Capítulo 3: Análisis Orientado a Objetos

Introducción

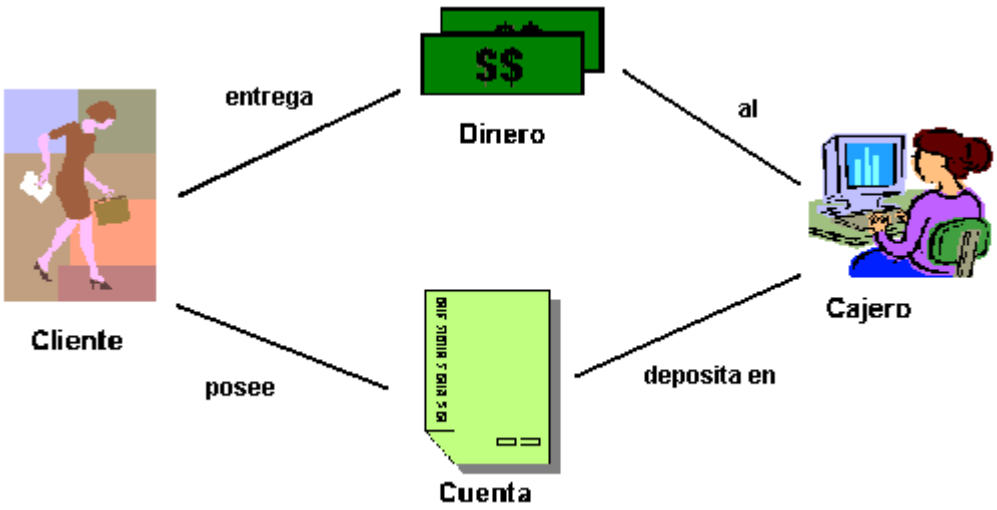
Se han hecho distintos enfoques del análisis de sistemas, con éxito variado. El *análisis de sistemas orientado a objetos* es un enfoque relativamente reciente para capturar y organizar la información pertinente al posterior diseño e implementación de un sistema de software. El enfoque hace énfasis en la comprensión del sistema y su documentación. La comprensión se realiza construyendo un modelo conceptual del sistema bajo estudio. La documentación resultante provee una base consistente para el diseño e implementación del sistema.

El modelo de análisis consiste en una especie de *"maqueta imaginaria"* que representa los aspectos de la realidad que son relevantes para el Sistema. La construcción de este modelo se hace con el fin de contextualizar hasta dónde debe abarcar el Sistema y qué cosas no interesan y deben quedar afuera del mismo.

La construcción del *Modelo de Análisis* se hace identificando los *Objetos* presentes en la realidad que son relevantes y cómo se relacionan entre sí, dejando afuera a todos aquellos que no son de importancia. Una vez identificados tales objetos y relaciones, se deben identificar las *Clases* a las que pertenecen tales Objetos y definir las *asociaciones* correspondientes entre ellas.

Los objetos de la realidad no existen en forma aislada, sino que se relacionan entre sí. Por ejemplo, si consideramos una realidad correspondiente a un Banco, vemos que se destacan los siguientes objetos: los *clientes* del banco, los *cajeros* del banco, las *cuentas* que poseen los clientes y el *dinero* que depositan y retiran de ellas. Dichos objetos se relacionan entre sí. Los clientes *entregan* dinero a los cajeros. Los cajeros *depositan* ese dinero en las cuentas que poseen los clientes.

A la hora de construir el *Modelo de Análisis* no solamente representaremos en el mismo a las clases de objetos observadas en la realidad, sino que también plasmaremos en el modelo las relaciones existentes entre los objetos de dichas clases. Dichas relaciones entre objetos serán representadas mediante *asociaciones* entre las clases a las cuales pertenecen dichos objetos. Una *asociación* entre dos clases es una representación conceptual de una relación concreta que puede darse entre dos objetos cualesquiera de una y otra clase.



Por ejemplo, en la realidad correspondiente al banco podrían darse las siguientes relaciones concretas:

- El cliente *Juan Pérez* entrega $100 al cajero *Luis Peña*
- El cliente *Juan Pérez* entrega $200 al cajero *Hugo Sosa*
- El cliente *Mafalda Coto* entrega $200 al cajero *Hugo Sosa*
- El cliente *Teresa Machado* entrega $200 al cajero *Luis Peña*

Observamos que constantemente se repite el hecho de que un cliente en concreto entrega dinero a un cajero en concreto. La forma de representar esta situación será definiendo una *asociación* llamada "entrega dinero" entre las clases *Cliente* y *Cajero*. La asociación representa al conjunto de todas las parejas de objetos (Cliente,Cajero) que pueden relacionarse al momento de entregar el dinero.

Al igual que cuando identificamos en la realidad a las clases relevantes para nuestro Sistema a construir, también es muy importante identificar las asociaciones relevantes entre dichas clases. Dos clases que hemos incorporado a nuestro modelo de análisis pueden tener muchas asociaciones entre sí, pero solo debemos elegir aquellas que sean relevantes para nuestros requerimientos.

Por ejemplo, entre las clases *Cliente* y *Cajero* pueden haber montones de asociaciones: El cliente *saluda* al cajero, el cliente *observa* al cajero cuando deposita el dinero, el cliente *se despide* del cajero, etc. Todas ellas son irrelevantes para nuestro Sistema.

La *herencia* no es más que un tipo especial de asociación. Cuando decimos que la clase *Vendedor* es derivada de la clase *Empleado*, estamos diciendo simplemente que un Vendedor ***es*** un Empleado. El nombre de la asociación entre las clases Vendedor y Empleado sería justamente ***es***.

Notación UML para la construcción del Modelo de Análisis

A la hora de construir cualquier modelo (en particular nuestro Modelo de Análisis), es vital la definición de una *notación* unificada para su construcción y posterior comprensión por parte de otros desarrolladores. Una notación es una *forma de expresar o escribir* utilizada para describir diferentes aspectos de interés durante cualquier proceso de construcción.

En el caso del análisis de un sistema Orientado a Objetos, tales aspectos de interés son esencialmente nombrar o describir las clases que queremos que el programa tenga o las asociaciones existentes entre ellas.

La existencia de una notación ayuda a realizar las actividades de la metodología con *mucha* más facilidad y planificación y además sirve como forma de *documentación* del proceso de construcción. Esto es algo muy bueno por muchas razones: rastrear errores si éstos ocurren, saber a qué clase recurrir para resolver algo, tener por escrito una suerte de *contrato* con el cliente ante eventuales discrepancias entre lo pedido por él y lo entregado por nosotros, etc.

