TEMA IV

UML (MODELIZACIÓN)

DIAGRAMA DE CLASES

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema así como sus atributos y comportamientos.

Un diagrama de clases está compuesto por los siguientes elementos:

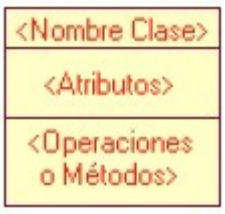
Clase: atributos, métodos.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Elementos

1. Clases

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:



En donde:

- Nombre de la Clase: primer carácter en mayúsculas.
- Atributos : caracterizan a la Clase. Son un conjunto de variables de los que hay que definir nombre y tipo.
  El tipo puede ser un tipo simple u otra clase

Además hay que indicar su visibilidad:

- public: se accede desde cualquier clase y cualquier parte del programa (+)
- private: sólo se accede desde la clase (-)
- protected: se accede desde la clase y cualquier clase derivada(#)
- package: se accede desde cualquier clase que pertenezca al mismo paquete

- Métodos u operaciones: son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected, public).

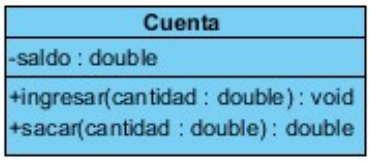
	clase	paquete	subclase	resto
Public	s	s	s	s
Protected	s	s	s	n
package	s	s	n	n
Private	s	n	n	n

Ejemplo:

Una Cuenta Corriente que posee como característica: saldo

Puede realizar las operaciones de: ingresar , sacar

El diseño asociado es:



2. Relaciones entre Clases:

Ahora ya definido el concepto de Clase, es necesario explicar cómo se pueden interrelacionar entre ellas. Una relación es una conexión que incluimos en el diagrama cuándo aparece algún tipo de relación entre dos clases.

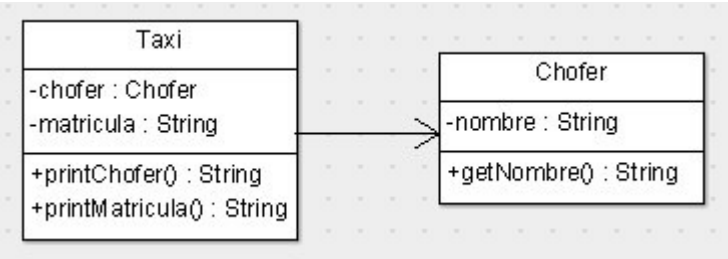
Las relaciones se caracterizan por su cardinalidad, que representa cuantos objetos de una clase pueden estar involucrados en la relación con objetos de otra. En una relación hay dos cardinales, uno para cada extremos de la relación y pueden tener los siguientes valores:

- **uno a uno:1..1**
- **uno o muchos**: 1..* (1..n)
- **cero a uno**:0..1
- **0 o muchos**: 0..* (0..n)
- **número fijo**: n..m (n,m denotan el número).

Tendremos tres tipos de relaciones: asociación, dependencia, herencia

2.1 Asociación: 

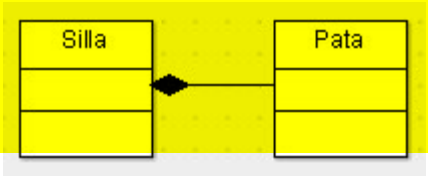
La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí. Un objeto u objetos de una clase son atributos de otra clase.



Codifiquemos estas clases

Para especificar más, casos particulares de una asociación :

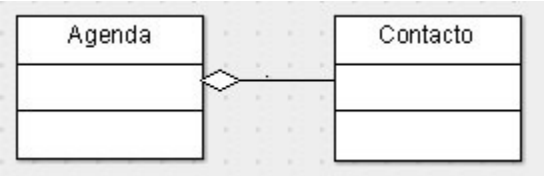
Composición: El tiempo de vida del objeto incluido está condicionado por el tiempo de vida del que lo incluye, ya que se crea (new) dentro de la clase que lo incluye. Se representa con un rombo relleno.



Cuando se destruye el objeto silla se destruyen los objetos patas asociados.

