UML

DIAGRAMA DE CLASES

DISEÑO DE APLICACIONES | 2021

Contenido

Diagrama de Clases

Muestra una vista estática de la estructura del sistema (o, de una parte), describiendo qué atributos, comportamiento y relaciones debe poseer el mismo.

Los diagramas de clases son los diagramas más comunes en el modelado de sistemas orientados a objetos. También se utilizan para modelado de datos detallado traduciendo las clases directamente a código.

Un diagrama de clases puede comenzarse a construir en el comienzo del desarrollo de software en la etapa de Requerimientos, estructurar en la etapa de Análisis, refinar en la etapa de Diseño y finalizar al momento de realizar la implementación.

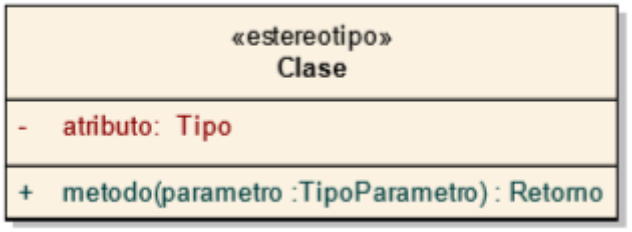
Componentes del Diagrama de Clases

Clase

Descripción de un conjunto de elementos que comparten los mismos atributos, operaciones y relaciones. Se utiliza para capturar el vocabulario del sistema que se está desarrollando. El comportamiento de una clase se describe mediante los mensajes que la clase es capaz de comprender junto con las operaciones que son apropiadas para cada mensaje.

Dentro del Diagrama de Clases, una clase se representa con un rectángulo dividido en tres compartimentos:

- el compartimiento superior contiene el nombre de la clase. Está en negrita y centrado, y la primera letra en mayúscula. Sobre este nombre suele aclararse el *tipo* de clase.
- El compartimento central contiene los atributos de la clase. Están alineados a la izquierda y la primera letra es minúscula. Comienzan con un símbolo que indica su *visibilidad*.
- El compartimiento inferior contiene los *métodos* de la clase. También están alineados a la izquierda y la primera letra es minúscula. También comienzan con un símbolo que indica su visibilidad. Deben colocarse los parámetros que recibe el método separando con dos puntos su tipo. Los atributos deben separarse por comas. Por último, debe colocarse el tipo de retorno del método.



Visibilidad

Normalmente se desea ocultar los detalles de implementación y mostrar sólo las características necesarias para llevar a cabo las responsabilidades de la abstracción.

Hay tres niveles disponibles:

- **Pública:** Cualquier clase externa con visibilidad hacia la clase dada puede utilizar la característica. Se representa con el símbolo "+".
- **Protegida:** Cualquier descendiente de la clase puede utilizar la característica. Se representa con el símbolo "#".
- **Privada:** Sólo la propia clase puede utilizar la característica. Se representa con el símbolo "-"

  *__Paquete:__ Visible para clases del mismo paquete. Se representa con el símbolo "~".

Clases de Análisis

UML, posee como parte del lenguaje el uso de tres tipos de clases, denominadas clases de análisis, cuyo propósito es lograr una estructura más estable y mantenible del sistema que será robusta para el ciclo de vida entero del sistema. Se ha asignado una representación **icónica** para cada uno de los tres tipos de clases de análisis.

Clase de Entidad

La clase de entidad modela información en el sistema que debería guardarse por mucho tiempo y, comúnmente, debería sobrevivir la ejecución del caso de uso del sistema en la que se creó. Todo el comportamiento naturalmente acoplado a esta información debería también ser ubicado en la clase de entidad. Las clases de entidad se utilizan para modelar abstracciones del dominio del problema. Las clases que se identifican en el modelo del dominio del problema son ejemplos de este tipo de clases. Un ejemplo de una clase de entidad es la clase que guarda comportamiento y datos relevantes de una **Película** para el complejo de cines. Las clases de entidad pueden identificarse desde los casos de uso. La mayoría de las clases de entidad se encuentran pronto y son obvias. Estas clases de entidad "obvias" se identifican frecuentemente en el modelo de objetos del dominio del problema. Los otros pueden ser más difíciles de encontrar. Las entidades comúnmente corresponden a algún concepto en la vida real.