Existen *muchas* notaciones que podemos utilizar al realizar un Análisis Orientado a Objetos. Algunas notaciones son más formales que otras, algunas más organizadas y completas que otras y algunas más gráficas que otras, pero todas tienen el mismo propósito: *describir con la mayor claridad posible los distintos aspectos del Sistema a efectos de lograr la mayor comprensión posible del mismo.*

Algunas notaciones para realizar el Modelo de Análisis de un Sistema Orientado a Objetos son las siguientes:

1. Lenguaje natural
2. Modelo Objeto Relación
3. UML

El *Lenguaje natural* es simplemente redactar en idioma español las descripciones de los aspectos que interesen. Es muy fácil de usar porque todos lo conocemos, pero es poco práctico porque tenemos que escribir *mucho* para expresar con claridad lo que queremos.

El *Modelo Objeto Relación* (MOR) es un lenguaje gráfico que permite describir - mediante un dibujo - las clases que debe tener el Sistema y cómo se relacionan entre sí. Es más práctico que el Lenguaje natural, y durante el auge de la metodología Orientada a Objetos fue un lenguaje muy popular, sustituido posteriormente por la notación UML.

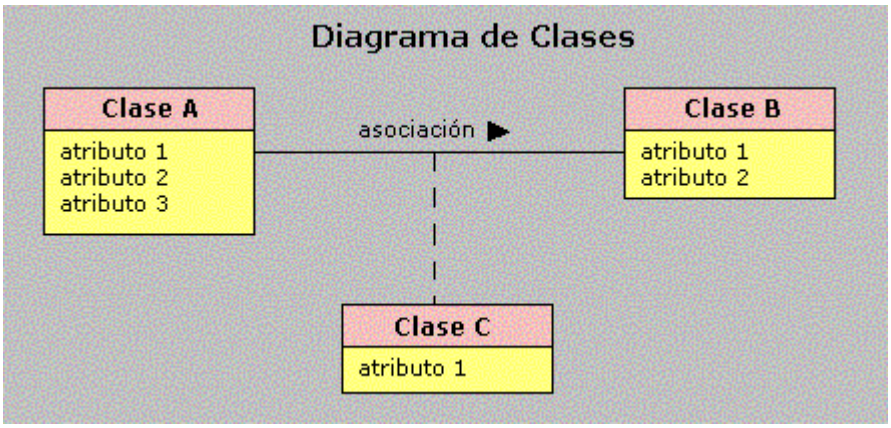
UML es un lenguaje gráfico que posee *muchos* diagramas que permiten describir *todos* los aspectos de interés relativos al Sistema en construcción (cuáles son sus requerimientos, qué clases y asociaciones tiene, cómo deben comportarse sus métodos, cómo programarlos eficientemente, etc.). Este lenguaje es más nuevo y completo que los anteriores. Por esta razón es que lo hemos elegido por sobre todas las demás notaciones como la notación a utilizar en el curso.

Dentro de los distintos diagramas provistos por la notación UML, el que se utiliza para la construcción del Modelo de Análisis recibe el nombre de *Diagrama de Clases Conceptual*.

El Diagrama de Clases Conceptual

El *Diagrama de Clases Conceptual* es particularmente apropiado como *Modelo de Análisis* porque reúne a los dos grandes elementos que debe tener el mismo, es decir *Clases* y *Asociaciones*. Por lo tanto, nuestra "maqueta imaginaria" no será más que un dibujo que la notación UML ha bautizado como Diagrama de Clases Conceptual.

En lo sucesivo, utilizaremos los términos *Modelo de Análisis* y *Diagrama de Clases Conceptual* en forma indistinta, porque en el marco de este curso son sinónimos. No obstante, es importante recordar que si UML ***no*** fuera la notación utilizada en el contexto de la Metodología, no podríamos usarlos como sinónimos porque no todas las notaciones representan el Modelo de Análisis de la misma manera.



Un ejemplo de un Diagrama de Clases Conceptual

Cuando nos enfrentemos a un nuevo Sistema a desarrollar y debamos construir su *Modelo de Análisis*, lo que haremos será estudiar las *Clases* y *Asociaciones* relevantes en la realidad correspondiente a dicho Sistema y dibujar un *Diagrama de Clases Conceptual* que las represente.

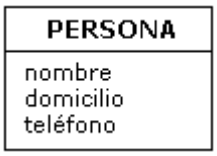
Es muy importante tratar de que el Diagrama las represente lo más fielmente posible porque luego en la etapa de *Diseño* utilizaremos dicho diagrama junto con los *requerimientos* del sistema como punto de partida en la toma de decisiones de implementación. Si el Diagrama de Clases Conceptual no es consistente con la realidad del problema, difícilmente podamos diseñar un Sistema que se apegue a la misma porque estaremos partiendo de un Análisis que no se apega.

En la notación UML existen, de hecho, dos tipos de Diagramas de Clases: El *conceptual* y el de *implementación*. El *Diagrama de Clases Conceptual* se construye durante la etapa de *Análisis* y es el que estudiaremos a lo largo de este capítulo. El Diagrama de Clases de implementación se construye durante la etapa de *Diseño* y sirve para definir las Clases *propias del lenguaje de programación a usar* junto con sus métodos. Veremos oportunamente que el Diagrama de Clases de implementación se construye a partir del Diagrama de Clases Conceptual.

En lo que resta del capítulo nos centraremos en el estudio detallado de la sintaxis del Diagrama de Clases Conceptual y presentaremos diversos ejemplos que faciliten su comprensión.

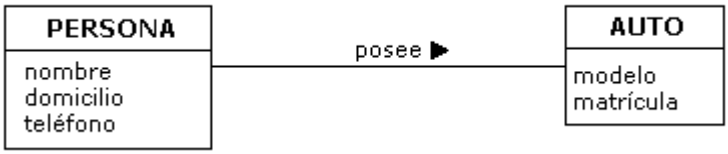
Clases y Asociaciones

Cada *Clase* se representa en el diagrama por medio de un recuadro dividido en dos rectángulos. El rectángulo superior contiene el nombre que identifica a la Clase mientras que el rectángulo inferior contiene los atributos definidos para la Clase. Por ejemplo:



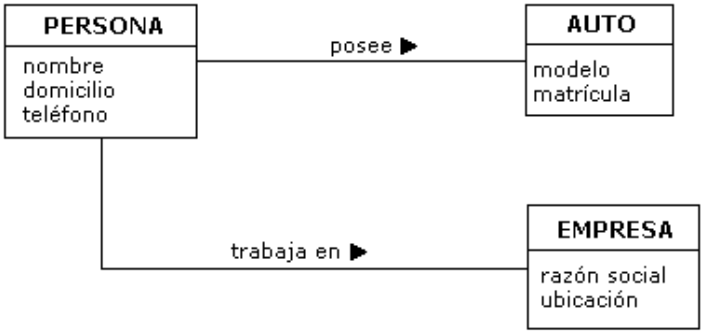
Recordemos que los atributos de una Clase especifican los *datos* relevantes de los objetos pertenecientes a dicha Clase. Se acostumbra poner el nombre de la Clase en mayúsculas y los atributos en minúsculas. Si una clase no tiene atributos relevantes o todavía no se tiene bien claro cuáles son, el rectángulo inferior puede no ponerse.

