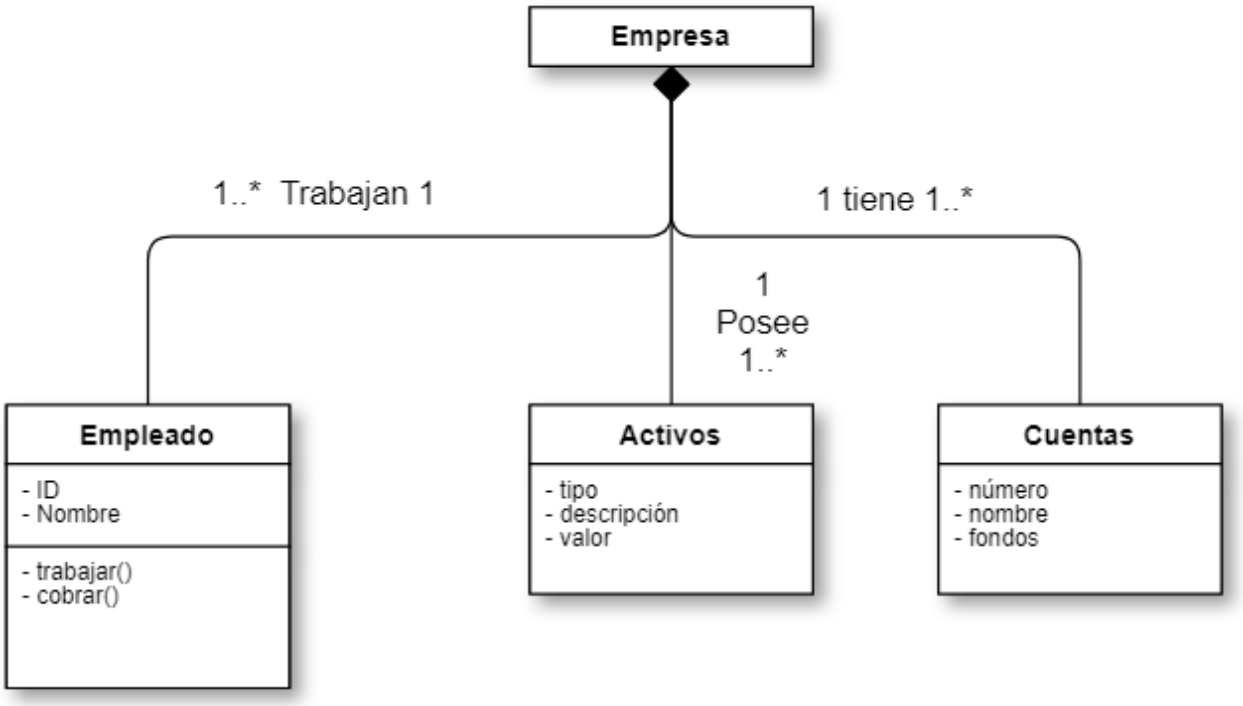
Ambos procesos terminan con el mismo resultado por lo tanto en el diagrama final ambos términos son intercambiables.

Composición

Una composición surge cuando tenemos varias clases, que nos interesa trabajen juntas, estas clases forman un conjunto en el que trabajarán. La clase que los contiene se la conoceremos como clase contenedora, cuando nos refiramos a esta clase, se entiende que también hablamos de cualquiera de los componentes.

A esto se le llama composición. La composición la reconoceremos con el rombo o diamante negro en el extremo de la relación que esta en el contenedor.

Veamos un ejemplo.