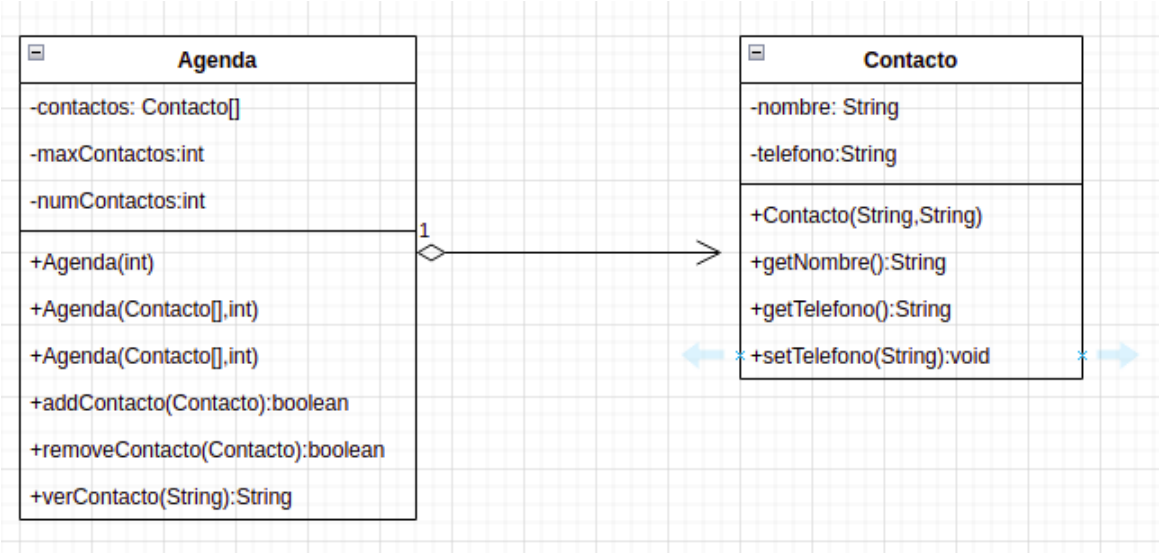
```
/*Clase Silla*/
class Silla{
        private Patas p[];
        public Silla(){
                p=new Pata[4];
                for(i=0;i<3;i++)
                        p[i]=new Pata();
        }
}
/* Clase Pata*/
Class Pata{
        Public Pata(){
        }
}
```

Agregación: El tiempo de vida del objeto incluido es independiente del que lo incluye, es decir no lo gestiona la clase que lo incluye. Se representa con un rombo transparente.

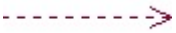


Cuando se destruye el objeto Agenda no se destruyen los objetos Contacto asociados, ya que éstos se crean (new) fuera de la clase Agenda.

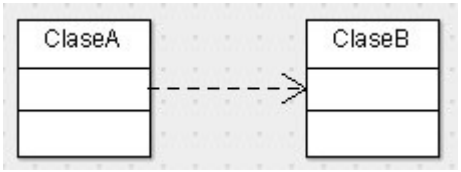
Ejemplo: Diagrama de clases y código java para un programa que almacena 5 contactos. Cada contacto lleva nombre y teléfono. En la agenda podemos añadir contacto, eliminar contacto y ver un teléfono dado un nombre. Para un contacto podemos modificar su teléfono.



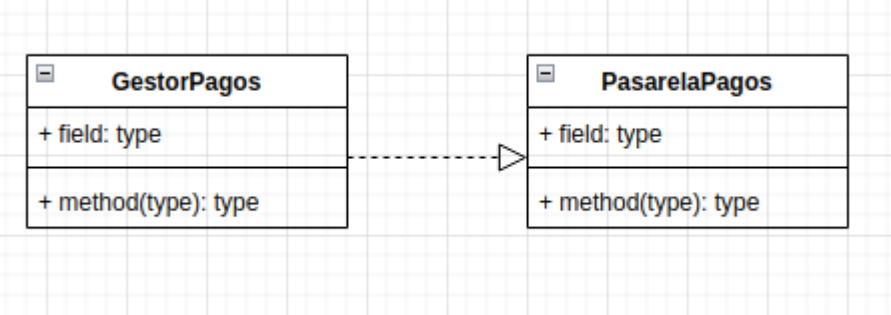
Codificamos el programa que se ajusta a este diseño.

2. Dependencia o Instanciación (uso): 

Es una relación de uso entre dos clases, una usa a la otra.



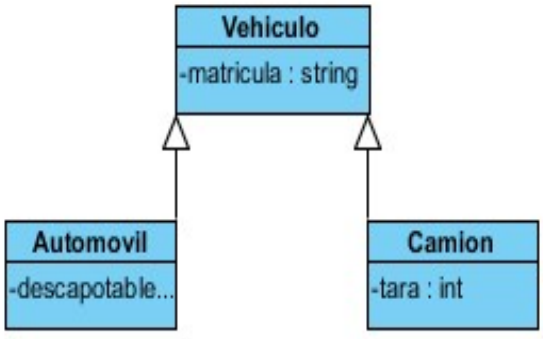
- La A usa a la B.
- La A depende de la B: un cambio en la clase B puede afectar a la A.
- La A conoce la existencia de B, pero la B no tiene por qué conocer la existencia de A.



Ejemplo: Usaremos este tipo de relación cuándo veamos bases de datos

3. Herencia(Especialización/Generalización)

Indica que una subclase hereda los métodos y atributos especificados por una Super Clase, la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Super Clase (public y protected)



En la figura se especifica que Automovil y Camión heredan de Vehículo, es decir, Automovil posee las Características de Vehículo (matricula) y además posee algo particular que es descapotable, Camión también hereda las características de Vehiculo (matricula) pero posee como particularidad propia la tara.

Ejemplo: Diagrama de clase, sin codificar, para una sucursal bancaria de la que guardamos nombre y la localización. En dicha sucursal hay empleados, para los que guardamos nombre y apellidos, dni, dirección y teléfono así como un identificador que

les asocia el banco. Para los clientes almacenamos nombre y apellidos, dni, dirección, teléfono y las cuentas que tiene en el banco. De cada cuenta guardamos un identificados y su saldo. Podemos dar de alta una cuenta o cancelar cuenta.