PROGRAMACION ORIENTADA A OBJETOS

UNIDAD 1

Características:

La programación orientada a objetos encara la resolución de cada problema desde la óptica del objeto.

El objeto combina los datos (atributos del objeto) con los procedimientos u operaciones (métodos) que actúan sobre dichos datos.

Los objetos interactúan entre sí enviando mensajes.

El principio fundamental del paradigma de programación orientada a objetos es construir un sistema de software en base a las entidades de un modelo elaborado a partir de un proceso de abstracción y clasificación.

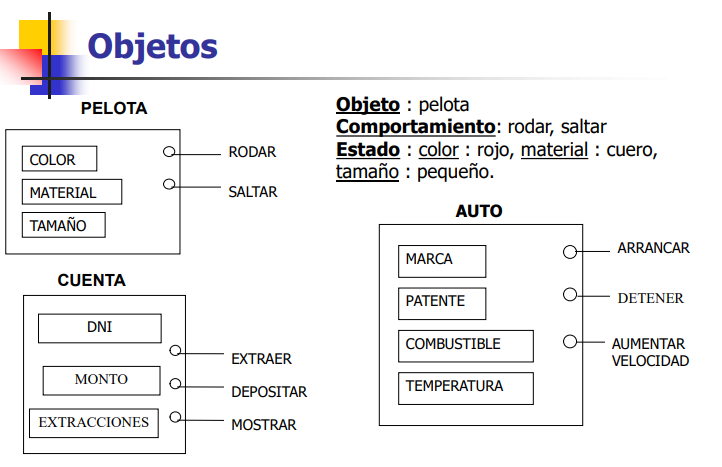
Para hacer buena POO hay que desarrollar todo el sistema utilizando el paradigma, empezando por un análisis y un diseño orientados a objetos. En general, el desarrollo de software implica, desde una visión orientada a objetos, una serie de etapas para hacer: requerimientos, análisis, diseño, implementación y prueba.

Elementos Básicos:

Objetos: Un objeto es una unidad atómica que encapsula estado y comportamiento. La encapsulación en un objeto permite una alta cohesión y un bajo acoplamiento. Los objetos son entidades que tienen atributos (datos) y comportamiento particular (procedimientos)

Atributos de un objeto: Los atributos describen la abstracción de características individuales que posee un objeto.

Comportamientos: Los comportamientos de un objeto representan las operaciones que pueden ser realizadas por un objeto.



El estado agrupa los valores instantáneos de todos los atributos de un objeto. El estado evoluciona con el tiempo.

El comportamiento describe las acciones y reacciones de ese objeto.

Las acciones u operaciones de un objeto se desencadenan como consecuencia de un estímulo externo, representado en forma de un mensaje enviado por otro objeto. El estado y el comportamiento están relacionados.

La identidad permite distinguir los objetos de forma no ambigua, independientemente de su estado. Esto permite distinguir dos objetos en los que todos los valores de los atributos son idénticos.

CLASES:

Una clase es una descripción de un conjunto de objetos, ya que consta de comportamientos y atributos que resumen las características comunes del conjunto.

Una clase abstrae las características de un conjunto de objetos con comportamientos similares.

La encapsulación de una clase permite la cohesión y presenta distintas ventajas básicas:

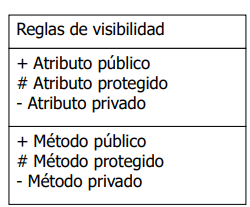
\* Se protegen los datos de accesos indebidos.

\*El acoplamiento entre las clases se disminuye.

\*Favorece el modularidad y el mantenimiento.

Cada objeto es instancia de una clase.

Los atributos de una clase no deberían ser manipulables directamente por el resto de objetos, no obstante, existen distintos niveles de encapsulación también llamados niveles de visibilidad.



Clases – Métodos y Mensajes

Los objetos tienen la posibilidad de actuar. La acción sucede cuando un objeto recibe un mensaje, que es una solicitud que pide al objeto que se comporte de manera determinada.

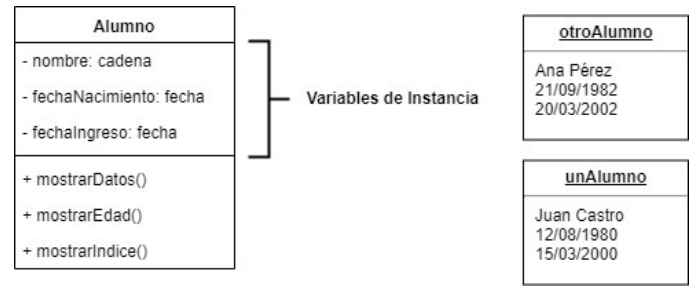
Cada objeto recibe, interpreta y responde a mensajes enviados por otros objetos.