Dentro del Diagrama existen muchas *Clases*. Ya sabemos que entre las diferentes clases existen *Asociaciones*. La forma de representar una asociación entre dos clases es mediante una línea recta (generalmente horizontal o vertical) que une a las Clases involucradas. Dicha línea se etiqueta con el nombre de la asociación y se le asigna un sentido, el cual sólo sirve para indicar en qué sentido debe *leerse* la asociación. Por ejemplo:



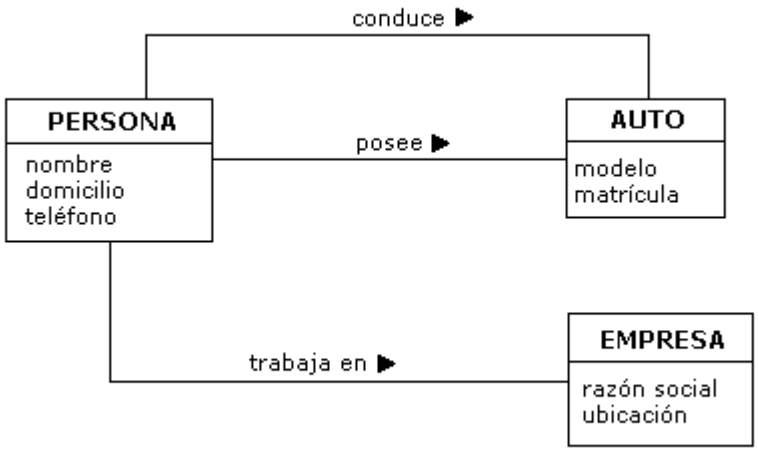
En este ejemplo, la asociación debe leerse como *Persona posee Auto* pero no debe leerse como *Auto posee Persona*.

De una misma Clase pueden salir líneas de asociación a varias Clases diferentes, y no todas las Clases del diagrama necesariamente deben estar asociadas. Por ejemplo:



En este ejemplo, salen dos asociaciones de la Clase Persona (*posee* y *trabaja en*) y no existe ninguna línea de asociación entre las Clases Auto y Empresa.

Puede ocurrir también que entre dos Clases se coloque más de una línea de Asociación. Esto se hace cuando interesa representar más de una Asociación entre las dos Clases. Por ejemplo:



Esto no quiere decir que *siempre* haya que tener varias asociaciones entre dos Clases. Sólo quiere decir que *si la realidad así lo exige*, UML nos lo permite representar. En el ejemplo anterior hay dos Asociaciones entre las clases Persona y Auto, pero hay una sola Asociación entre las Clases Persona y Empresa.

Multiplicidad

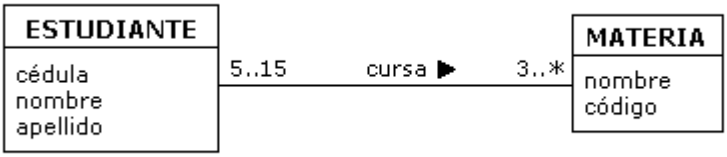
Sabemos que entre dos Clases pueden existir una o más líneas de Asociación. Sin embargo, aún no hemos dicho nada acerca de la *cantidad* de objetos de dichas Clases que pueden relacionarse según las Asociaciones existentes entre ellas.

Por ejemplo, sabemos que las Clases *Cliente* y *Cuenta* de la realidad correspondiente al Sistema para el banco se encuentran asociadas entre sí, ya que los Clientes *poseen* Cuentas en el banco. Surgen entonces las siguientes preguntas: ¿Cuántas Cuentas posee un Cliente? ¿A cuántos Clientes puede pertenecer una misma Cuenta?. Para reflejar las cantidades de objetos que pueden participar en una Asociación entre dos Clases, se introduce el concepto de *multiplicidad*.

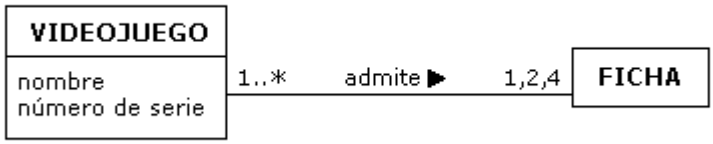
La *multiplicidad* es una notación que se incluye en *ambos* extremos de una línea de Asociación entre dos clases. La misma consiste en marcar cada extremo de la línea de Asociación con la cantidad de *instancias* (objetos) de la Clase correspondiente a *ese* extremo que pueden relacionarse con *una* instancia de la Clase que se encuentra en el *otro* extremo de la línea de Asociación.

Veamos tres ejemplos:

Esta Asociación debe leerse así: *"Cada Persona posee cero o más Autos"* y *"Cada Auto es poseído por una sola persona"*.



Esta Asociación debe leerse así: *"Cada Estudiante cursa tres o más Materias"* y *"Cada Materia es cursada por entre cinco y quince Estudiantes"*.



Esta Asociación debe leerse así: *"Cada Videojuego admite una, dos o cuatro Fichas"* y *"Cada Ficha es admitida por uno o más Videojuegos"*.

Un par de consideraciones a tener en cuenta: La primera es que si en un extremo de una línea de Asociación no se incluye multiplicidad, se asume que la misma vale 1 (uno). La segunda es que la multiplicidad * se puede leer como *"cero o más"* o simplemente como *"muchos"*. A los efectos del Diagrama, ambas lecturas se consideran indistintas.

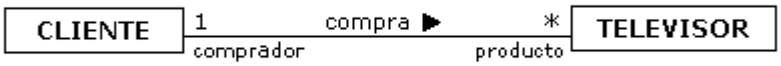
Para terminar, cabe mencionar que al igual que con las Clases y Asociaciones, las multiplicidades a introducir en los extremos de una línea de Asociación entre dos Clases surgen de la realidad que estamos observando al construir el Diagrama de Clases.

Roles, Autoasociaciones y Restricciones

Hasta el momento hemos visto cómo dibujar las Clases del diagrama, las Asociaciones existentes entre ellas y las multiplicidades correspondientes a las Asociaciones. Veremos ahora tres nuevos elementos que pueden formar parte del Diagrama de Clases.