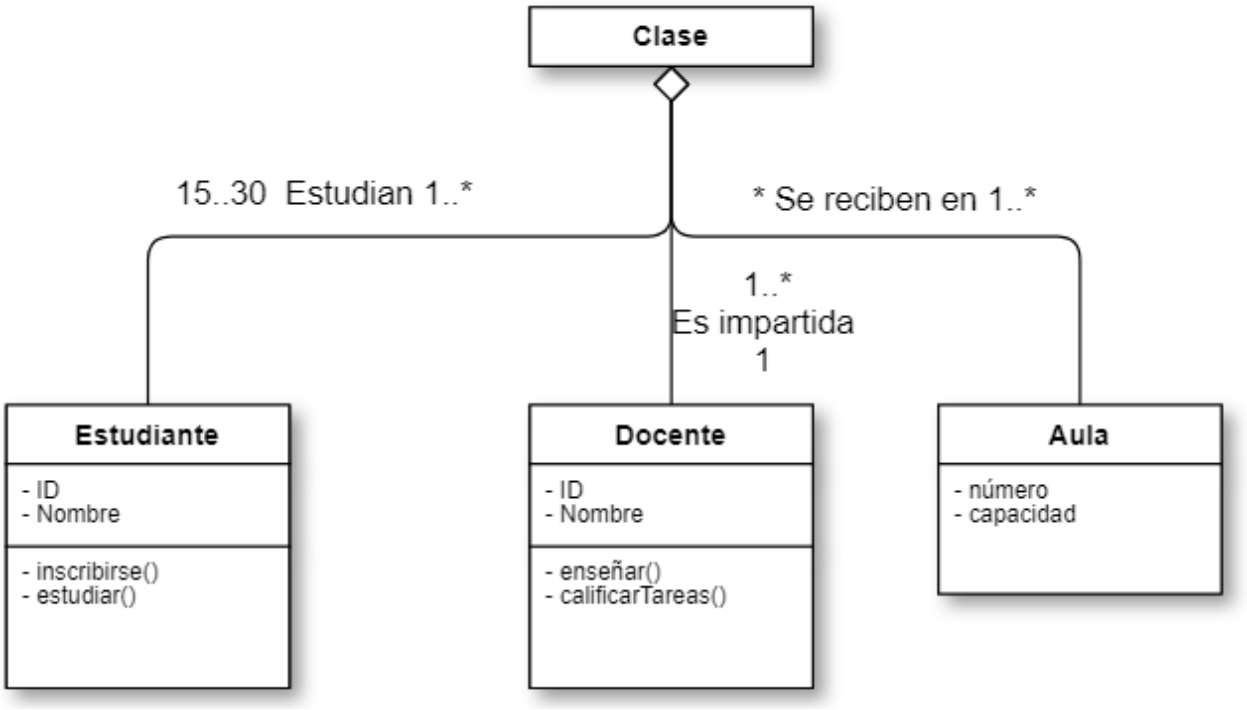


La característica más importante de la composición es que los componentes que la conforman no pueden trabajar fuera de la composición, es decir si por alguna razón la composición fuera destruida o fuera invalidada, los componentes también serán destruidos o invalidados. Veamos el ejemplo: Una empresa es una composición de sus empleados, sus activos, y sus cuentas o inversiones. Si la empresa fracasara y fuera cerrada, los empleados ya no son empleados.

Agregación

Una agregación es un tipo de relación de conjunto, similar a la composición. La diferencia radica en que en una agregación, los componentes que forman el conjunto, no son totalmente dependientes del conjunto en si.

Veamos un ejemplo

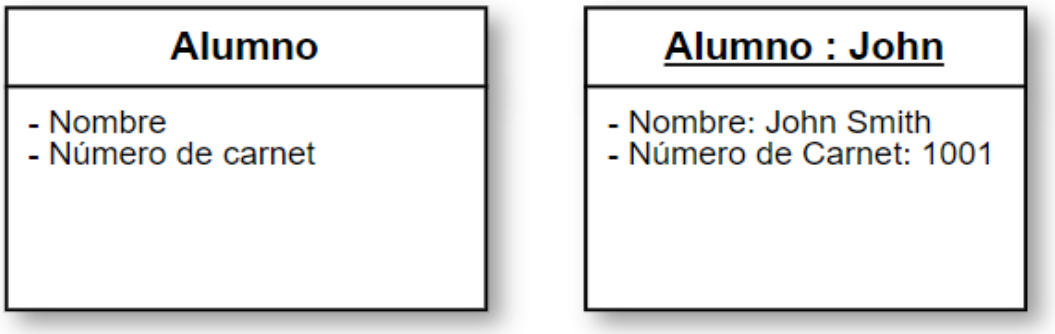


En una escuela cualquiera hay varias clases con muchos alumnos, docentes y aulas en las que se reciben. Una vez termina una clase, los alumnos aun tienen que recibir otras clases, los docentes tienen que preparar su siguiente lección, y el aula queda vacía para el siguiente periodo. Es decir, una vez terminada la clase, los componentes de la agregación no pierden su estado, siguen siendo clases funcionales.