Clase de Interfaz

La clase de interfaz modela el comportamiento e información que es dependiente de la frontera del sistema con el ambiente. Así todo lo que concierne a cualquier vínculo entre el sistema y los actores, se ubica en un objeto de interfaz. Un ejemplo de un objeto de interfaz sería la clase que representa la pantalla que vincula un actor con un caso de uso, para pedirle los datos de una película: **PantallaAdmPelicula**. Toda la funcionalidad especificada en las descripciones de los casos de uso, que es directamente dependiente del ambiente del sistema se pone en los objetos de interfaz. Es mediante estos objetos que los actores se comunican con el sistema. La tarea de un objeto de interfaz es trasladar la entrada del actor al sistema en eventos del sistema, y traducir esos eventos del sistema en los que el actor está interesado en algo que será presentado al actor. Las clases de interfaz pueden, en otras palabras, describir la comunicación bidireccional entre el sistema y sus usuarios. Cada actor concreto necesita su propia interfaz para comunicarse con el sistema. Así, para identificar qué parte del flujo de un caso de uso debería destinarse a una clase de interfaz, enfocamos las interacciones entre los actores y los casos de uso. Esto significa que deberíamos buscar unidades que muestran una o más de las características siguientes:

- Presentan información al actor o piden información de él.
- Su funcionalidad cambia conforme cambia el comportamiento del actor.
- Su curso es dependiente de un tipo particular de interfaz.

Clase de Control

La clase de control modela funcionalidad que implica operar sobre varios objetos diferentes de entidad, haciendo algunos cálculos y retornando el resultado al objeto de interfaz. Contiene comportamiento de la lógica de negocio definida en un caso de uso. Tiene responsabilidades de coordinación de la ejecución de un caso de uso y funciona como intermediario entre las clases de interfaz y las de entidad. Un ejemplo de una clase de control es el **GestorPelicula**, utilizada para coordinar la realización del caso de uso *registrar película*. La clase de control actúa como vínculo, que une los otros tipos de clase. Son comúnmente las más efímeras de todos los tipos de objeto y comúnmente duran mientras dura la ejecución de un caso de uso. Es, sin embargo, difícil lograr un equilibrio razonable entre qué se pone en los objetos de entidad, en los objetos de control y en los objetos de interfaz. Daremos aquí algunas heurísticas con respecto a cómo encontrarlos y especificarlos. Los objetos de control se encuentran directamente desde los casos de uso. En una primera vista asignaremos un objeto de control para cada caso de uso. Los tipos típicos de funcionalidad ubicados en las clases de control son comportamiento relacionado a la transacción, o secuencias específicas de control de uno o varios casos de uso, o la funcionalidad que separa los objetos de entidad de los objetos de interfaz. Los objetos de control conectan cursos de eventos y así llevan adelante la comunicación con otros objetos. Estos dos últimos tipos de clases de análisis suelen llamarse de fabricación pura, y eso se debe a que son creadas para modelar el dominio de la solución y no tienen un vínculo directo con el dominio del problema.

¿Por qué modelar utilizando los tres tipos de clases de análisis?

La suposición básica es que todos los sistemas cambiarán. Por lo tanto, la estabilidad ocurrirá en el sentido que todos los cambios sean locales, esto es, que afectan (preferentemente) a una única clase del sistema.

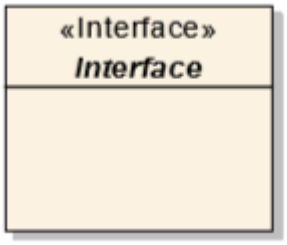
Los cambios más comunes al sistema son a su funcionalidad y a su interfaz. Los cambios a la interfaz deberían afectar únicamente a las clases de interfaz. Los cambios en la funcionalidad son más difíciles. La funcionalidad puede ubicarse sobre todos los tipos de clases, por eso ¿cómo logramos la localización de estos cambios? Si es la funcionalidad que está asociada a la información retenida por el sistema, por ejemplo, cómo calcular la edad de una persona, tales cambios afectan a las clases de entidad que representa esa información. Los cambios de funcionalidad de una interfaz, por ejemplo, cómo recibir y recolectar

información acerca de una persona, deberían afectar a la clase de interfaz correspondiente. Los cambios de funcionalidad entre clases, por ejemplo, cómo calcular impuestos con varios criterios diferentes, deberían ser locales a la clase de control. La funcionalidad atribuible a uno o varios casos de uso se asigna a una clase de control. En síntesis, asignamos el comportamiento descrito en un caso de uso, según los siguientes principios:

- Aquellas funcionalidades del caso de uso, que son directamente dependientes del ambiente del sistema se ponen en las clases de interfaz.
- Aquellas funcionalidades que tratan con el almacenamiento y manipulación de información que no está naturalmente ubicada en ninguna clase de interfaz y que representa al dominio del problema, se ubica en las clases de entidad.
- Las funcionalidades específicas a uno o varios casos de uso y que cumplen roles de coordinación de un caso de uso, son modeladas con clases de control.

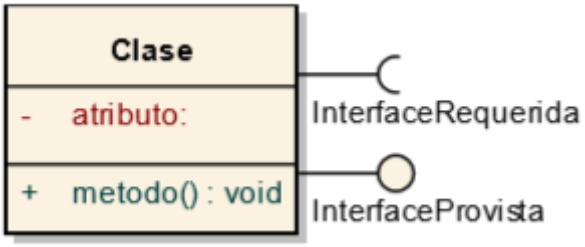
Interfaz

Una interfaz es un tipo especial de clase que agrupa una colección de operaciones que especifican un servicio de una clase o componente.



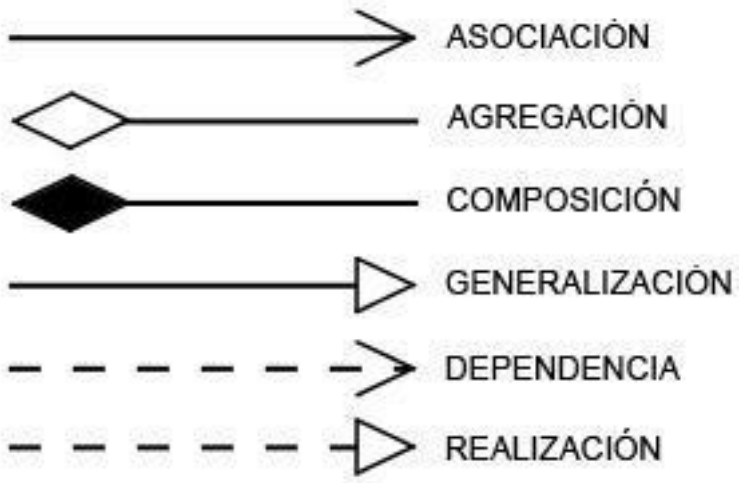
Existen dos tipos de interfaces:

- **Interfaces Requeridas:** Muestran lo que el clasificar al que pertenecen puede requerir del ambiente a través de un puerto dado.
- **Interfaces Provistas:** Muestra la funcionalidad que expone un clasificador a su ambiente.


Clase
- atributo:
+ metodo() : void
InterfaceRequerida
InterfaceProvista

Relaciones

Una relación es una conexión entre dos elementos. En el contexto del diagrama de clases, los elementos que se relacionan son clases e interfaces. Los tipos de relaciones son:



Asociación

Relación estructural que especifica que los objetos de un elemento se conectan a los objetos de otro elemento. Este tipo de relación se suele implementar con una referencia en uno de los dos elementos que participan de la relación hacia el otro elemento. Estas relaciones pueden incluir multiplicidad, navegabilidad y el nombre del rol que se establece en la relación. Al implementarse esta relación el rol de la asociación suele ser del tipo de uno de los elementos en la relación.

Agregación

Es un tipo especial de asociación, que representa una relación completamente conceptual entre un “Todo” y sus “Partes”.

Composición

Es una variación de la agregación que muestra una relación más fuerte entre el todo y sus partes. Se utiliza cuando existe una relación de contenedor-contenido entre dos elementos. Usualmente una instancia de una parte suele pertenecer sólo a una instancia del todo. Esta relación en general implica que al borrarse el todo se borran todas sus partes.

