Técnico en
DESARROLLO DE SOFTWARE

Introducción a UML

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 es 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 experto 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 varias 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.

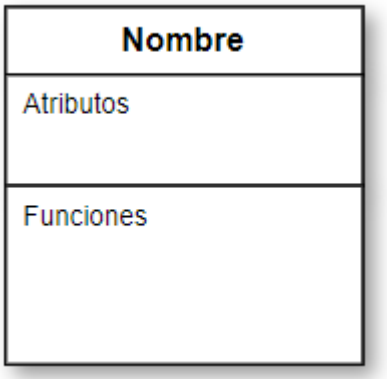
Para poder crear un sistema con un nivel de complejidad considerable la clave es organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas comprendan y convengan con él. UML proporciona las herramientas para organizar un diseño y generar un modelo sólido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

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

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

Los comportamientos llevan al final un par de paréntesis

Ej. *escribir()*

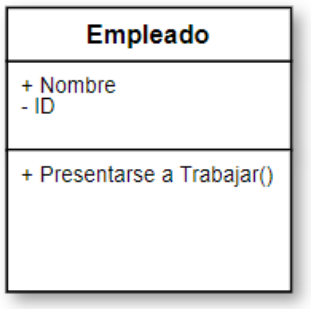
Los comportamientos también pueden recibir parámetros, o valores con los que realizan sus acciones. Estos se colocan dentro de los paréntesis separados por coma, por ejemplo si decimos llamarPorTelefono() nos queda la duda de a que teléfono hay que llamar. Si en cambio decimos llamarPorTelefono(numero), sabemos que vamos a llamar por teléfono al número que está indicado como parámetro. Si tuviéramos que llamar a varios números entonces tenemos que indicar todos los números como parámetros por medio de una lista separada por comas así: llamarPorTelefono(numero1, numero2, etc)

Visibilidad de los Atributos y Comportamientos 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, veremos que significa que una clase herede de otra más adelante

La misma simbología se usa para describir a los comportamientos



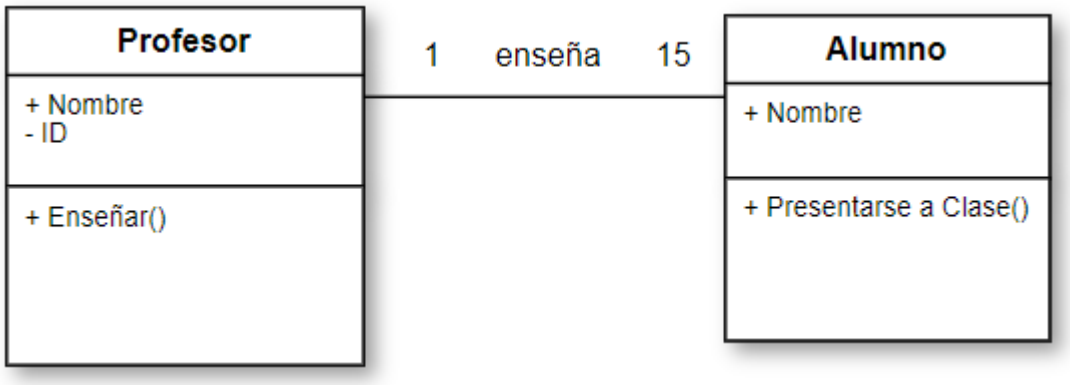
Una clase con un atributo y un comportamiento público, además de un atributo privado

Relaciones entre clases

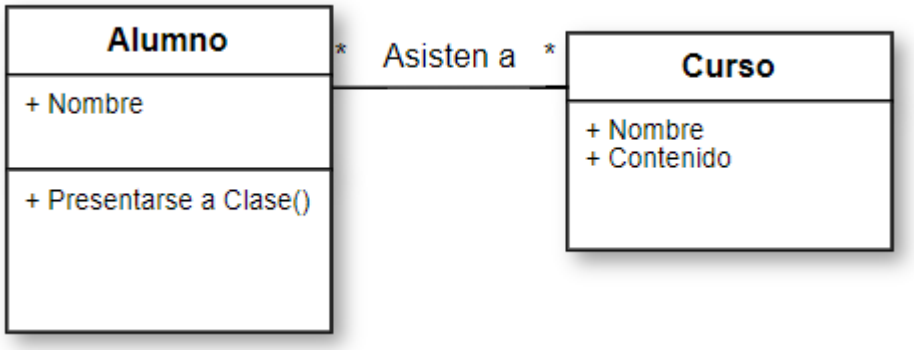
A veces las clases que creamos están relacionadas con otras clases, esto lo representamos por medio de distintos tipos de líneas