A las agregaciones también se les conoce como composición débil.

Diagrama de Objetos

Un diagrama de objetos es un diagrama vinculado a un diagrama de Clases, un Objeto, en este contexto, es una instancia de una clase, y el diagrama representa un diagrama de clases en un momento concreto



A la izquierda, una clase, a la derecha un objeto (una instancia de una clase).

El nombre de un objeto se escribe como “clase : objeto” separado por dos puntos (:)

En un diagrama de objetos todos los atributos cuentan con un valor

Multiplicidad en un diagrama de Objetos

La multiplicidad no se coloca en un diagrama de objetos, en su lugar se instancia cada clase y se forman las relaciones

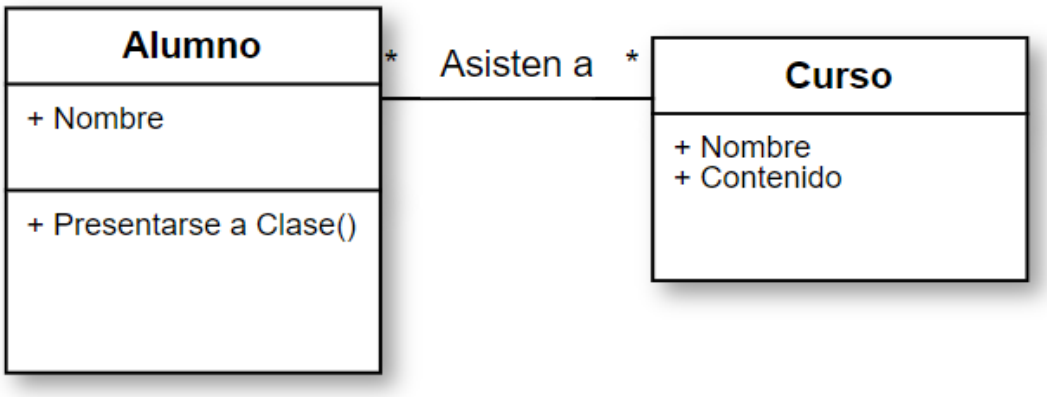
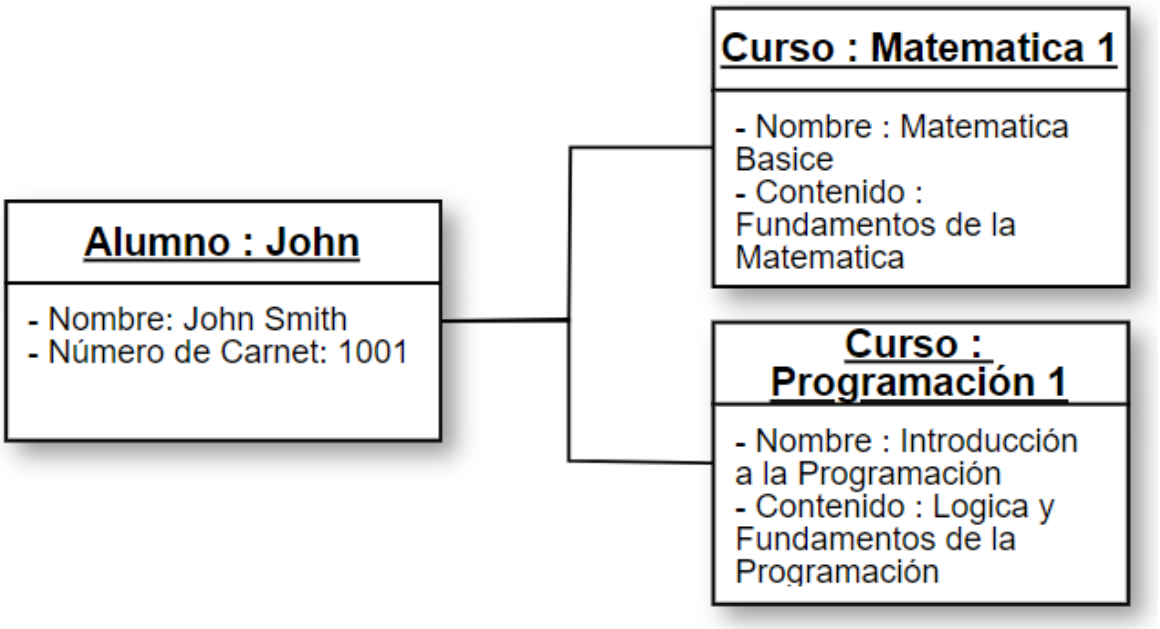


Diagrama de Clases



Diagramas de Casos de Uso

Un diagrama de caso de uso es una representación de cómo puede interactuar un usuario con un sistema

Contiene varias partes de interés

Actor

Es una entidad que interactúa con el sistema, puede ser una persona, un computador u otro sistema.