Roles

Si se desea, se puede bautizar cada extremo de una línea de Asociación con el *rol* que desempeñan en la Asociación los objetos pertenecientes a la clase correspondiente a dicho extremo. Esto sirve simplemente para entender con mayor claridad la forma en que los objetos participan de la Asociación, pero no cambia el significado ni las multiplicidades de la misma. Por ejemplo:



Autoasociaciones

Estamos acostumbrados a que un objeto de una Clase se asocie con un objeto de otra Clase diferente. Sin embargo, a veces ocurre que un objeto se puede asociar con otro objeto de la misma Clase. Esto puede representarse en el Diagrama de Clases mediante una Autoasociación. Esto es, una línea de Asociación que va de una Clase hacia sí misma.

Por ejemplo, en la realidad correspondiente a un Sistema de Software para el registro civil de la ciudad de Montevideo, ocurre que las Personas se casan entre sí. Esto significa que un objeto de la Clase Persona se asocia con otro objeto de la misma Clase.

Cuando dos personas se casan, adoptan diferentes roles. Una de ellas se convierte en marido y la otra en esposa. Los *roles* que etiquetan los extremos de una línea de Asociación suelen usarse más comúnmente en las Autoasociaciones que en las Asociaciones entre Clases disjuntas. Esto es porque generalmente cuesta más diferenciar el rol que cumplen los objetos cuando pertenecen a la misma Clase que cuando pertenecen a Clases diferentes.

Restricciones

Vamos a ver que en ocasiones puede suceder que el diagrama permite modelar situaciones *inconsistentes* con la realidad del problema. Es decir, permite reflejar situaciones específicas que no ocurren en la realidad correspondiente al problema, o que no deberían ocurrir.

Por ejemplo, consideremos nuevamente la realidad correspondiente al Sistema de Software para el registro civil de la ciudad de Montevideo. Ya vimos que en el registro civil las personas se casan. Sin embargo, nunca ocurre que una Persona se casa consigo misma pero el diagrama que dibujamos antes no refleja esta situación, pues permite que, por ejemplo, el objeto "Juan Pérez" se case con sí mismo.

Necesitamos entonces incorporar al diagrama un nuevo elemento que evite la ocurrencia de situaciones inconsistentes con el dominio del problema. Dicho elemento se conoce como *restricción*. Una restricción es una frase que dice explícitamente cuál es la situación inconsistente que no debe ocurrir. Las restricciones se escriben al pie del Diagrama de Clases encerradas entre

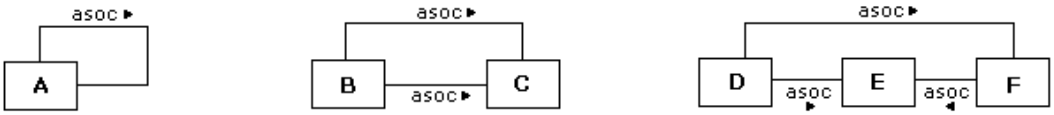
llaves { }. Para un mismo diagrama pueden existir cero o más restricciones, dependiendo de la realidad que se esté modelando.

{Una Persona no se puede casar consigo misma}

Una vez incorporada la restricción, el diagrama representa solamente el hecho de que Personas distintas pueden casarse, tal y como queríamos representar.

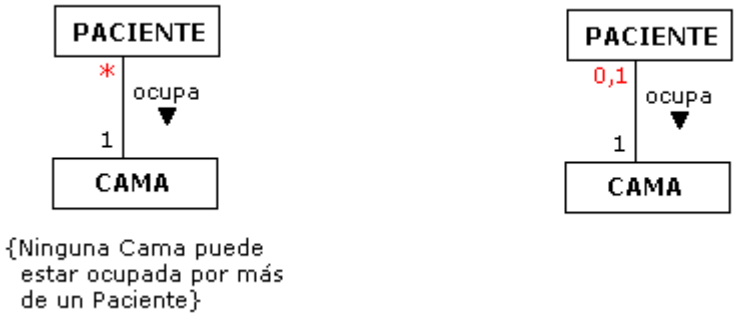
Veamos ahora algunas consideraciones a tener en cuenta a la hora de incorporar restricciones al diagrama. Generalmente, se acostumbra dibujar primero el Diagrama de Clases para la realidad a modelar y luego incorporar las restricciones que sean pertinentes. Para detectar la existencia de restricciones se suelen aplicar dos estrategias:

1. Buscar *ciclos* en el Diagrama de Clases. Un ciclo es cualquier "camino" en el dibujo que parte de una Clase y llega a sí misma viajando a través de líneas de Asociación. La existencia de un ciclo *no necesariamente* significa que existe una restricción, eso depende de cada realidad y debe estudiarse cada ciclo por separado. Algunos de ellos quizás traigan restricciones aparejadas y otros no, aunque generalmente sucede que en la mayoría de los casos sí las traen. Los siguientes dibujos ilustran ejemplos de ciclos.



2. Considerar aquellos elementos de la realidad no representables en el diagrama. Con esto nos referimos a reglas específicas existentes en la realidad que son difíciles o imposibles de dibujar porque no representan objetos propiamente dichos.

Existe una regla en UML que dice que el Diagrama de Clases debe ser *lo más expresivo posible*. Esto significa que el dibujo en sí mismo debe reflejar lo más fielmente que se pueda la realidad y que ***debe contener la menor cantidad posible de restricciones***. Siempre que sea viable, debemos tratar de que el dibujo en sí mismo no permita la situación inconsistente. En caso de que esto no sea posible (y sólo en ese caso) incorporaremos una restricción. En definitiva, optaremos por incorporar una restricción sólo cuando no tengamos más remedio. Por ejemplo, consideremos los dos dibujos siguientes:



El primero de ellos, si bien es correcto en su sintaxis, no es lo más expresivo posible porque incorporamos una restricción que perfectamente puede evitarse cambiando la multiplicidad de la Asociación. El segundo dibujo incorpora dicho cambio y elimina la restricción porque el propio dibujo impide que una misma Cama esté ocupada por más de un Paciente.

Clases de Asociación

Existen situaciones en las cuales interesa registrar atributos que son propios de una *Asociación* entre dos Clases de Objetos. Por ejemplo, consideremos una realidad referente a negocios inmobiliarios en la cual cada inmobiliaria tiene muchas propiedades en venta y cada propiedad puede estar en venta a través de varias inmobiliarias. Modelamos esta situación así:

Supongamos ahora que queremos registrar en nuestro Modelo de Análisis la *fecha* en la cual cada Propiedad fue puesta en venta en cada Inmobiliaria junto con el *número de contrato* correspondiente. Notemos que:

- No podemos poner esos datos como atributos de la Clase Inmobiliaria, porque son datos que también tienen que ver con las Propiedades. Una misma Inmobiliaria tiene *muchas* Propiedades en venta y con cada una tiene un número de contrato y una fecha de puesta en venta *diferentes*.