Los comportamientos u operaciones que caracterizan un conjunto de objetos residen en la clase y se llaman métodos.

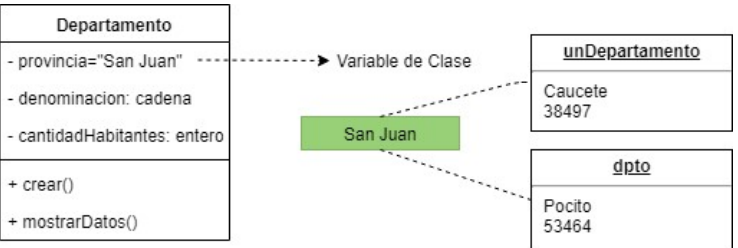
Los métodos son el código que se ejecuta para responder a un mensaje, y el mensaje es la llamada o invocación a un método.

Clases – variables de clases y de instancia

Variables de instancia: Las variables de instancia o miembros dato se usan para guardar los atributos de un objeto particular.

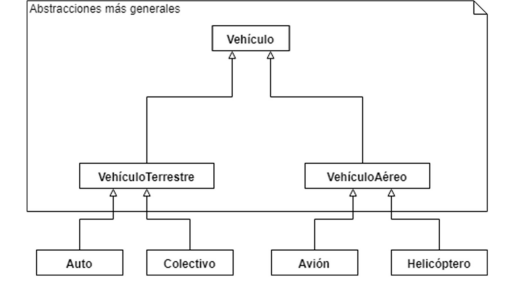


Variables de clase: son aquellos atributos que tienen el mismo valor para cada objeto de la clase. Si el valor de la variable de clase es cambiado para una instancia, el mismo cambia para todas las instancias de la clase y subclase

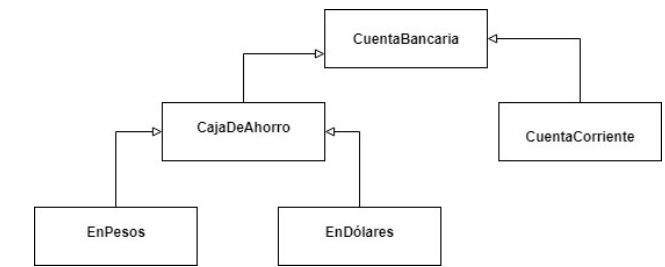


Herencia:

La generalización consiste en factorizar los elementos comunes (atributos, métodos y restricciones) de un conjunto de clases en una clase más general llamada superclase.



La especialización permite capturar las particularidades de un conjunto de objetos no discriminadas por las clases ya identificadas.

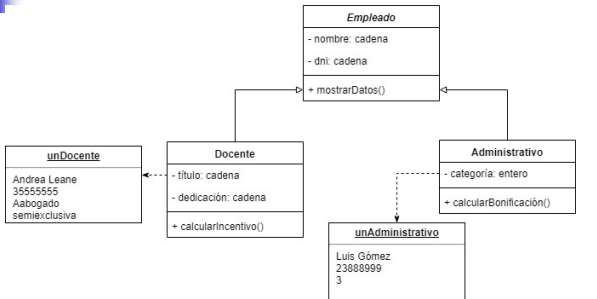


Las clases se ordenan según una jerarquía, una superclase es una abstracción de sus subclases.

Una subclase identifica el comportamiento de un conjunto de objetos que hereda las características de la clase padre y adicionan algunas específicas que ésta no posee.

Clase Abstracta: Una clase es abstracta cuando no existe un objeto que sea instancia directa de ella, pero sí existe una subclase de ella que es instanciable. Generalmente se utilizan para resumir los comportamientos comunes a un conjunto de subclases.

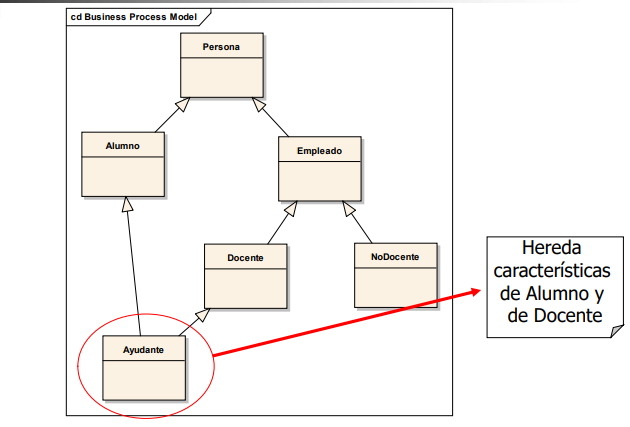
Clase Concreta: Una clase concreta es aquella que es instanciable, es decir que existe al menos un objeto de esa clase.