Caso de Uso

Es una forma en la que interactúa un actor con el sistema, es representado por un óvalo y contiene el nombre de la acción que se ejecuta.

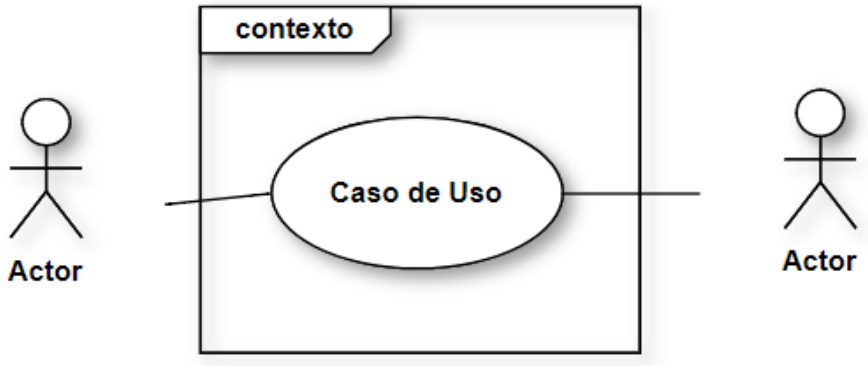
Más de un actor puede interactuar con el mismo caso de uso.

Contexto

Es la representación del sistema con el que se está tratando, es un rectángulo, tiene el nombre del sistema en la parte superior, y contiene los casos de uso en su interior, los actores están en el exterior del rectángulo

Asociaciones

Son representadas como líneas que unen al actor con el caso de uso



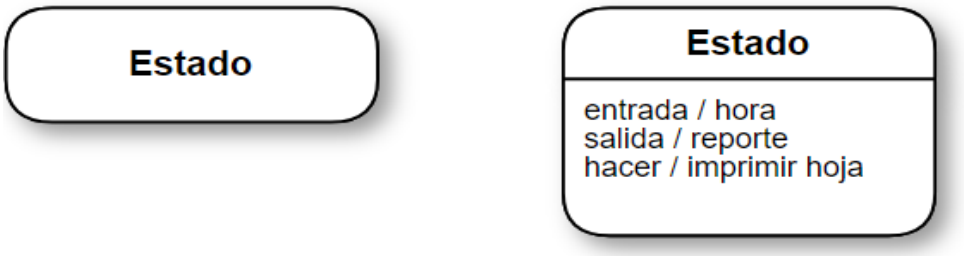
Ej. Dos actores interactúan con un caso de uso

Diagramas de Estado

Representa el conjunto de diferentes estados por los cuales pasa un caso de uso o un objeto a lo largo de su vida.

Estado

Representa situaciones durante la vida de un objeto

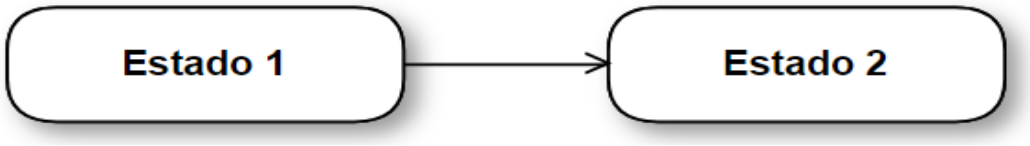


Un estado puede ser simple, o puede tener condiciones de entrada, salida y acciones que debe de realizar, se indican como

- Entrada / *variable de entrada*
- Salida / *variable de salida*
- Hacer / *acción a realizar*