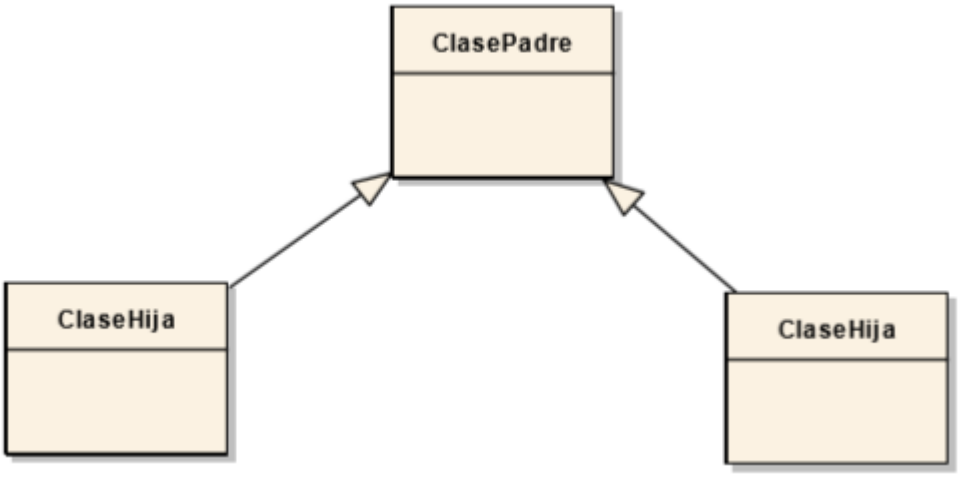
Generalización

Es una relación entre un elemento general (superclase o padre) y un tipo más específico de ese elemento (subclase o hijo). El hijo hereda los métodos y atributos del padre, luego puede añadir nueva estructura y comportamiento, o modificar el comportamiento del padre. Existen dos tipos de herencia:

- Herencia Simple: Un hijo hereda de un único padre.
- Herencia Múltiple.

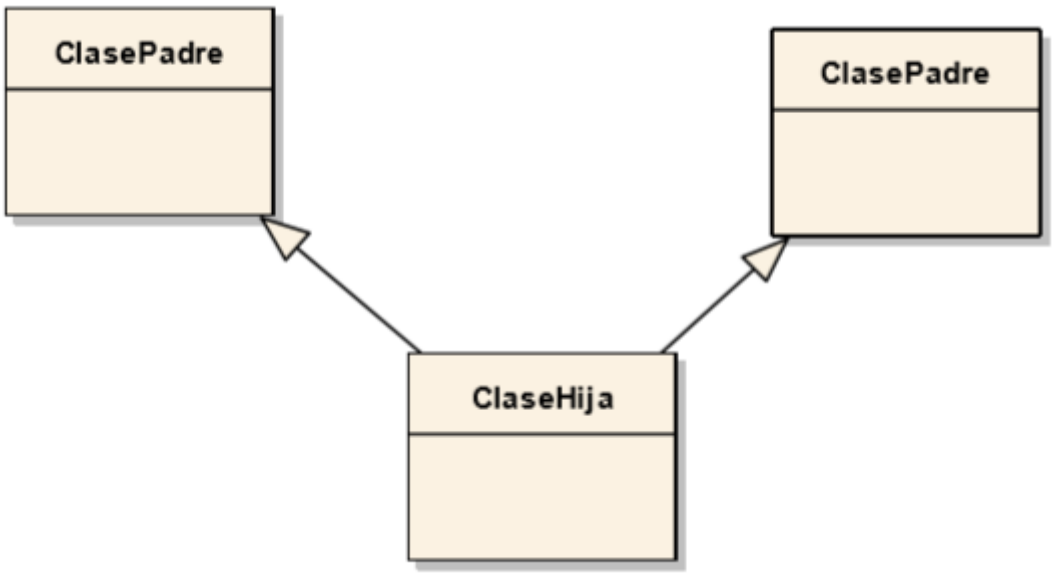
Herencia Simple

Un hijo hereda de un único padre.



