***Conceptos UML***

*Salgado Landa Cristian Gael*

*Edith Nancy Báez Pérez*

*Laboratorio Empresarial*

*Secuencia 2GM2*

*Licenciatura en Negocios Digitales*

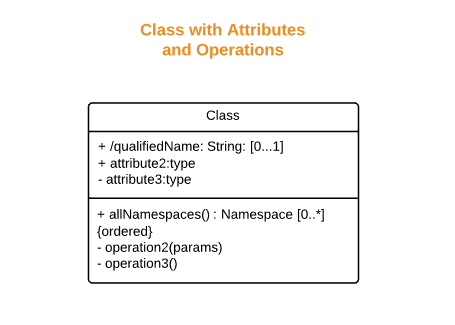
*Instituto Politécnico Nacional*

**La Clase:**

En los diagramas de clases, la clase es un elemento-modelo y una especialización del clasificador encapsulado (*EncapsulatedClassifier*) y del clasificador de comportamiento (*BehavioredClassifier*). **Comprende un conjunto de instancias**. Las instancias-objeto dentro de una clase comparten las mismas propiedades (*Attributes*) y comportamientos (*Methods*), así como la misma semántica, es decir, utilizan los mismos caracteres con el mismo significado. Esto convierte a la clase en una especie de plantilla para sus objetos, “instanciando” los objetos y definiendo su conducta en el sistema.

Las clases poseen **propiedades** que las definen tanto a ellas como a sus objetos subordinados, entre ellas:

* Propiedades (*Properties* o *Attributes,* si pertenecen a la clase)
* Operaciones o métodos (*Operations*, pueden invocarse para un objeto)
* Receptores (*Receptions*) desde UML 2.0
* Conexiones (*Ports*) desde UML 2.0
* Conectores (*Connectors*)

[](https://www.ionos.mx/digitalguide/fileadmin/DigitalGuide/Screenshots_2019/klassendiagramme-uml-ES-2.png)

La notación más simple de una clase en los diagramas de clases UML representa gráficamente el nombre de la clase, los atributos y las operaciones en áreas separadas del mismo rectángulo en que se escriben.

Cuando se crea un diagrama de clases, estas propiedades se añaden a la notación. En UML, las clases se representan como un rectángulo dibujado con línea continua y compuesto por tres filas horizontales. Solo es obligatorio modelar la superior porque aquí se define el **nombre de la clase**. Las otras dos, una que contiene a los **atributos** (intermedio) y una que expresa las **operaciones** o los métodos que puede utilizar la clase (inferior) son opcionales. Añadiendo a los nombres de estos miembros (atributos y métodos) uno de los símbolos que se detallan a continuación (modificadores de acceso a miembros), se especifica su visibilidad:

* **+** = pública
* **-** = privada
* **#** = protegida
* **/**= derivada (
* **~**= paquete
* **\***= aleatoria

**Propiedades:**

Las propiedades son **elementos interdependientes**. Los atributos propios de la clase (*ownedAttributes*) son siempre roles y se conectan por medio de conectores. Si poseen la propiedad *isComposite= true*(“está compuesto = verdad”), reciben el nombre de partes (*Parts*).

La propiedad UML es una **característica de las estructuras** que puede tener aplicaciones diferentes. Además de la función de atributo en una clase, también puede representar los extremos en las relaciones de asociación.

El **tipo de propiedad** se deriva del nombre del clasificador, pero también se puede especificar un valor estándar para una propiedad, y los modificadores pueden concretar cómo se comporta:

* Ordenado (notación: *isOrdered = true*)
* Único (*isUnique = true*)
* No único (*isUnique = false*)
* Protegido contra escritura (la propiedad solo puede leerse: *isReadOnly = true*)
* Secuencia (la propiedad es una colección ordenada: *isUnique = false* e *isOrdered = true*)
* Asociación (derivada de subconjuntos: *union*)
* ID (pertenece a la denominación de su clasificador, *id*)
* Limitación de propiedad (una limitación que influye en la propiedad: *property-constraint*)
* Redefinición de una propiedad (redefine a una propiedad heredada: *redefines [Merkmalsname]*)
* Subconjunto de la propiedad (simboliza a una propiedad que es un subconjunto de una propiedad: *subsets [Merkmalsname]*)

**Operaciones:**

Las operaciones son **funciones de comportamiento** y se utilizan en clases, en tipos de datos o en interfaces. Las operaciones invocan la instancia de una clase y especifican estos aspectos de una llamada:

* Nombre
* Tipo
* Parámetro
* Limitaciones

La operación pertenece al clasificador al cual se subordina, que podría modificarla redefiniendo el tipo o el parámetro.

Antes de que una operación se ejecute, se ha de cumplir una serie de **condiciones previas**, pero UML no define cómo se comporta la llamada de comportamiento si no se cumplen las condiciones previas. Se han de especificar también las **condiciones posteriores** que se han de cumplir si la operación finaliza. Las **condiciones de campo** restringen el valor devuelto a un valor que se calcula a partir de sus especificaciones y que ha de cumplir con las condiciones posteriores. Pero mientras se ejecuta, la operación también puede invocar una **excepción**. Si esto sucede, probablemente no pueda cumplir con las condiciones posteriores.