Multiplicidad

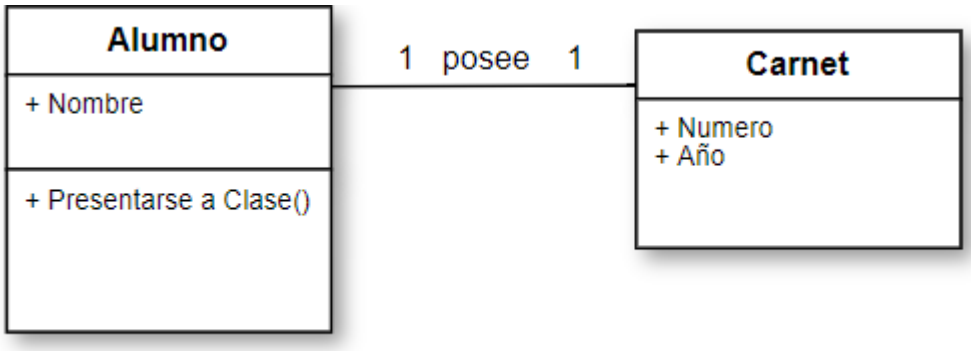
La multiplicidad en una relación nos indica cuantas instancias de la clase estarán presentes, es decir dado una relación entre dos clases ¿Cuántas clases de una y de la otra pueden estar presentes en un momento especifico?. Existen varios tipos de multiplicidad.



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, estableciendo tanto un límite superior como un limite inferior en la 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 soló un carnet, de otro modo podría ocasionar problemas, el segundo carnet podría ser usado por personas no autorizadas, también se podría entender como que un alumno puede tener varios carnets distintos, este puede ser el caso si por ejemplo, el alumno está llevando cursos en dos facultades distintas, por ejemplo, en la facultad de ciencias de la computación y en la facultad de leyes. Este caso una relación de 1 alumno a 1..* carnet sería valida. 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.

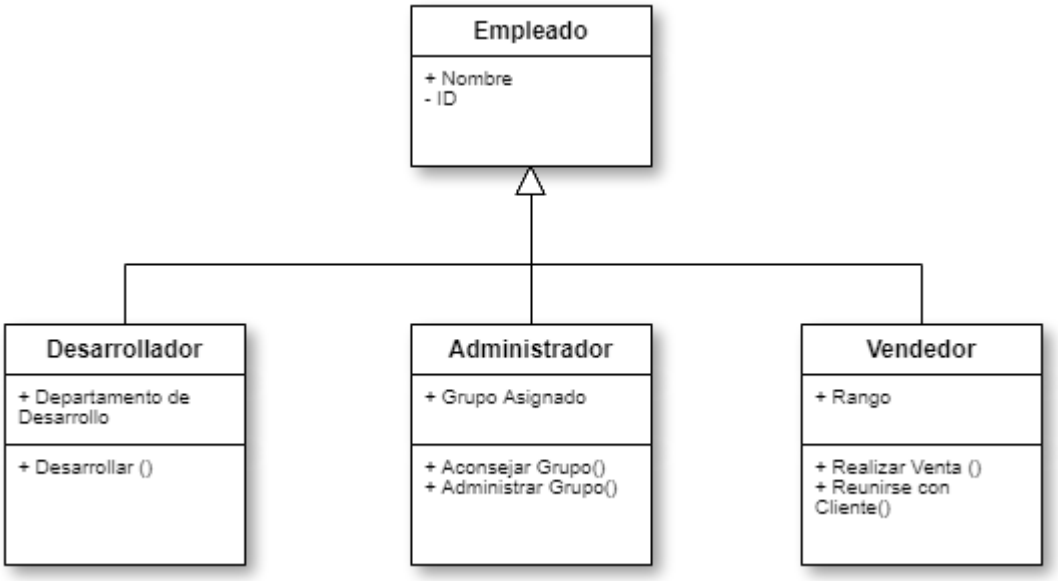
Generalización

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 está 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, especialización es el proceso opuesto, vemos que tenemos una clase, pero queremos que tenga varios comportamientos distintos. Entonces la separamos en varias clases, que aun comparten

algunos atributos básicos, pero cada uno tiene métodos y atributos únicos, en este caso se creo de una base a varias especializaciones.