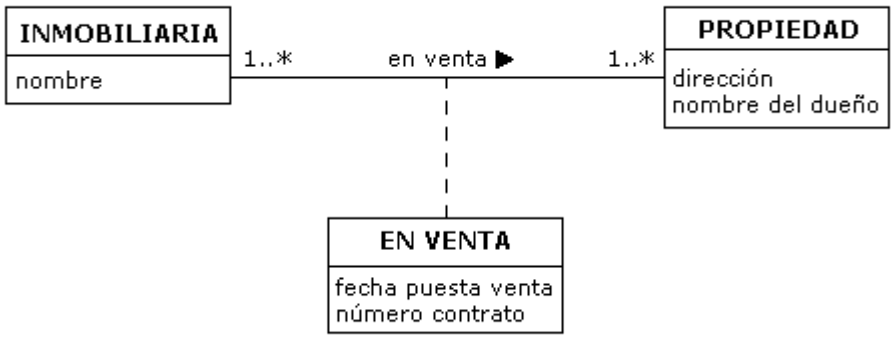
- Tampoco podemos ponerlos como atributos de la Clase Propiedad, porque son datos que también tienen que ver con las Inmobiliarias. Una misma Propiedad puede estar en venta en *varias* Inmobiliarias, y la fecha de puesta en venta y el número de contrato *varían* de una Inmobiliaria a otra.

Por lo tanto, la fecha de puesta en venta y el número de contrato son datos correspondientes a la *relación* entre cada Objeto de la Clase Inmobiliaria y cada Objeto de la Clase Propiedad. Necesitamos entonces un mecanismo que nos permita representar atributos de la *Asociación* entre las dos Clases.

UML nos provee dicho mecanismo, el cual se conoce como *Clase de Asociación*. Una Clase de Asociación es una Clase "artificial" que no aparece explícitamente en la realidad del problema. Es decir, no representa ningún Objeto "real" presente en la realidad. Se introduce al diagrama con el único propósito de representar atributos que son propios de una Asociación entre dos Clases, en

caso de que interese representar datos de la misma. Para las Asociaciones que no tienen datos que sea de interés representar, no ponemos ninguna Clase de Asociación.

La sintaxis para las Clases de Asociación es la misma que para cualquier Clase: Un rectángulo superior con el nombre de la Clase y otro inferior con los atributos de la misma. Se acostumbra que el nombre de la Clase de Asociación sea el *mismo* nombre que el de la Asociación, aunque esto no es obligatorio. Por otro lado, la forma de *vincular* una Clase de Asociación con la Asociación que le corresponde es a través de una línea recta punteada que va de la Clase de asociación a la línea de Asociación. Por ejemplo:



En este ejemplo se incorporó al dibujo la Clase de Asociación llamada "En Venta", la cual contiene como atributos la fecha de puesta en venta de una determinada Propiedad en una determinada Inmobiliaria junto con el número de contrato correspondiente.

Tomemos un momento para analizar el significado de una Clase de Asociación en el Diagrama de Clases. Dijimos que es una Clase "artificial" en el sentido de que no representa ningún Objeto "real" presente en la realidad del problema a resolver. Podemos pensar entonces que la Clase de Asociación entre dos Clases A y B representa un conjunto de Objetos *imaginarios*, que se encargan de vincular a un Objeto de A con uno de B. Cada pareja concreta de Objetos (A,B) tendrá un Objeto de la Clase de Asociación que los relacione.

A efectos de entender estas ideas, retomemos la realidad de los negocios inmobiliarios. Imaginemos que en ella tenemos a los siguientes objetos de las Clases Inmobiliaria y Propiedad:

- La inmobiliaria *Sánchez S.A*
- La inmobiliaria *Rodríguez S.A*
- La Propiedad situada en *18 de Julio 1543* cuyo dueño es *Juan Pérez*
- La Propiedad situada en *Bulevar Artigas 3327* cuya dueña es *Amalia Coto*

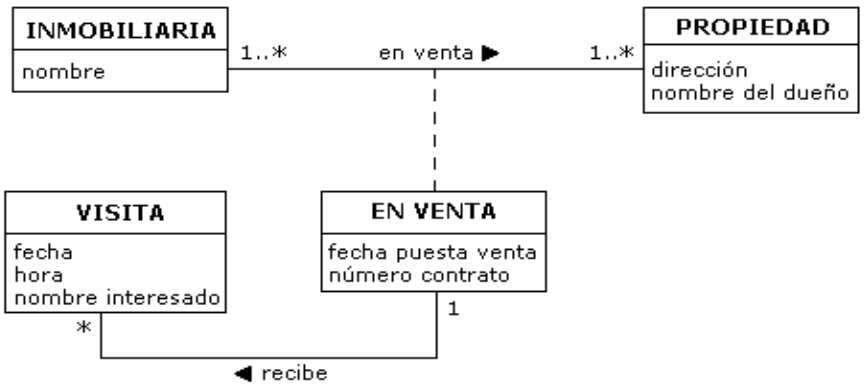
Imaginemos que ambas Propiedades están en venta en ambas Inmobiliarias. A continuación enumeramos los Objetos "imaginarios" pertenecientes a la Clase de Asociación llamada "En Venta".

- El Objeto imaginario "IP1" dice que la Propiedad de 18 de Julio fue puesta en venta en la Inmobiliaria Sánchez S.A. el día 15 de Abril de 2004 con número de contrato 16.533
- El Objeto imaginario "IP2" dice que la Propiedad de 18 de Julio fue puesta en venta en la Inmobiliaria Rodríguez S.A. el día 18 de Abril de 2004 con número de contrato 7.211
- El Objeto imaginario "IP3" dice que la Propiedad de Bvar. Artigas fue puesta en venta en la Inmobiliaria Sánchez S.A. el día 26 de Mayo de 2004 con número de contrato 16.884
- El Objeto imaginario "IP4" dice que la Propiedad de Bvar. Artigas fue puesta en venta en la Inmobiliaria Rodríguez S.A. el día 30 de Mayo de 2004 con número de contrato 7.441

Por ejemplo, el Objeto imaginario "IP1" vincula la Propiedad de 18 de Julio con la inmobiliaria Sánchez S.A. mientras que el Objeto imaginario "IP2" vincula a la *misma* Propiedad con la inmobiliaria Rodríguez S.A. Observemos que cada uno tiene una fecha de puesta en venta y un número de contrato *diferentes*. Eso es porque la *misma* Propiedad fue puesta en venta en cada Inmobiliaria en una fecha *distinta* y bajo *diferente* número de contrato.

Una vez que una Clase de Asociación ha sido incorporada al Diagrama de Clases se la puede tratar como a cualquier otra Clase del diagrama. Es decir, puede participar de Asociaciones del mismo modo que las Clases comunes y corrientes pueden hacerlo.

Por ejemplo, imaginemos que en la realidad de los negocios inmobiliarios queremos registrar todas las ocasiones en las cuales cada Propiedad ha sido mostrada por cada Inmobiliaria a posibles interesados en comprarla desde que la Propiedad fue puesta en Venta en la Inmobiliaria en cuestión. Podemos incorporar al diagrama una nueva Clase que se Asocie con la Clase de Asociación "En Venta". Esa nueva clase será la encargada de representar a todas las Visitas que los interesados hacen a las Propiedades a través de las Inmobiliarias que las tienen en venta.



Cada Visita tiene como atributos la fecha y hora de la visita junto con el nombre del interesado. Intuitivamente sería más natural que la Clase Visita se asociara con la Clase Propiedad, ya que la Visita se hace concretamente a la Propiedad. No obstante, recordemos que también interesa saber a través de cuál Inmobiliaria se coordinó la visita ya que una misma Propiedad puede estar en Venta a través de varias Inmobiliarias. La forma de hacerlo es Asociar la Clase Visita con la Clase de asociación, la cual representa a la "pareja" formada por la Propiedad visitada y la Inmobiliaria que la tiene en Venta encargada de coordinar la Visita.

Agregación y Composición

Ya sabemos que una *asociación* es una representación conceptual de una relación concreta que puede darse entre dos objetos cualesquiera de dos clases. Destacaremos ahora dos tipos especiales de asociaciones que merecen una consideración particular dentro de la notación UML.

Nos referimos a las asociaciones de *Agregación* y *Composición*. Ambas asociaciones describen una relación de *pertenencia en términos físicos* entre objetos, pero se diferencian en la "fortaleza" de dicha pertenencia.

Agregación

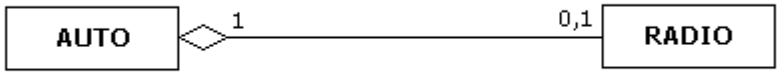
Es un tipo especial de asociación que describe una relación *contiene* entre dos clases de objetos. Esta asociación se produce cuando un objeto de una clase está formado (entre otros) por un objeto de otra clase, pero el objeto contenedor podría seguir existiendo aún si no contuviera al segundo objeto. Por ejemplo, un *Automóvil* contiene una *Radio*, pero en el caso de que esto no fuera así, el automóvil podría seguir existiendo y funcionando normalmente.

Composición

Es un tipo especial de asociación que, al igual que la agregación, también describe una relación *contiene* entre dos clases de objetos, pero se diferencia de la agregación en que el objeto contenedor ***no*** podría seguir existiendo si no contuviera al segundo objeto. Por ejemplo, un *Automóvil* contiene un *motor*, pero en el caso de que esto no fuera así, el automóvil no podría seguir existiendo y funcionando normalmente. A pesar de que el automóvil en realidad sí existiría desde el punto de vista físico, conceptualmente sería un objeto que no tendría utilidad alguna y a esto nos referimos al decir que no existe.

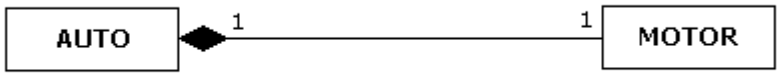
Debido a la importancia que tienen estos dos conceptos en la Orientación a Objetos, UML ha establecido notaciones especiales para representar estos dos tipos particulares de Asociaciones.

La forma de representar una Asociación de Agregación en el Diagrama de Clases es colocando un rombo blanco sobre la línea de Asociación del lado de la Clase que representa al Objeto Contenedor. Por ejemplo:



La multiplicidad 0,1 del lado de la Clase Radio se explica por el hecho de que el Auto eventualmente podría no tener una Radio, pero en caso de tener, tendrá una y sólo una. Nótese que no se etiqueta ningún nombre a la línea de Asociación, ya que el propio rombo blanco dice cuál es el significado de la Asociación. Con respecto a la multiplicidad de la Asociación, la misma se coloca en ambos extremos de la línea de Asociación, igual que como ocurre con las Asociaciones comunes y corrientes.

Con respecto a la Composición, la forma de representarla es colocando un rombo negro sobre la línea de Asociación del lado de la Clase que representa al Objeto contenedor. Por ejemplo:



Al igual que con la Agregación, no se etiqueta ningún nombre a la línea de Asociación. Un punto importante a tener en cuenta para este tipo de Asociación es que la multiplicidad de la misma del lado de la Clase que representa al Objeto contenido *nunca puede ser cero*. Eso es porque el Objeto contenedor necesita contar con el Objeto contenido para que tenga sentido su existencia. Si admitiéramos multiplicidad cero del lado de la Clase que representa al Objeto contenido, estaríamos admitiendo una relación de Composición en la cual *no* existe el Objeto contenido, contradiciendo el significado de la Composición.

En este contexto, cabe resaltar que tanto para la Agregación como para la Composición, tiene sentido cualquier multiplicidad mayor que cero del lado de la Clase que representa al Objeto contenido, ya que el Objeto contenedor podría estar formado por *varios* Objetos de dicha Clase. Con respecto a la multiplicidad del lado de la Clase que representa al Objeto Contenedor, la misma es *casi siempre* 1. Existen algunos casos muy particulares en los cuales podría llegar a ser mayor

que 1, pero en este curso no abordaremos ese tema porque se trata de situaciones sumamente peculiares que no aportan casi nada valioso al estudio de estas Asociaciones.

Herencia

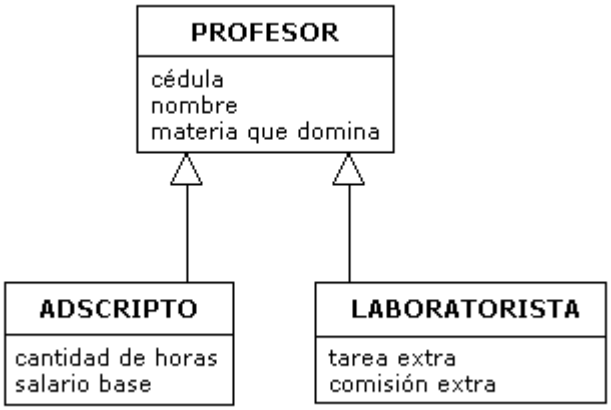
En este tema nos ocuparemos de representar el concepto de Herencia entre Clases de Objetos en nuestros Diagramas de Clases Conceptuales. Comencemos por recordar el concepto de herencia definido en el capítulo anterior.