La notación para diagramas de clases prescribe que las operaciones se escriban en el **cuerpo de la clase**. Si bien son **datos obligatorios** según el estándar UML, puede prescindirse de ellos. Solo el nombre es imperativo.

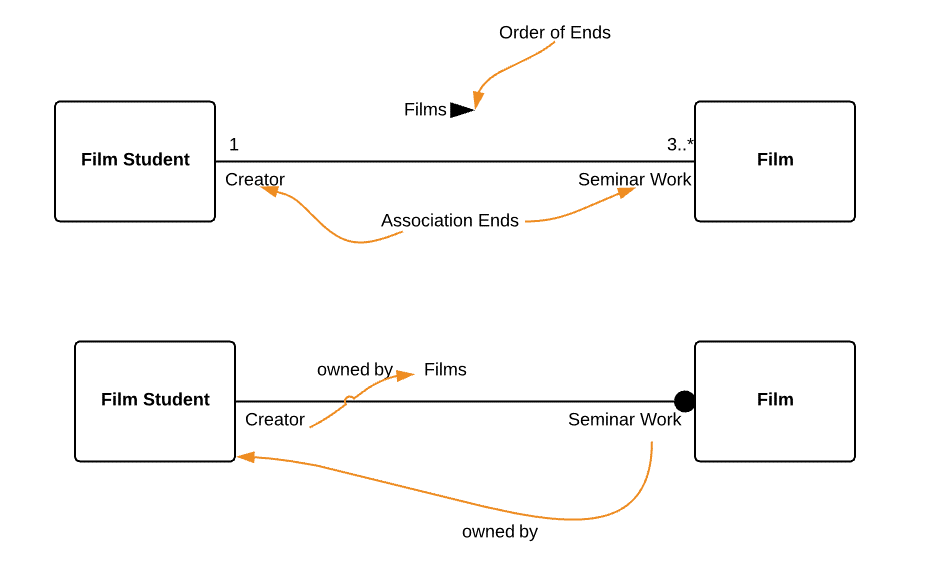
La **abstracción** conecta elementos de capas diferentes. También puede mostrar perspectivas diferentes. Los elementos mencionados en esta relación de dependencia representan al mismo concepto. Según el estándar UML el elemento más específico es el cliente, que depende del proveedor, el elemento más abstracto. El final de la flecha debería estar en la subclase y la punta en la superclase. Pero UML autoriza también una notación inversa. Si es más conveniente que el elemento abstracto depende de su subclase la punta de la flecha se dibuja en el elemento más específico.

La abstracción tiene **dos subclases**: la realización (***Realization***) y la manifestación (***Manifestation***).

**La asociación**

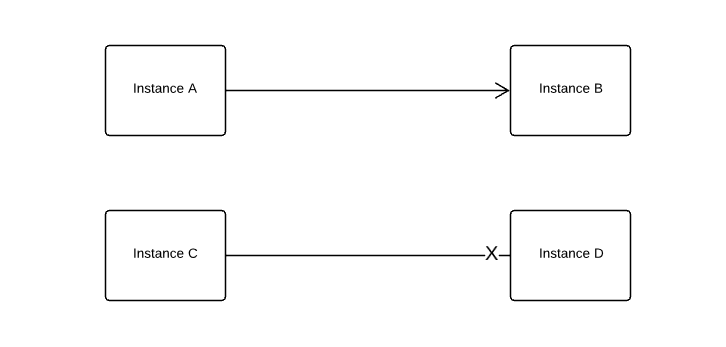
La asociación es una relación que conecta **tuplas**. En informática, las tuplas son colecciones de datos ordenadas. Al contrario que en los conjuntos, aquí intervienen conexiones y secuencias lógicas. Por eso, no sería erróneo asignar también un componente estructural a la asociación que se sume al título oficial de **relación semántica**. La asociación es una conexión entre clasificadores. Los elementos de esta relación tienen una **proximidad lógica o física**. En función de la cantidad de miembros, la asociación se llama binaria (dos instancias), ternaria (tres instancias) o n-aria (a partir de cuatro instancias).

Los **extremos de una asociación** en los diagramas de clases UML conectan asociaciones con instancias. El extremo tiene un nombre que expresa el **rol** de la instancia en cada relación. Pensemos en un estudiante que graba varias versiones de un cortometraje para un seminario de cine. El rol del estudiante en relación con el film sería el de “creador”. El rol de la película sería “trabajo de seminario”. Estos dos nombres se escriben bajo la línea que los une, junto al símbolo de la instancia que describen. El extremo pertenece a la asociación misma o al clasificador. En el caso de tratarse de más de dos extremos, el rol pertenece a la asociación.

[](https://www.ionos.mx/digitalguide/fileadmin/DigitalGuide/Screenshots_2019/klassendiagramme-uml-ES-7.png)La instancia “Film student” tiene el rol “creador” y está unida a la instancia “Film” por medio de la asociación “Films”. En la asociación puede añadirse una multiplicidad (3..\*) como en el diagrama superior.

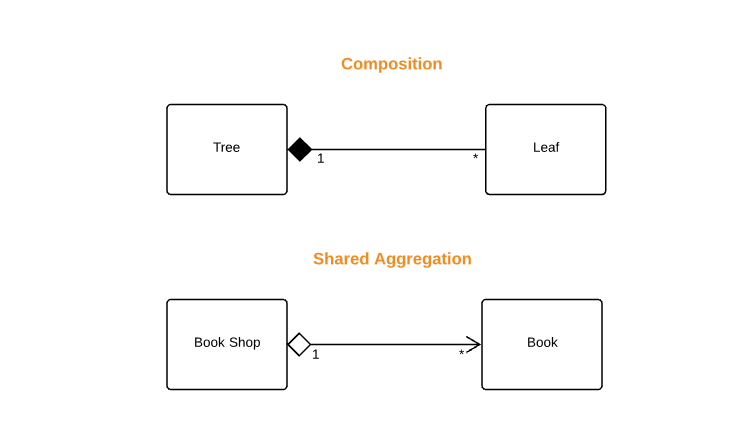
La **flecha junto al nombre de la asociación** en el diagrama de clases de arriba indica la dirección de la relación. En el diagrama inferior el punto en la instancia “Film” señaliza que el extremo “Seminar work” pertenece a la instancia “Film student”. Como el extremo “Creator” no tiene esta marca, pertenece a la asociación misma. La **multiplicidad** “1” muestra que existe una sola instancia “Film student”, mientras que la instancia “Film” tiene por lo menos tres manifestaciones.

La **navegabilidad** es una propiedad final (*End Property*) que muestra si puede accederse a una **instancia** en este final de la asociación desde el otro final de la asociación. Si la instancia B es **accesible** para instancia A, se dibuja una **punta abierta de flecha** en la línea de la asociación en dirección a instancia B exactamente en el símbolo de la instancia B. Si la instancia C no puede acceder a la instancia D, se dibuja una **X** en la línea en la instancia D. Si no quieres indicar navegabilidad, no se escribe nada en especial.

[](https://www.ionos.mx/digitalguide/fileadmin/DigitalGuide/Screenshots_2019/klassendiagramme-uml-ES-8.png)Los extremos de las asociaciones que pertenecen a una clase son siempre navegables. Si pertenecen a la asociación, es posible que no lo sean.

Hay dos variantes de la asociación: el enlace (link) y la agregación.

* El **enlace** es una instancia de la asociación. Dispone de **al menos dos extremos** con una multiplicidad cada uno. Este valor ha de ser una instancia del tipo datos de la terminación. En nuestro ejemplo un estudiante rueda tres películas mientras estudia. El valor para la instancia “estudiante” es “1”; para la instancia “Película”, “3”. La conexión se modela como línea continua entre los participantes. Al contrario que la asociación, el enlace conecta instancias, pero no clasificadores.
* La **agregación** es una asociación binaria: tiene siempre **dos participantes**. Al contrario que el enlace, la agregación no crea ninguna relación en el mismo nivel, sino que muestra relaciones entre una parte y el todo. La agregación se representa por una propiedad en el extremo de la asociación en forma de rombo junto a la instancia que representa el todo.

[](https://www.ionos.mx/digitalguide/fileadmin/DigitalGuide/Screenshots_2019/klassendiagramme-uml-ES-9.png)La agregación es una relación entre una parte y el todo. La composición (rombo negro) tiene una conexión fuerte y la agregación, una débil.

La subespecie **composición** (*Composite Aggregation*) describe la relación entre un conjunto de partes y una parte de este todo. Si el sistema elimina el todo (la composición o conjunto de partes) también destruye todas las partes, p. ej., si un árbol es el todo, una hoja es una parte, y si el árbol cae víctima de un incendio, este incendio también destruye a las hojas. Para representar a esta relación en un diagrama de clases se dibuja una línea continua entre las instancias y se modela un **rombo negro** en el lado de la composición (en nuestro ejemplo, la instancia “árbol”). A este lado se le llama también **extremo de la agregación**.

La segunda subespecie de la agregación es la **agregación compartida** (o agregación a secas). Esta **relación asimétrica** existe entre una **propiedad** (*Property*) y una instancia que representa a un **conjunto de instancias**. La relación debería ser directa porque si no una instancia composición podría interpretarse como parte de sí misma. Esto podría pasar si se modela la relación de dependencia de forma cíclica. La propiedad compartida puede pertenecer a varias composiciones. Al mismo tiempo, su instancia puede existir de forma independiente a la composición. Si el sistema borra a una composición (o a todas) la instancia parte puede seguir existiendo, por eso esta relación se considera débil en comparación con la composición.

La asociación presenta otra particularidad: la **clase asociación**, clase y relación al mismo tiempo. Esto hace que puedan asignarse atributos a la clase asociación en el diagrama de clases.

**Bibliografía:**

Desarrollo Web. (2019, 12 febrero). *Diagramas de clases: crea diagramas estructurales con UML*. IONOS Digitalguide. <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/diagramas-de-clases-con-uml/>