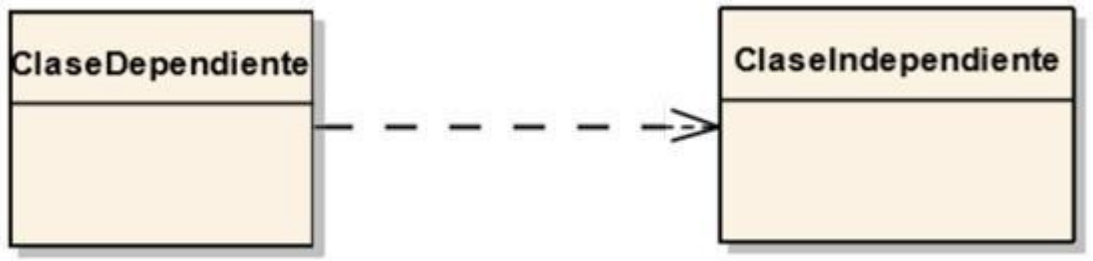
Herencia Múltiple

Un hijo hereda de más de un padre. Este tipo de herencia ha sido prácticamente reemplazada por la realización de clases de interfaz, ya que las buenas prácticas recomiendan no utilizar herencia múltiple y muchos lenguajes de programación orientados a objetos no lo soportan.



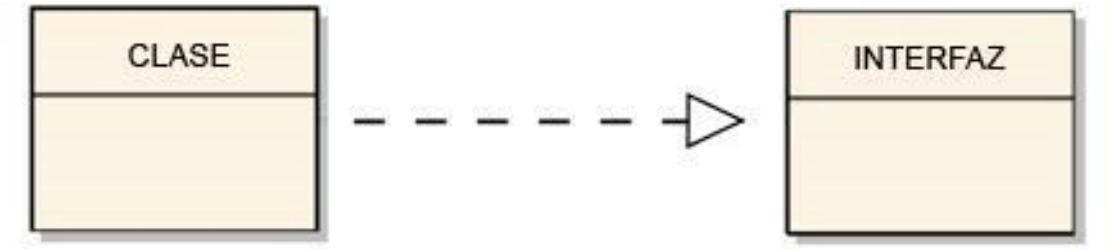
Dependencia

Es una asociación de uso, la cual especifica que un cambio en la especificación de un elemento puede afectar a otro elemento que lo utiliza, pero no necesariamente a la inversa.



Realización

Esta relación implica que el elemento de donde parte la relación implementa el comportamiento definido en el otro elemento relacionado. Esta relación se suele utilizar entre una clase y una interfaz.



Multiplicidad

Indica la cantidad de elementos que participan de la relación. Pueden ser:

0..1	Cero o una instancia
0 .. *	Cero o más instancias
1 .. *	Una o más instancias
N	N instancias

Uso de los Diagramas de Clases

Se utilizan para:

- Modelar el modelo de objetos de dominio del problema o la solución.
- Modelar las clases que luego se mapearán en componentes de código, para la implementación del sistema.

- Mantener trazabilidad de la vista de la estructura a través de los diferentes modelos construidos en el proceso de desarrollo.
- Mostrar estructuras de clases como las siguientes:
    - Las clases más importantes y sus relaciones.
    - Clases relacionadas funcionalmente.
    - Clases que pertenecen al mismo paquete.
    - Jerarquías de agregación importantes.
    - Estructuras importantes de objetos de entidad, incluyendo estructuras de clases con relaciones de asociación, agregación y generalización.
    - Paquetes y sus dependencias.
    - Clases que participan en una realización de caso de uso específica.
    - Una clase, sus atributos, operaciones y relaciones con otras clases.
- Modelar el vocabulario de un sistema: Implica tomar decisiones sobre qué abstracciones son parte del sistema en consideración y cuáles caen fuera de los límites. Se construye el Modelo de Objetos del Dominio.
- Modelar colaboraciones simples: Sociedad de clases, interfaces y otros elementos que colaboran para proporcionar un comportamiento cooperativo mayor que la suma de todos los elementos.
- Modelar la estructura del sistema: Incluye las clases del dominio del problema y las de los diagramas de interacción. Representa una vista estática del sistema.
- Modelar un esquema lógico de base de datos: Un plano para el diseño conceptual de una base de datos. Muestra aquellas clases que deben ser persistentes.