A nivel conceptual, la Herencia define una relación *"ser"* entre dos Clases de Objetos, las cuales se denominan *clase base* (o *superclase*) y *clase derivada* (o *subclase*). Esta relación significa que todos los objetos de la clase derivada *son* también objetos de la clase base. Sin embargo, no necesariamente todos los objetos de la clase base son también objetos de la clase derivada.

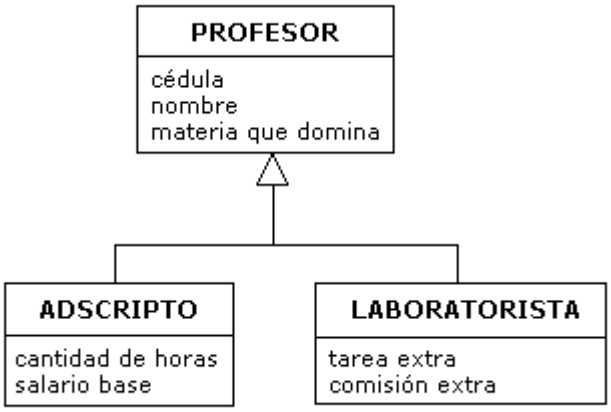
A la hora de detectar la presencia de este particular tipo de Asociación en una realidad, se aplican dos estrategias llamadas *generalización* y la *especialización*. La primera consiste en introducir una nueva clase base que reuniera atributos comunes a varias Clases de la realidad, mientras que la segunda consiste en agregar una nueva clase derivada de otra ya existente.

Se cumple además que la Herencia implica que las Clases derivadas *heredan* (valga la redundancia) todos los atributos de su Clase base, incorporando además otros nuevos, propios de cada Clase derivada.

Veremos a continuación la sintaxis utilizada para representar el concepto de Herencia en nuestros Diagramas de Clases Conceptuales. UML define una notación especial para representar esta Asociación. Se trata de un triángulo blanco que se coloca sobre la línea de Asociación del lado de la Clase base y apuntando hacia ella. Al igual que con la Agregación y la Composición, no se etiqueta ningún nombre a la línea de asociación. A diferencia de ellas, *nunca* se incorpora multiplicidad alguna en los extremos de la línea de Asociación, porque es siempre la misma. Vale 1 del lado de la Clase base y 0,1 del lado de la Clase derivada. Dejamos al lector la tarea de razonar porqué. Ahora veamos un ejemplo:



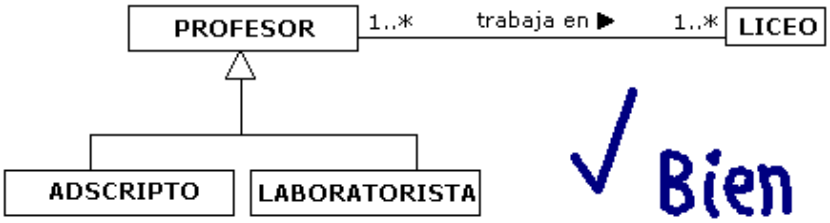
Este diagrama debe interpretarse así: Un Adscripto es un Profesor. Un Laboratorista también es un Profesor. Sin embargo un Profesor puede ser un Adscripto, un Laboratorista o simplemente un Profesor común y corriente que sólo va al liceo a dar clases sin tener ninguna actividad extra dentro del liceo. Otra forma igualmente válida de representar el dibujo anterior es la siguiente:



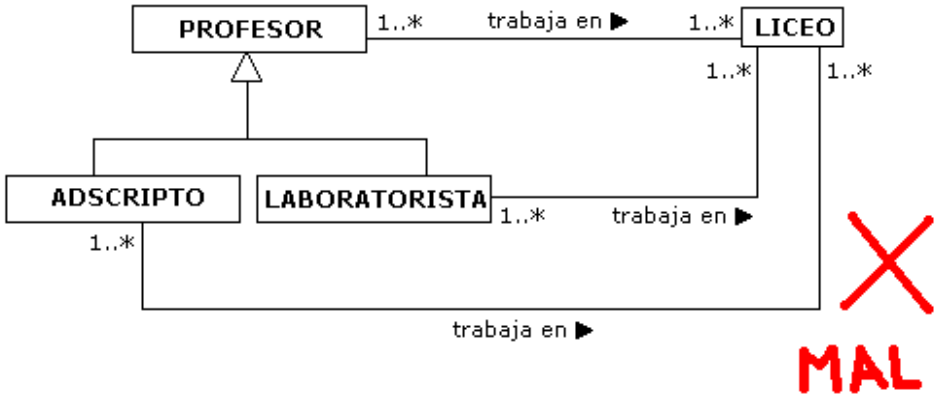
Notemos que la Clase Profesor cuenta con *tres* atributos, mientras que las Clases Adscripto y Laboratorista cuentan con *cinco* atributos cada una, es decir, los tres heredados mas los dos que son propios de cada una. Un posible conjunto de Objetos de la realidad representada por el diagrama podría ser el siguiente:

- El Profesor "Rebo Tonazo" cuya cédula es 1.234.567 y cuya materia es Matemática.
- El Adscripto "Solopa Solista" cuya cédula es 2.345.678, cuya materia es Historia y que además trabaja 3 horas como adscripto y su salario base es de $3500.
- El Laboratorista "Tubode Ensayo" cuya cédula es 3.456.789, cuya materia es Química y que además se encarga de preparar compuestos para experimentos percibiendo una comisión de $50 por cada uno.

Las Clases derivadas no sólo heredan atributos, sino que también heredan las Asociaciones en la que participa su Clase base, heredando también las multiplicidades correspondientes a dichas Asociaciones. Esto hace que no haga falta repetir nuevamente la línea de Asociación en la Clase derivada. Más aún, sería *incorrecto* hacerlo porque estaríamos creando *redundancia de información* en el diagrama. A continuación, ilustramos estas ideas en los siguientes dos diagramas.