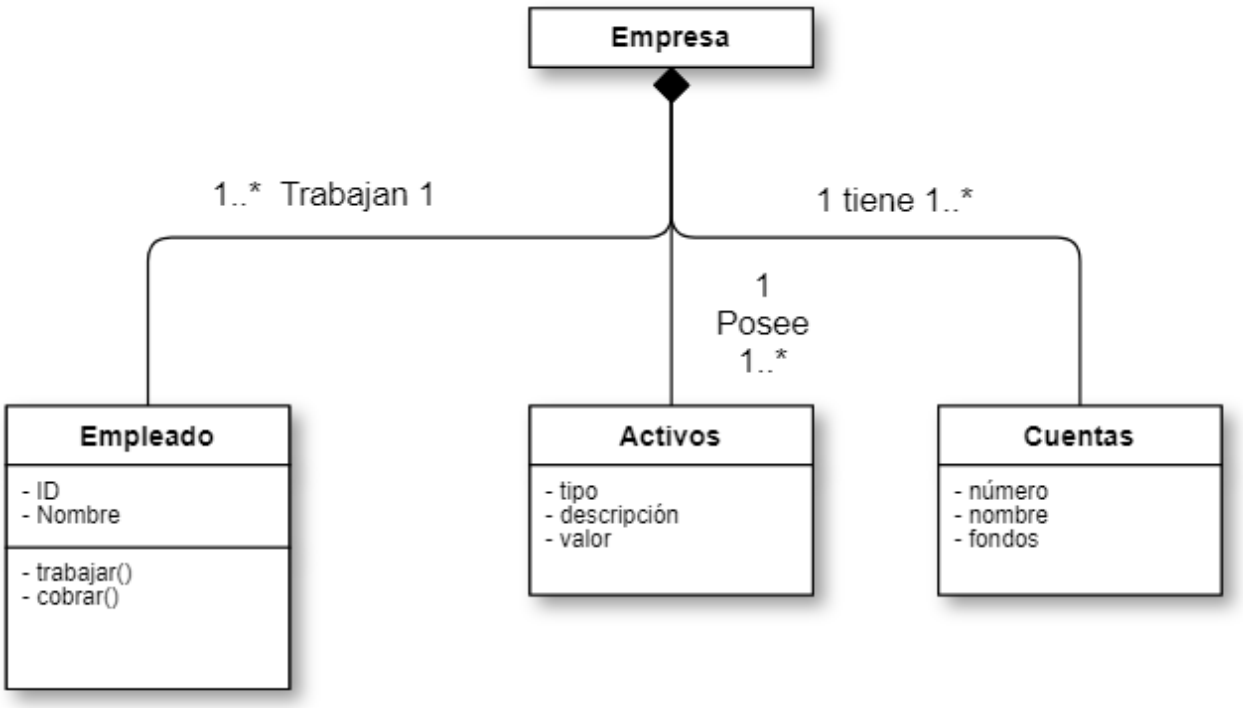
Ambos procesos terminan con el mismo resultado por lo tanto en el diagrama final ambos términos son intercambiables.

Composición

Una composición surge cuando tenemos varias clases, que nos interesa trabajen juntas, estas clases formaran un conjunto en el que trabajaran. 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.