Transición

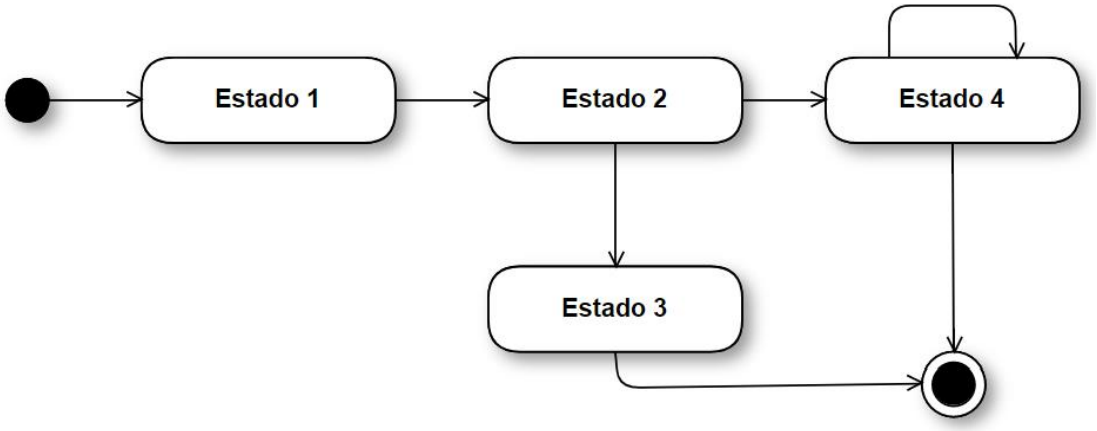
Una flecha representa el pasaje entre diferentes estados de un objeto. Se etiqueta con el evento que lo provoca y con la acción resultante.



Estado Inicial y Estado Final

El diagrama de estados puede tener un estado inicial, también llamado un estado de creación, es en el que un objeto está cuando se creó por primera vez, mientras que un estado final es aquel en que no hay transiciones llevan fuera de él.

A la izquierda la representación de un estado inicial, y a la derecha la de un estado final



Un estado puede regresar a sí mismo o avanzar hacia otro estado

Diagrama de Secuencias

Se usa para modelar el flujo de la lógica dentro de su sistema de una manera visual, como los objetos interactúan entre sí al transcurrir el tiempo. Estos pueden ser usados para modelar:

- Escenarios de uso
- Lógica de métodos, de alguna operación compleja.
- Lógica de servicios, que puede ser invocada por una gran variedad de clientes.

Línea de Vida

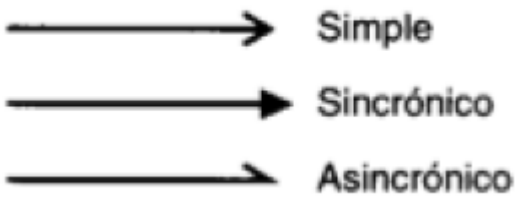
Es la línea punteada que se coloca debajo de un objeto, representa el tiempo transcurrido en un diagrama de secuencias

Tiempo de Activación

Es un rectángulo que se coloca sobre la línea de vida de un objeto, representa el tiempo de ejecución de una acción determinada.

Mensajes

Flechas que van de un objeto a otro, dependiendo del tipo de flecha depende el tipo de mensaje



- Simple :Se transfiere el control de un objeto a otro.
- Sincrónico: El objeto queda a la espera de la respuesta a un mensaje antes de continuar con su trabajo.
- Asincrónico: Un objeto no espera una respuesta al mensaje para continuar.

En ocasiones el mensaje simple y el mensaje asincrónico se usan de la misma manera

