

DIAGRAMA DE CLASES EN UML

Mg. Juan José Flores Cueto

jflores@usmp.edu.pe

Ing. Carmen Bertolotti Zuñiga

cbertolotti@usmp.edu.pe

INTRODUCCIÓN

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente:

- Diagramas de estructura enfatizan en los elementos que deben existir en el sistema modelado:
 1. Diagrama de clases
 2. Diagrama de componentes
 3. Diagrama de objetos
 4. Diagrama de estructura compuesta (UML 2.0)
 5. Diagrama de despliegue
 6. Diagrama de paquetes
- Diagramas de comportamiento enfatizan en lo que debe suceder en el sistema modelado:
 7. Diagrama de actividades
 8. Diagrama de casos de uso
 9. Diagrama de estados
- Diagramas de Interacción, un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:
 10. Diagrama de secuencia
 11. Diagrama de comunicación
 12. Diagrama de tiempos (UML 2.0)
 13. Diagrama de vista de interacción (UML 2.0)

DIAGRAMA DE CLASES

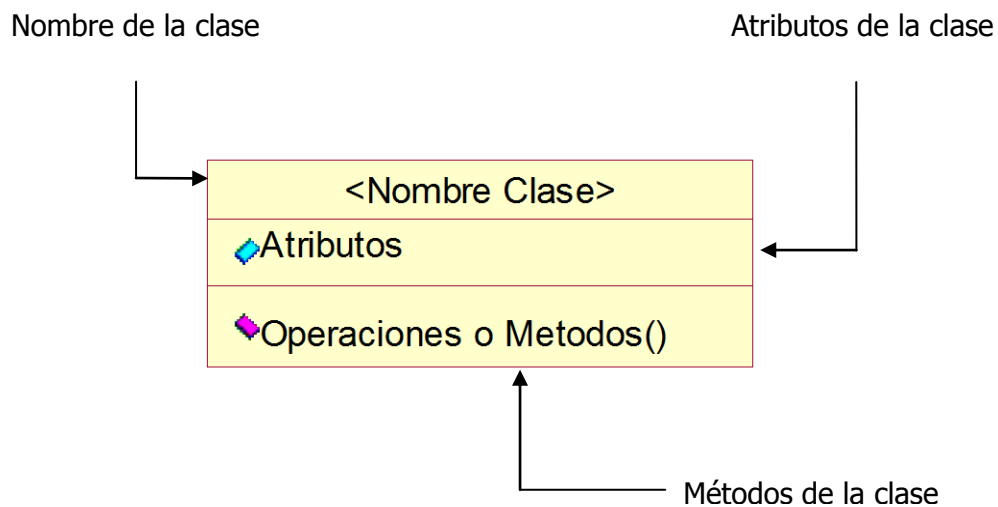
Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro. En un diagrama de clases se pueden distinguir principalmente dos elementos: clases y sus relaciones.

CLASES:

La clase es la unidad básica que encapsula toda la información de un objeto a través de la cual podemos modelar el entorno en estudio.

En UML, una clase es representada por un rectángulo que posee tres divisiones (ver la figura 1).

Figura 1: Representación de una clase en UML



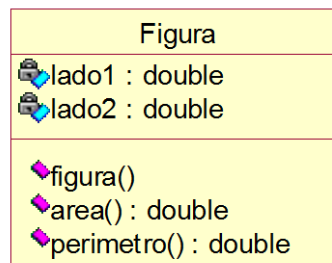
Fuente: Elaboración Propia

En donde:

- El rectángulo superior contiene el nombre de la clase
- El rectángulo intermedio contiene los atributos (o variables de instancia) que caracterizan a la clase (pueden ser *private*, *protected* o *public*).
- El rectángulo inferior contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: *private*, *protected* o *public*).

Por ejemplo, podemos representar una clase denominada Figura que contiene dos atributos (lado1 y lado2) y 3 métodos (método constructor Figura, método área y método perímetro), de la siguiente manera:




Figura 2: Representación de una clase en UML






Fuente: Elaboración Propia

Al analizar la representación de una clase en UML podemos encontrar lo siguiente:

- Los atributos o características de una clase pueden ser de tres tipos, que definen su grado de comunicación y visibilidad con el entorno, estos son:

- **public** (+, ) : Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados
- **private** (-, ) : Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos pueden manipular los atributos)
- **protected** (#, ) : Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser manipulado por métodos de la clase y de sus subclases

- Los métodos u operaciones de una clase son la forma en cómo ésta interactúa con su entorno, éstos pueden tener las características siguientes:

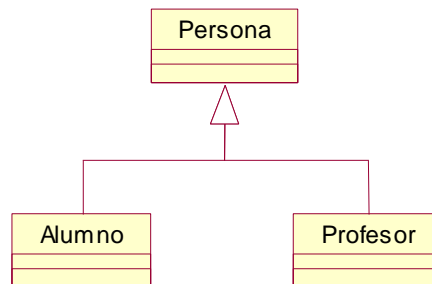
- **public** (+, ) : Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados
- **private** (-, ) : Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden utilizar)
- **protected** (#, ) : Indica que el método no será accesible desde fuera de la clase, pero si podrá ser utilizado por métodos de la clase y de sus subclases

RELACIONES:

1) Herencia (Especialización/Generalización):

Indica que una clase (clase derivada) hereda los métodos y atributos especificados por una clase (clase base), por lo cual una clase derivada además de tener sus propios métodos y atributos, podrá acceder a las características y atributos visibles de su clase base (*public* y *protected*). En la siguiente figura podrá observar un ejemplo de este tipo de relación:

Figura 3: Relación de Especialización/Generalización en UML



Fuente: Elaboración Propia

En este ejemplo se especifica que las clase *Alumno* y *Profesor* heredan de la clase *Persona*, es decir, *Alumno* y *Profesor* podrán acceder a las características de *Persona*. También puede tener su respectiva diferenciación, ya que un *Alumno* puede obtener sus notas previa evaluación realizada por parte de un *Profesor*.

2) Composición:

La composición es un tipo de relación estática, en donde el tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye (el objeto base se construye a partir del objeto incluido, es decir, es parte/todo). En la siguiente figura podrá observar un ejemplo de este tipo de relación:

Figura 4: Relación de Composición en UML

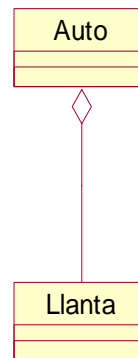


Fuente: Elaboración Propia

3) Agregación:

La agregación es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye (el objeto base utiliza al incluido para su funcionamiento). En la siguiente figura podrá observar un ejemplo de este tipo de relación:

Figura 5: Relación de Agregación en UML



Fuente: Elaboración Propia

4) Dependencia o Instanciación (uso):

Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada. El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana (la creación del Objeto Ventana está condicionado a la instanciación proveniente desde el objeto Aplicación).

Figura 6: Relación de Dependencia o Instanciación en UML



Fuente: Elaboración Propia

Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

5) Asociación: →

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro. En la siguiente figura podrá observar un ejemplo de este tipo de relación:

Figura 7: Relación de Asociación en UML



Fuente: Elaboración Propia

Del ejemplo se puede deducir que una persona puede usar diferentes tipos de ropa (varias ropas), en cambio una ropa solo puede ser usada por sólo una persona en un momento determinado.

Los elementos adicionales que pueden aparecer en una relación de este tipo son los siguientes:

- **Rol:** Identifica con nombres a los elementos que aparecen en los extremos de la línea que denota la relación, dicho nombre describe la semántica que tiene la relación en el sentido indicado.
- **Multiplicidad:** La multiplicidad de una relación determina cuantos objetos de cada tipo intervienen en la relación. Presenta las siguientes características:

MULTIPLICIDAD	SIGNIFICADO
1	Uno y solo uno
0..1	Cero o uno
X..Y	Desde X hasta Y
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios

- Cada asociación tiene dos multiplicidades (una para cada extremo de la relación)
- Para especificar hay que indicar que la multiplicidad mínima y máxima (mínima...máxima)
- Cuando a multiplicidad mínima es 0, la relación es opcional
- Una multiplicidad mínima mayor igual que 1 establece una relación obligatoria