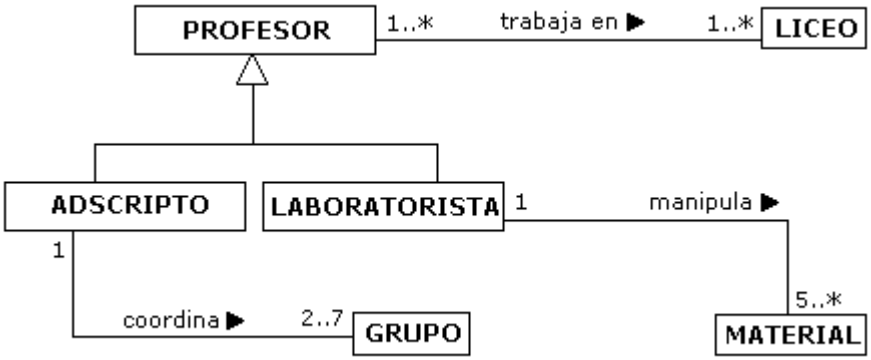


En este diagrama las Clases Adscripto y Laboratorista heredan de Profesor la Asociación *trabaja en*. Eso significa que cualquier Profesor trabaja al menos en un liceo, sin importar si es Adscripto, Laboratorista o un simple Profesor común y corriente. Este diagrama es *correcto*. Consideremos ahora este otro diagrama:



Este diagrama ***no*** es correcto porque *repite* en las clases Derivadas la ***misma*** Asociación con la Clase Liceo. Las Clases Adscripto y Laboratorista ***no*** deben definir nuevamente la Asociación *trabaja en* con la Clase Liceo, porque dicha Asociación es heredada de la Clase Profesor. Es interesante observar que, si bien el diagrama es correcto en su sintaxis, conceptualmente es incorrecto en su significado, porque tiene información redundante. En UML el hecho de que un diagrama no tenga errores de sintaxis no significa necesariamente que sea un diagrama *correcto*.

Dijimos que es incorrecto que una Clase derivada *repita* una Asociación heredada de una Clase base. En cambio, es perfectamente válido que una Clase derivada tenga *sus propias* Asociaciones con otras Clases, sin que los objetos propios de la Clase base participen de dichas Asociaciones. Por ejemplo:



En este ejemplo los Adscriptos coordinan Grupos de Alumnos, y son los *únicos* Objetos de la realidad que pueden hacerlo. No lo hacen los Laboratoristas ni los Profesores comunes y corrientes. Así mismo, son sólo los Laboratoristas quienes manipulan materiales de Laboratorio. No lo hacen ni los Adscriptos ni los Profesores comunes y corrientes.

Clases Abstractas

Para terminar con nuestro estudio del Diagrama de Clases Conceptual introduciremos un concepto muy importante relacionado con la Herencia: El concepto de *Clase Abstracta*. Una Clase Abstracta es una que ***no*** posee instancias (Objetos que pertenezcan a ella propiamente) pero que representa un concepto fundamental a la realidad del problema que debe ser representado. Las Clases Abstractas generalmente se introducen al aplicar la técnica de *generalización* usada para detectar la presencia de Herencia. Esta técnica consiste en detectar Clases de la realidad que tienen características en común y agrupar dichas características en una nueva Clase base a todas ellas, generando así una jerarquía de Herencia.

Por ejemplo, consideremos una realidad referente a un zoológico. En el zoológico tenemos por ejemplo Elefantes, interesando de ellos su nombre, edad, peso y comida recomendada. También tenemos por ejemplo Tigres, interesando de ellos su nombre, edad, color de pelo y país de origen. Si miramos a los Elefantes y a los Tigres podemos observar que tienen dos cosas en común. La primera es que ambos son *Animales*. La segunda es que comparten los atributos nombre y edad. En el zoológico existen más animales pero a los efectos de este ejemplo, sólo consideraremos Elefantes y Tigres.

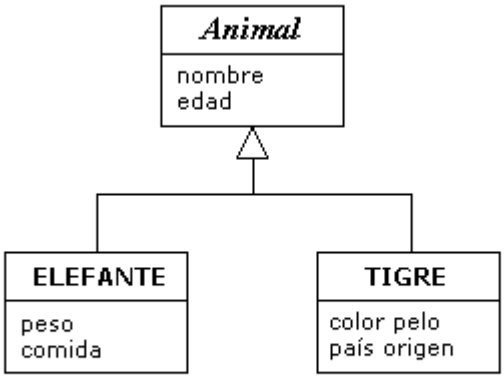
A partir de las consideraciones anteriores introduciremos una nueva Clase llamada *Animal*, en la cual englobaremos los atributos comunes a Tigres y Elefantes. A partir de ahora, Elefante y Tigre serán Clases derivadas de la Clase *Animal*. En resumen, tendremos las siguientes Clases:

- *Animal*, con los atributos nombre y edad
- Elefante (derivada de Animal) con los atributos peso y comida recomendada (además de los heredados)
- Tigre (derivada de Animal) con los atributos color de pelo y país de origen (además de los heredados)

Observemos que en la realidad del zoológico existirán Objetos que pertenezcan a la Clase Tigre así como Objetos que pertenezcan a la Clase Elefante. Sin embargo, ***no*** existirán Objetos que pertenezcan a la Clase *Animal* y que no pertenezcan a ninguna Clase derivada de *Animal*. Decimos entonces que *Animal* es una Clase Abstracta. No puede ser instanciada, pero tiene Clases Derivadas que sí pueden ser instanciadas.

Nótese la diferencia entre Profesores y *Animales*. En la realidad presentada en el ejemplo del liceo podían existir Profesores comunes y corrientes que no fueran Adscriptos ni Laboratoristas. Por eso es que Profesor, si bien era una Clase base, no era Abstracta. *Animal* también es una Clase base, pero no pueden existir animales que no sean Tigres, Elefantes o integrantes de cualquier otra Clase derivada de *Animal*. Por esta razón es que *Animal* sí es una Clase Abstracta, ya que no pueden existir "animales comunes y corrientes". La palabra *Abstracta* de la definición del Concepto de Clase Abstracta surge simplemente del hecho de que representa un concepto de la realidad que es justamente *abstracto*, es decir intangible o conceptual, pero no físico o real.

La forma de indicar en UML que una Clase es abstracta es poniendo el nombre de la Clase en letra *cursiva*. Eso sirve para saber que no existirán Objetos de dicha Clase propiamente, sino de sus Clases derivadas no abstractas. En el ejemplo del zoológico nos referimos a Elefante, Tigre y cualquier otro animal del zoológico que más tarde se nos pueda ocurrir.



Nótese que el nombre de la Clase *Animal* aparece escrito en *cursiva* por tratarse de una Clase Abstracta. Los nombres de las Clases Elefante y Tigre no están en cursiva porque son Clases comunes y corrientes. Para terminar, resta decir que todas las consideraciones estudiadas en el tema anterior (Herencia) siguen valiendo independientemente de que se trabaje con Clases Abstractas o no.