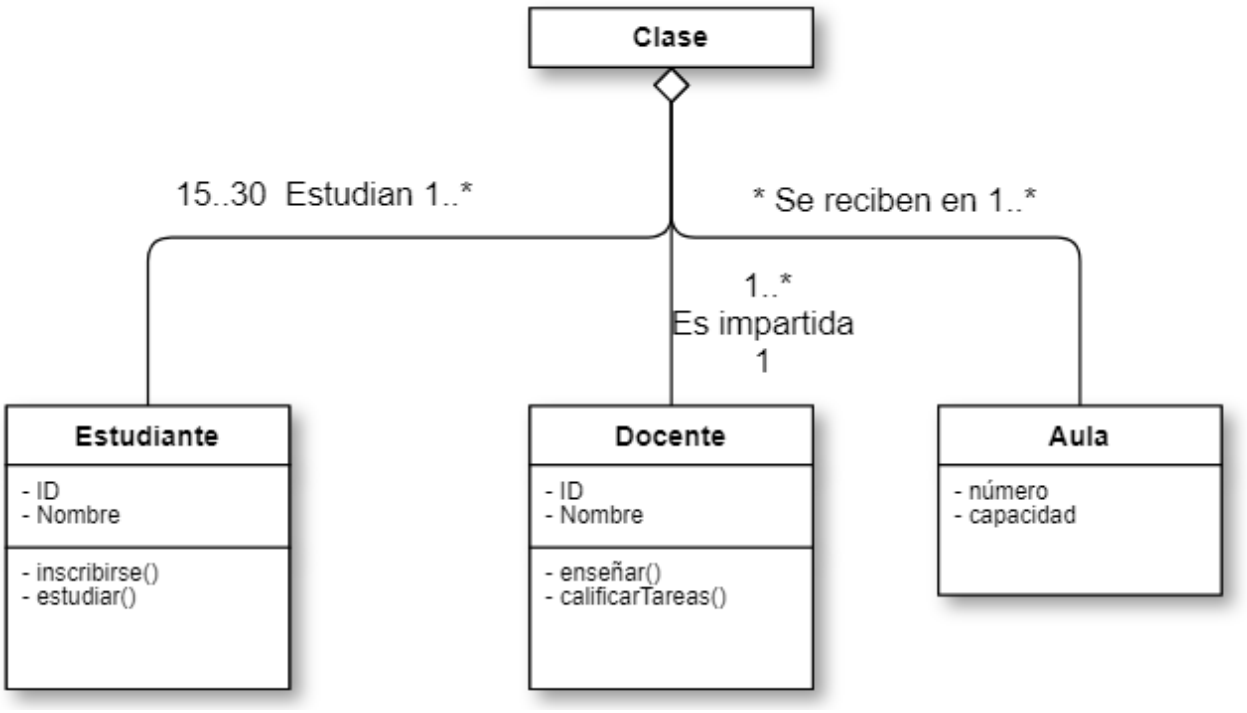


La característica más importante de la composición es que los componentes que la conforman no pueden trabajar fuera de la composición, es decir si por alguna razón la composición fuera destruida o fuera invalidada, los componentes también serán destruidos o invalidados. Veamos el ejemplo, Una empresa 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.

Ejemplo

Vamos a trabajar los distintos diagramas con un ejemplo:

Estamos en el proceso de diseño de un sistema de bases de datos para una clínica, lo que tenemos que hacer es diseñar el sistema de tal forma que nos permita relacionar a: pacientes, doctores y las citas, esto tendrá el fin de tener todos los recursos y personas organizados de tal forma que no haya conflictos de horario.

Tenemos las siguientes características a considerar

- Un paciente es una persona que tiene una aflicción, puede ser una molestia, una herida o algún síntoma.
- Un Doctor es una persona que es capaz de tratar dichas aflicciones, sin embargo, existen muchos tipos de doctores especializados, si bien todos son capaces de realizar practicas de medicina general, hay doctores que se especializan en problemas del corazón, otros que se especializan en tratar con niños y otros que son cirujanos.
- Una cita es un compromiso entre un paciente y un doctor, con una hora y un tema determinado. Si estamos tratando con un cirujano, una cita también puede ser una cirugía.

Empecemos definiendo estos tres como nuestras clases, a cada uno vamos a definir los distintos atributos y comportamientos que podemos realizar.

Empecemos con el paciente:

Atributos

- Nombre
- Teléfono
- Id

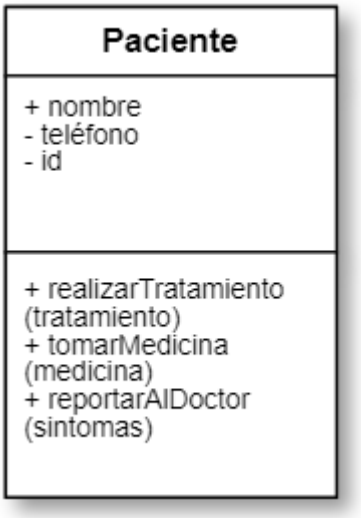
Comportamientos

- reportarAlDoctor (síntomas)

- realizarTratamiento (tratamiento)
- tomarMedicina (medicina)

Los comportamientos tratan de lo que un paciente puede hacer, lo primero que puede hacer es informar a un doctor que síntomas tiene, luego puede realizar dos cosas, tomar el medicamento o realizar el tratamiento que le receto el doctor.

La clase se vería así



El segundo será el doctor, al igual que el paciente tendrá varios atributos, pero, como mencionamos antes un doctor puede tener una especialización, lo que le da una distinción ante otros doctores y le permite tener otros comportamientos.

La clase de doctor se puede ver así

Empleado

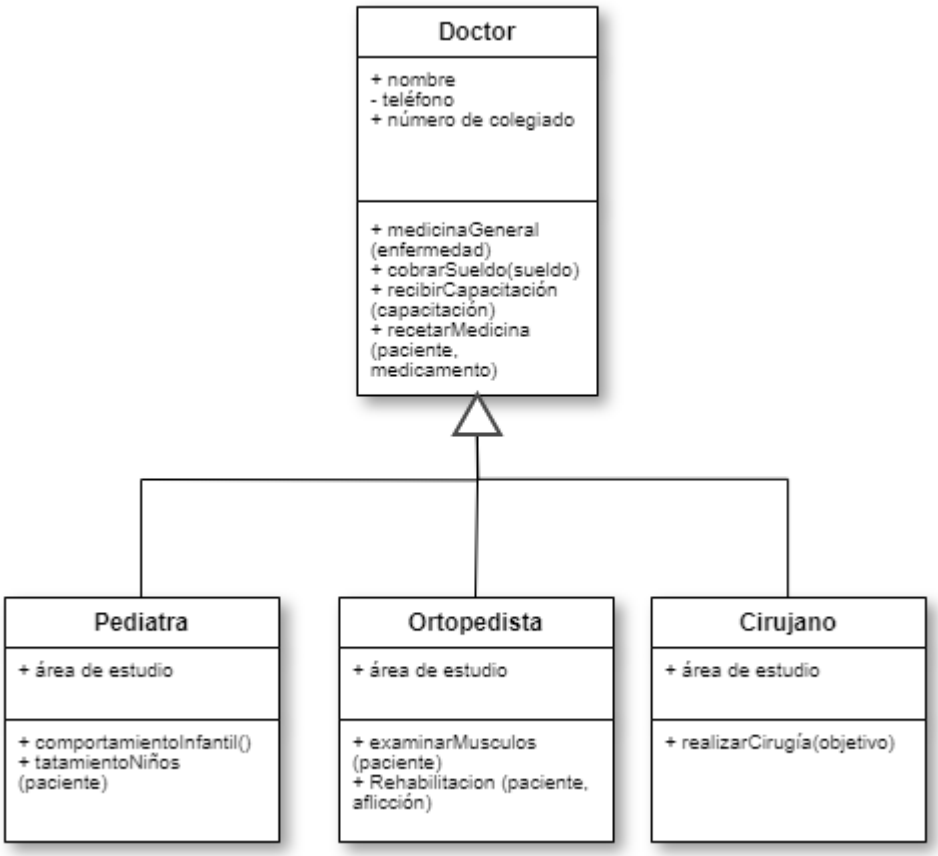
Atributos

- Nombre
- Teléfono
- Número de Colegiado

Comportamientos

- medicinaGeneral (enfermedad)
- cobrarSueldo(sueldo)
- recibirCapacitación (capacitación)
- recetarMedicina (paciente, medicamento)

Luego podemos definir otros tipos de doctor, los cuales estarán relacionados con la clase doctor por medio de la herencia o generalización



En este caso vemos tres tipos de doctor, un pediatra se encarga de tratar con niños, por lo que tiene que conocer las diferencias entre las enfermedades entre adultos y niños, tiene que saber como tratar con los niños, que no se comportan como los adultos y tiene que saber como manejar los comportamientos y realizar tratamientos efectivos para los pequeños. Un ortopedista es un doctor que puede tratar con los músculos y los huesos,

como enfermedades en las piernas o en la columna. Conoce muchos tratamientos que ayudan a manejar estas aflicciones. Finalmente, un cirujano es aquel que realiza cirugía para tratar daños internos a los órganos, músculos o huesos.

Cada uno de estos es un doctor, por lo que puede ejercer medicina general, pero no podemos poner a un pediatra a realizar cirugía. De ahí la necesidad de usar la generalización/especialización.

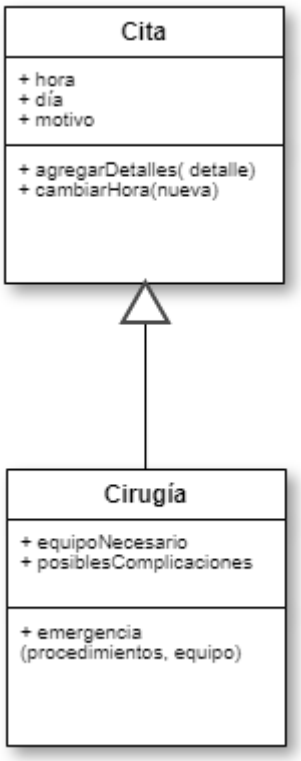
Ahora definamos una cita, es una reunión entre un doctor y un paciente, tiene una hora particular, un día y un motivo.

Cita

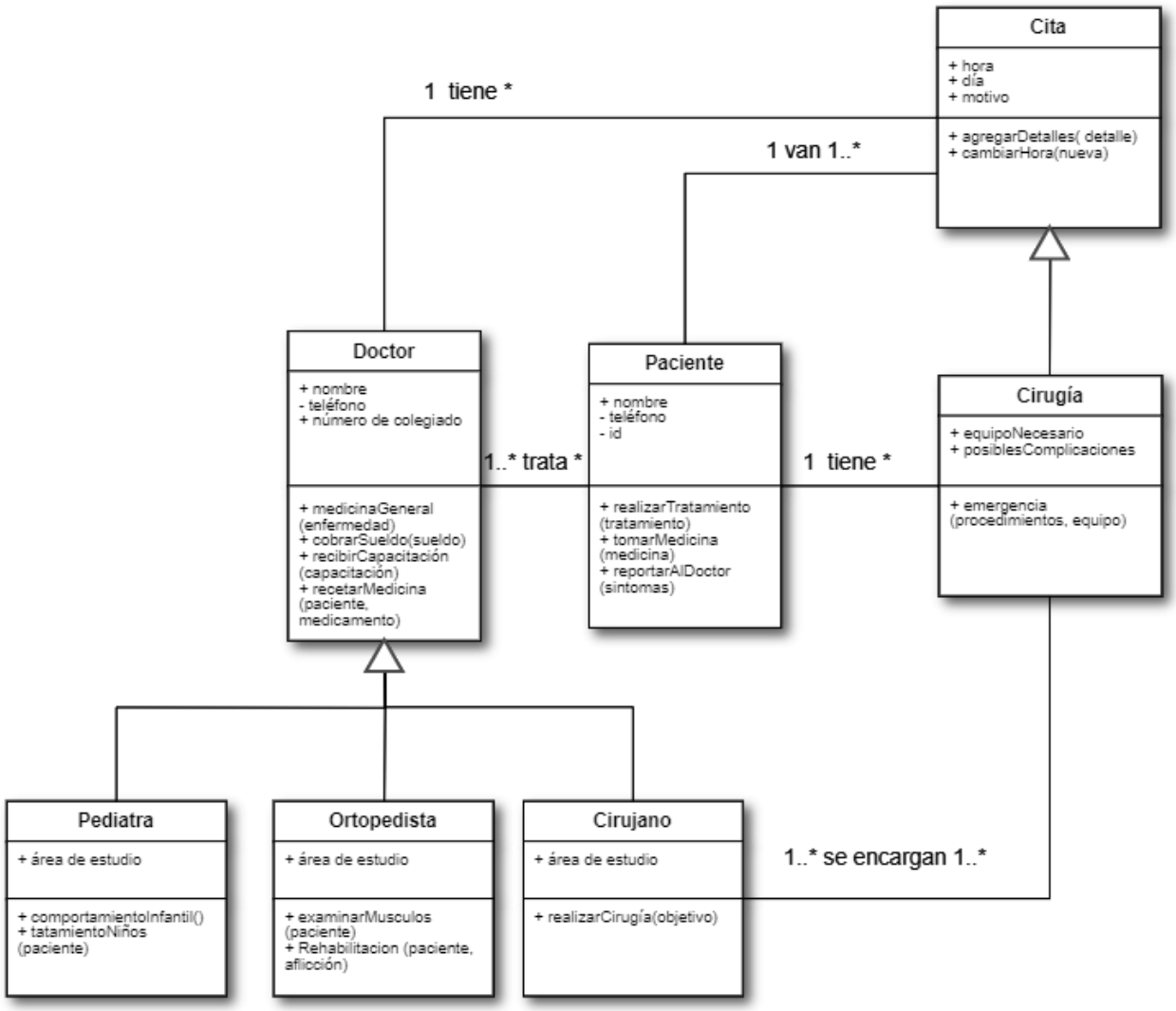
Atributos

- Hora de Inicio
- Día
- Motivo

Ya que hablamos de cirugías, una cirugía puede ser un tipo de cita, las cirugías tratan con el paciente y el doctor, pero son procedimientos más arriesgados. Necesitan preparación, equipo e incluso planes de contingencia.


Cita
+ hora
+ día
+ motivo
+ agregarDetalles(detalle)
+ cambiarHora(nueva)
Cirugía
+ equipoNecesario
+ posiblesComplicaciones
+ emergencia
(procedimientos, equipo)

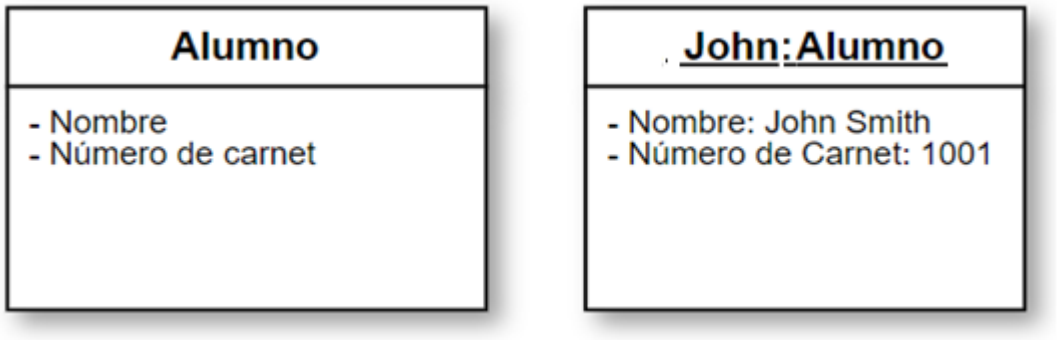
Ya con esto podemos armar nuestro diagrama de clases, es muy importante no olvidar las multiplicidades.



Vemos que un paciente puede estár en comunicación con un doctor, y estos pueden realizar una cita para discutir las aflicciones del paciente. Sin embargo si se trata de una cirugía solamente se puede realizar con un doctor que sea un cirujano.

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 “objeto : clase” 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

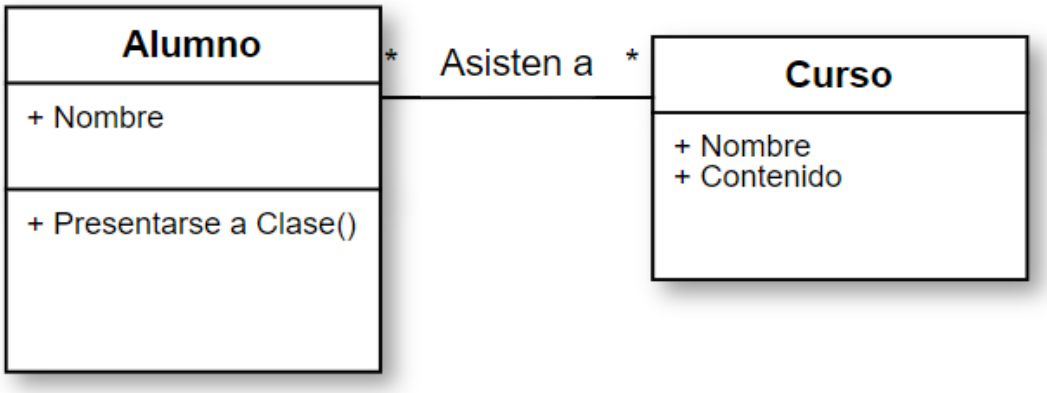
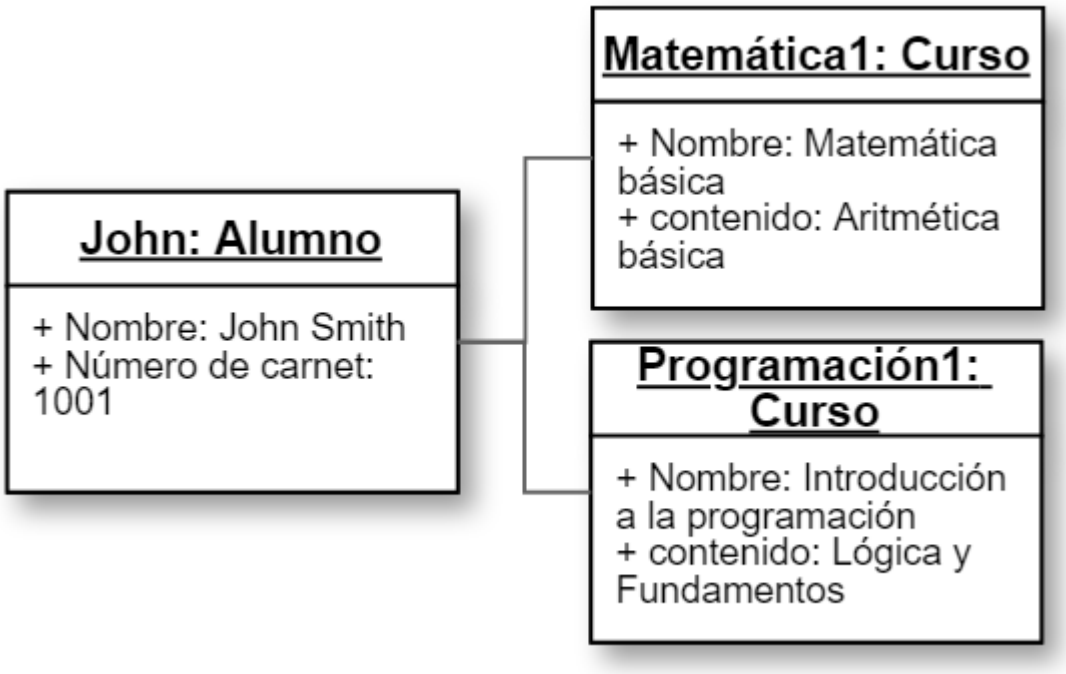
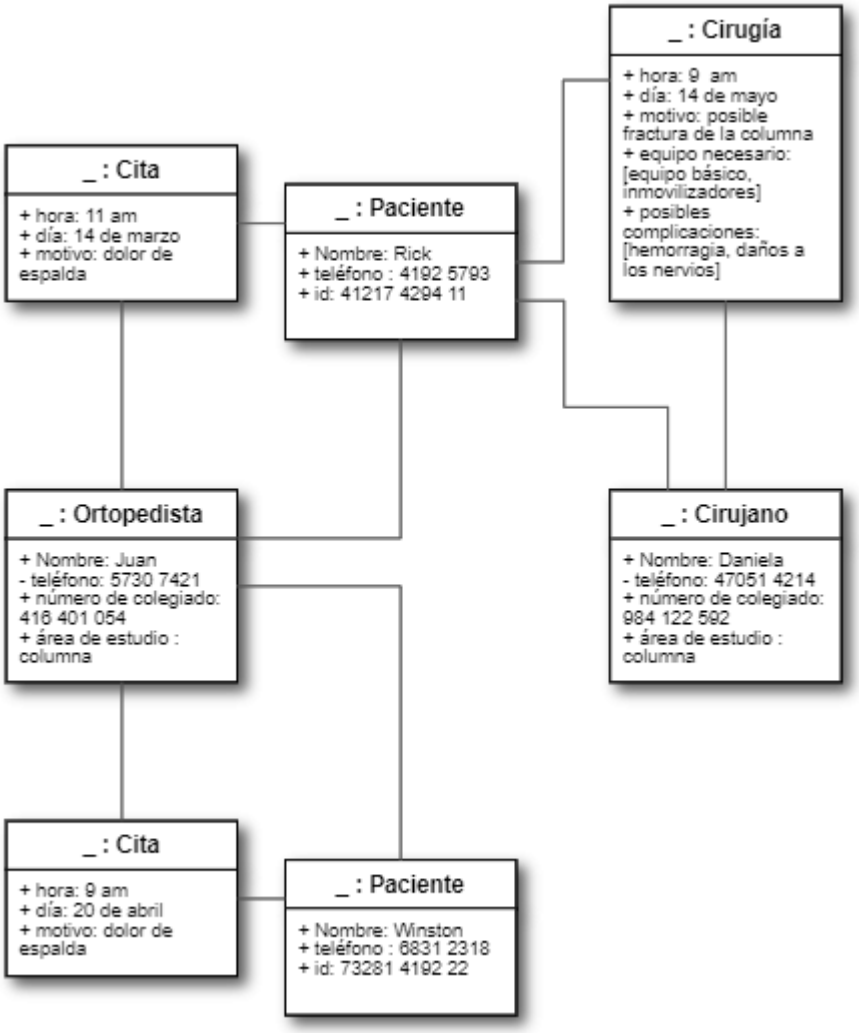