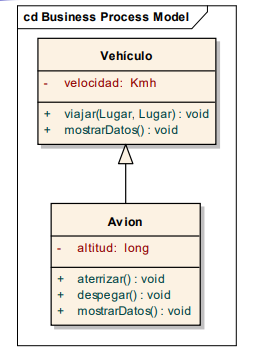
La clase empleado es una clase abstracta, y las clases docentes y administrativos son una clase concreta.

Tipos de herencia:

Herencia múltiple:



Herencia simple:



Conclusión: podemos decir que, dentro de una jerarquía de clases, la herencia propaga las características de la clase superior en sus clases descendientes, de modo que varias clases pueden compartir una misma descripción.

Relación entre clases:

Las relaciones entre las clases expresan cómo se comunican los objetos de esas clases entre sí. Las relaciones muestran el acoplamiento de las clases. (Acoplamiento entre clases es el número de clases con las que una clase concreta está relacionada. Una clase está acoplada si sus objetos lo están. Un objeto está acoplado con otro si uno de ellos actúa sobre el otro.).

Según el tipo de acoplamiento que presentan las clases podemos distinguir distintos tipos de relaciones.

Asociación.

Agregación.

Generalización / Especialización / Herencia.

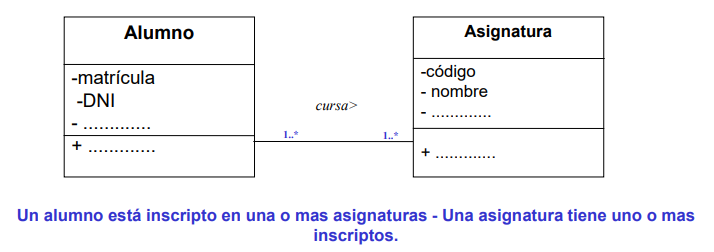
Asociación:

Una asociación es una conexión entre dos clases; refleja una conexión que existe en el ámbito de una aplicación.

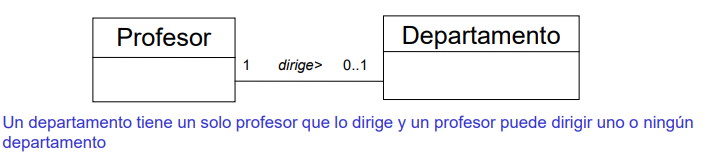
La comunicación puede ser tanto uni como bi-direccional.

Una asociación puede verse como una abstracción de los vínculos que existen entre objetos instancias de las clases asociadas.

Ejemplo 1:



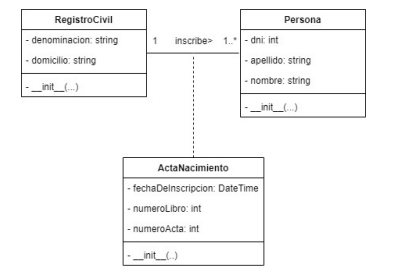
Ejemplo 2:



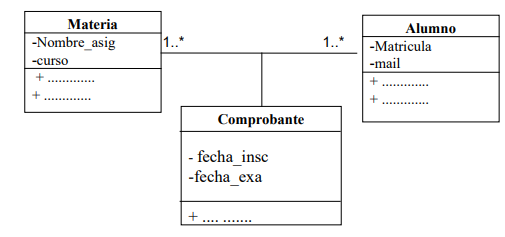
Existen situaciones especiales en las que es necesaria agregar información que es propia de la relación y no de las clases. Para esto se trabaja con dos conceptos distintos según sea el caso:

Clase asociación (propiamente dicha):

Una dupla de objetos, instancias de cada una de las clases que participan en la asociación, se relaciona con una única instancia de la clase asociación. No importa la multiplicidad en ambos extremos.



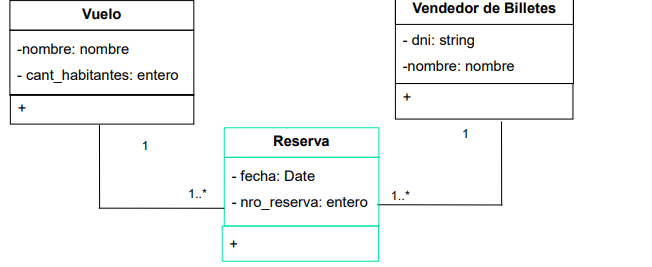
si bien, hay una asociación entre Registro Civil y Persona, los datos propios del acta de nacimiento, no son datos inherentes al Registro Civil, ni a la Persona, esta información debe ser almacenada como atributos de una clase asociación.



Comprobante es una clase Asociación: Para una dupla de objetos (materia, alumno) existe un único objeto asociado a la clase comprobante.

Clase que modela la asociación:

Para una dupla de objetos (vuelo, vendedor de billetes) existe más de objeto asociado a la clase Reserva.