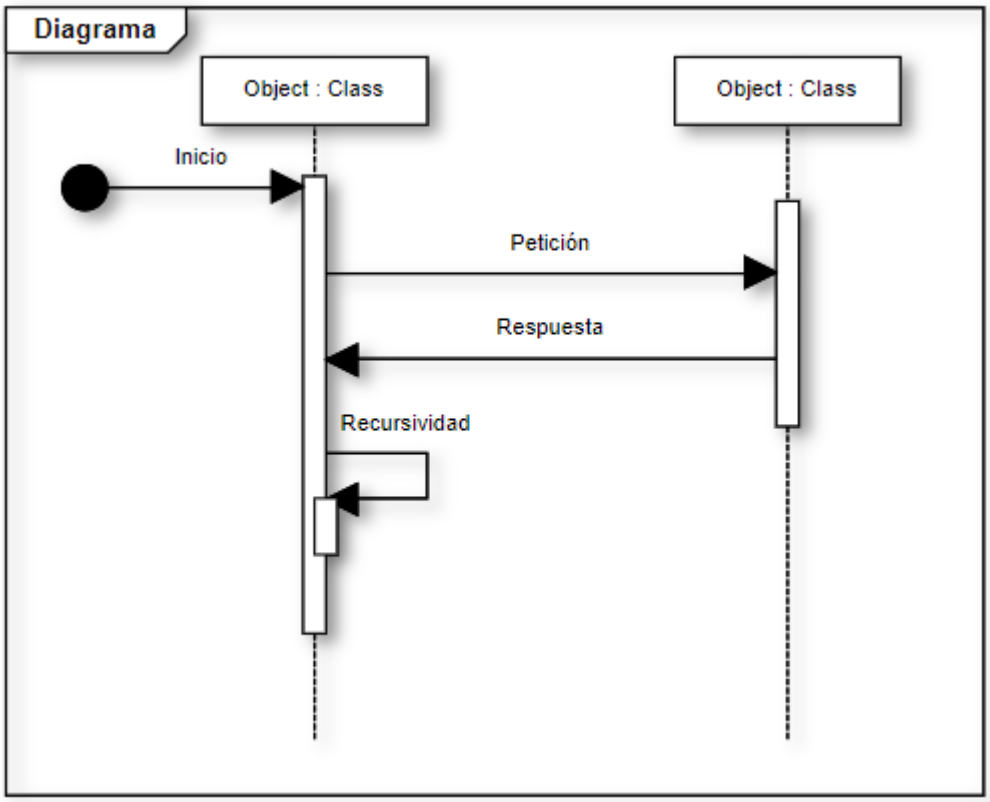


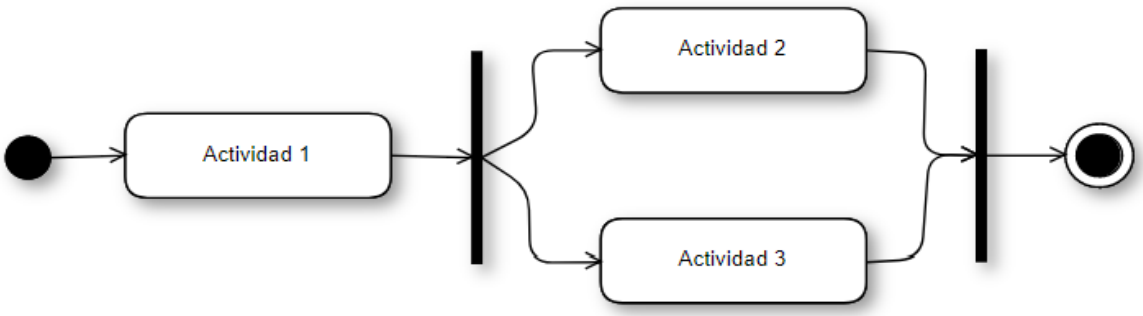
Diagrama de Actividades

El diagrama de actividades es una extensión del diagrama de estados, mientras el diagrama de estados representa los diferentes estados en los que se puede encontrar un sistema, el diagrama de actividades representa las distintas acciones que pueden causar el cambio de un estado a otro, usa muchos de los mismos elementos del diagrama de estados

Algunos casos en los que se usa un diagrama de actividades

- Demostrar la lógica de un algoritmo.
- Describir los pasos realizados en un caso de uso UML.
- Ilustrar un proceso de negocios o flujo de trabajo entre los usuarios y el sistema.

- Simplificar y mejorar cualquier proceso clarificando casos de uso complicados.
- Modelar elementos de arquitectura de software, tales como método, función y operación.



Ejemplo de diagrama de Actividades

Lo que no hemos visto del diagrama de estados es

Clonación y Unión

Barras laterales, las cuales se usan para indicar el inicio y el final de acciones concurrentes, o que se ejecutan al mismo tiempo, luego esas actividades se unen nuevamente para continuar la ejecución.

Sin embargo, no siempre las actividades se van a reunir nuevamente.

En el caso de que se encuentre una situación de decisión se utiliza un rombo

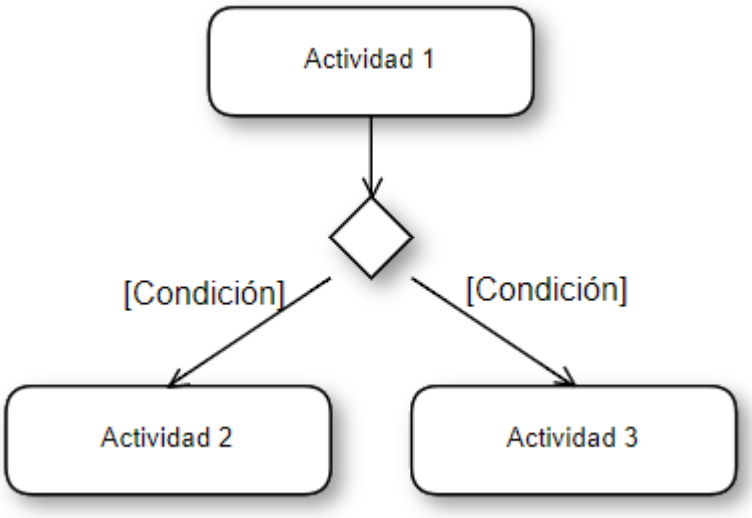


Diagrama de Colaboraciones

El diagrama de colaboraciones también llamado diagrama de comunicación, describe las interacciones entre los objetos en términos de mensajes secuenciados. Los diagramas de colaboración representan una combinación de información tomada de los diagramas de clases, de secuencias y de casos de uso, describiendo el comportamiento, tanto de la estructura estática, como de la estructura dinámica de un sistema.

Es una extensión del diagrama de objetos, mientras que el diagrama de objetos muestra las relaciones entre los objetos, el diagrama de colaboración nos muestra también los mensajes y el orden de los mismos.

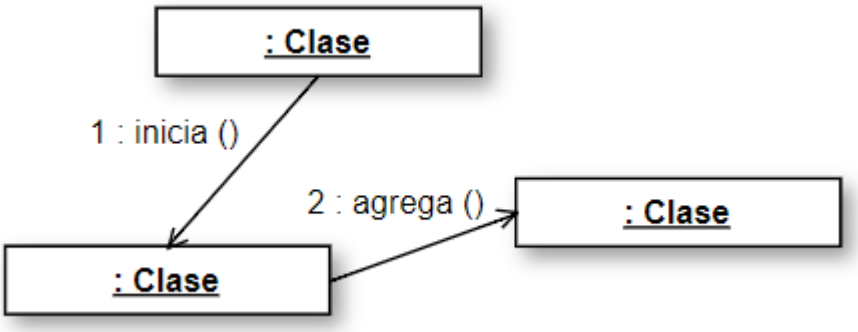


Diagrama de Componentes

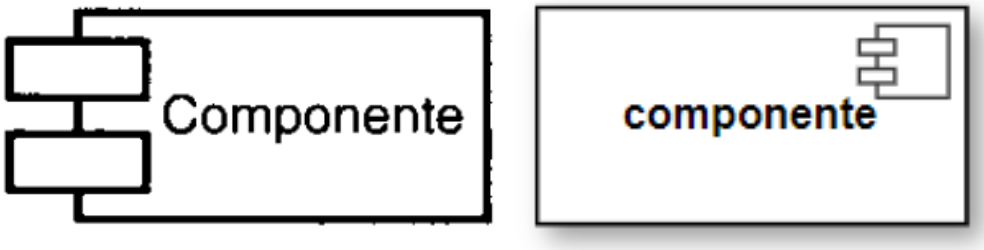
Un diagrama de componentes describe la organización de los componentes físicos de un sistema y las interfaces con los componentes

El diagrama está formado por:

Componentes

Bloque de construcción físico del sistema (una base de datos, un servidor, etc.) estos componentes pueden proveer y requerir interfaces.

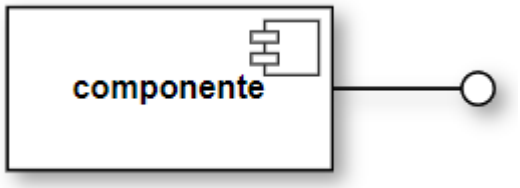
Un componente puede representarse de dos formas



Interfaces

Es una colección de uno o más métodos y cero o más atributos y define un conjunto de comportamientos.

Un componente que provee una interfaz se representa de la siguiente manera



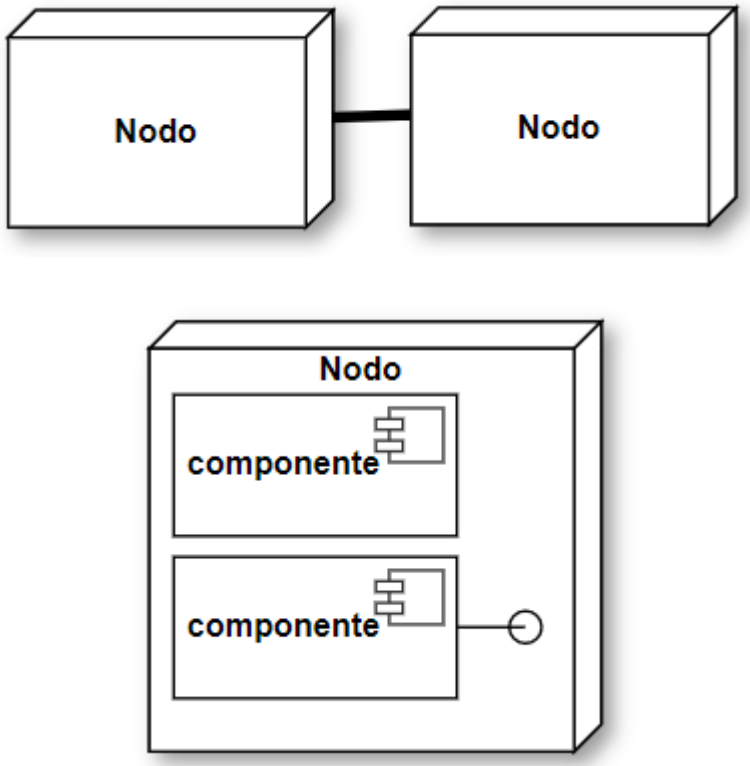
Y cuando un componente requiere una interfaz se representa de la siguiente forma

Diagrama de Distribución

Un diagrama de distribución muestra la arquitectura física de un sistema. Puede representar el hardware de su sistema, el software que se instala en ese hardware y el software intermedio utilizado para conectar las máquinas entre sí.

Nodos

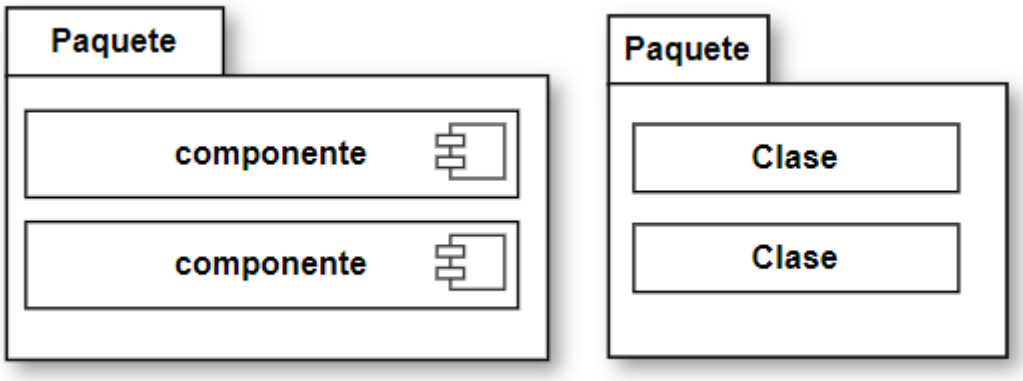
Son los elementos del diagrama que representa un dispositivo físico o de software. Los nodos pueden contener otros nodos o artefactos de software. Los nodos están representados por un cubo y las interacciones entre los nodos están representadas por líneas que conectan los cubos entre sí.



3. Otras características

Paquetes

Los paquetes son elementos de UML que permite organizar los elementos del modelo en grupos, haciendo los diagramas UML más simple y fácil de entender. Los paquetes se representan como carpetas de archivos y se puede utilizar en cualquiera de los diagramas UML, aunque son más comunes en los diagramas de casos de uso y diagramas de clases debido a que estos modelos tienen una tendencia a crecer.



Notas

Las notas nos proveen un mecanismo para aclarar o explicar con mayor profundidad algún elemento de un diagrama UML. Las notas tienen una esquina doblada y se adjunta al elemento conectándolo mediante una línea punteada.