Diagrama de Clases

Hay que cambiar la imagen que ya esta por esta



Veamos entonces como se vería el diagrama de clases del ejemplo de la clínica como un diagrama de objetos. Lo importante del diagrama de objetos es que nos representa una vista del sistema, es decir, para un punto particular nos va a mostrar como se ven las relaciones y los objetos



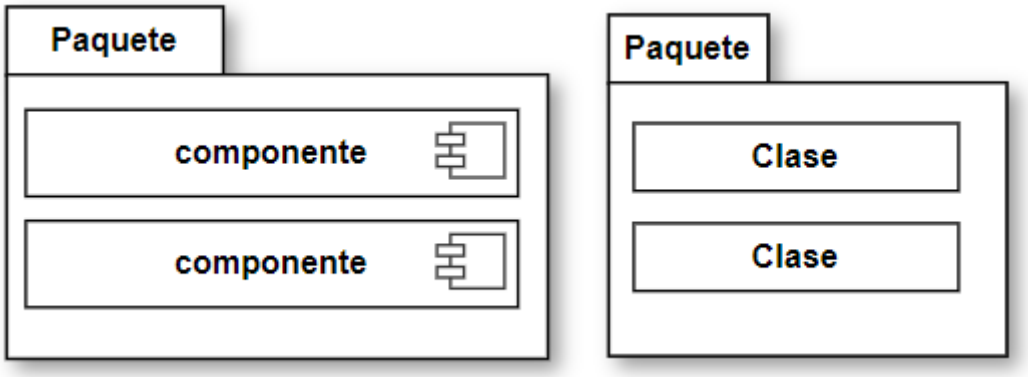
En este caso podemos ver las relaciones de multiplicidad, el mismo doctor puede tratar a diferentes pacientes, el mismo paciente puede tener varias citas y asimismo puede ver a muchos doctores

3. Otras características

Paquetes

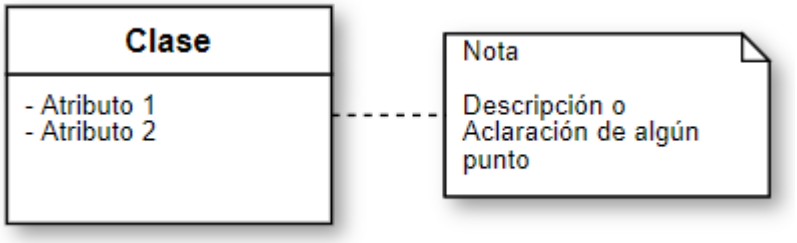
Los paquetes son elementos de UML que permite organizar los elementos del modelo en grupos, haciendo los diagramas UML más simple y fácil de entender. Los paquetes se representan como carpetas de archivos y se puede utilizar en cualquiera de los

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.



Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.