Para una instancia de Vuelo y una instancia de Vendedor de Billetes, existen más de un objeto de la clase Reserva. Es decir, podemos tener múltiples reservas por vuelo y vendedor. En este caso no podemos definir a Reserva como una clase asociación, sino que existe una relación de asociación entre Vuelo y Vendedor de Billetes que se deriva o modela a través de Reserva.

Agregación:

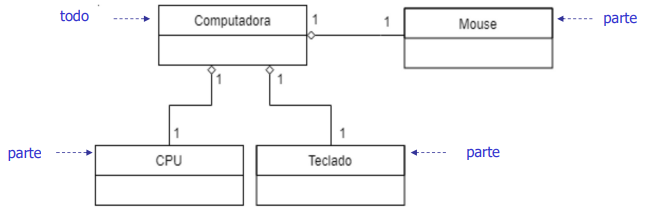
La agregación consiste en definir como atributos de una clase a objetos de otras clases ya definidas.

La agregación es una relación no simétrica (todo/parte) en la que una de las clases cumple un papel predominante respecto de la otra.

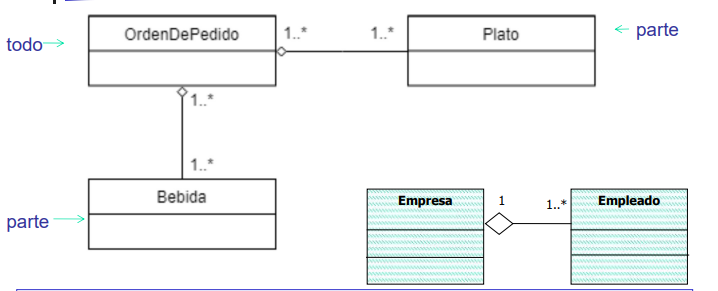
Agregación (propiamente dicha):

Este tipo de relación se presenta en aplicaciones en las cuales un objeto contiene como partes a objetos de otras clases, pero de tal modo que la destrucción del objeto continente no implica la destrucción de sus partes. Esto es, el objeto continente o contenedor incluye referencias a objetos de otras clases y que los objetos contenidos pueden existir independientemente del objeto contenedor.

Ejemplo 1:



Ejemplo 2:



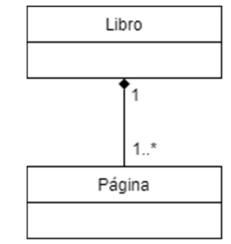
En el caso de las agregaciones propiamente dichas, no hay restricciones en la multiplicidad del agregado u objeto continente.

Composición:

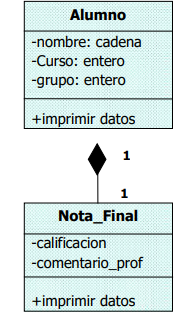
La forma más simple de reutilizar una clase es simplemente haciendo una nueva clase que la contenga. Esta técnica se llama composición.

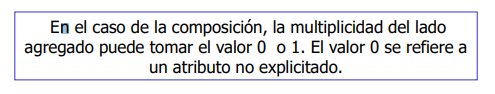
Un objeto de una clase contiene como partes a objetos de otras clases y estas partes están físicamente contenidas por el agregado. Los objetos agregados no tienen sentido fuera del objeto resultante. La destrucción del objeto continente implica la destrucción de sus partes.

Ejemplo 1:



Ejemplo 2:





Algunos criterios para detectar agregaciones:

Los objetos de una clase están subordinados a los objetos de otra clase.

Una acción sobre objetos de una clase implica una acción sobre objetos de otra clase.

Los valores de los atributos de un objeto de una clase se propagan en los valores de los atributos de otro objeto de otra clase.

Conceptos Claves:

Encapsulamiento:

Término formal que describe al conjunto de métodos y de datos de un objeto de manera tal que el acceso a los datos se permite solamente a través de los métodos propios de la clase a la que pertenece el objeto.

La comunicación entre los distintos objetos se realiza solamente a través de mensajes explícitos.

Abstracción: La orientación a objetos fomenta que los programadores y usuarios piensen en las aplicaciones en términos abstractos. A partir de un conjunto de objetos, se piensa en los comportamientos comunes de los mismos para situarlos en superclases, las cuales constituyen un depósito para elementos comunes y reutilizables.

Polimorfismo: Capacidad que tienen objetos de clases diferentes, relacionados mediante la herencia, a responder de forma distinta a una misma llamada de un método.

Persistencia: designa la capacidad de un objeto de trascender el tiempo o el espacio.

un objeto persistente conserva su estado en un sistema de almacenamiento permanente. el objeto puede ser reconstruido por otro proceso y se comportará exactamente como en el proceso inicial.

Unidad 2: