

DIAGRAMAS DE CLASE EN UML

Diseño y Programación Orientada a Objetos

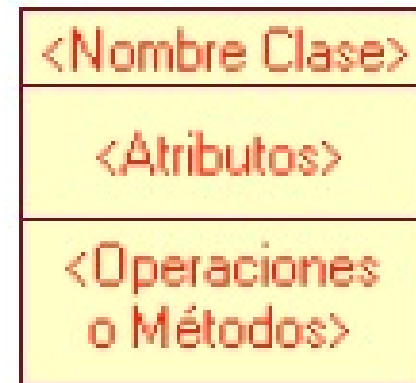
<http://www.dsi.uclm.es/asignaturas/42579>

Curso 06/07

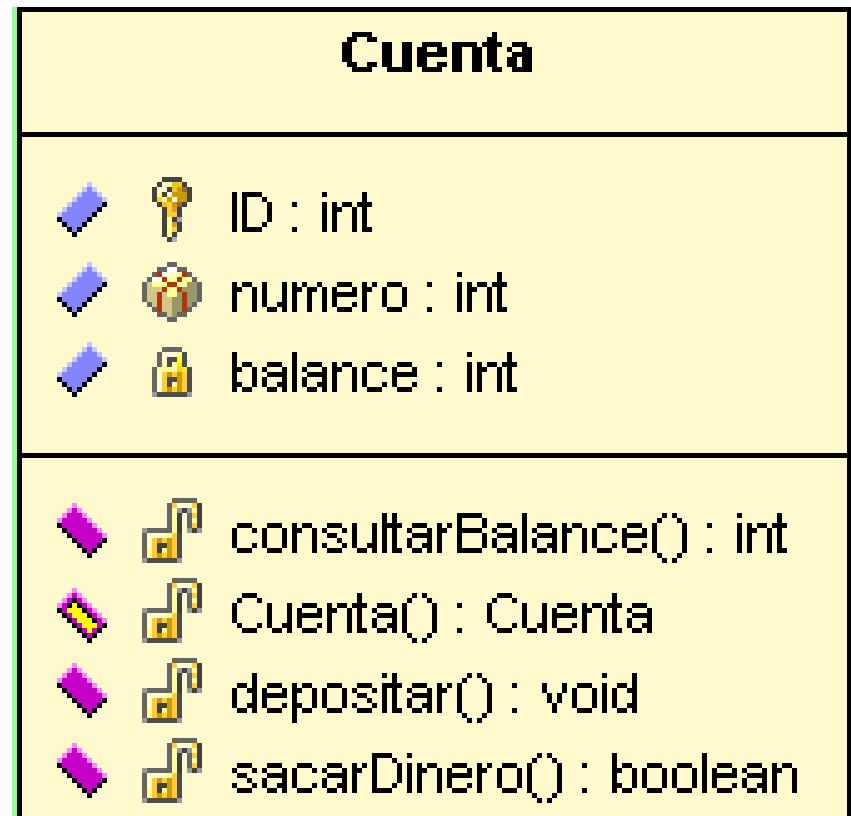


- Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación.
- Un diagrama de clases esta compuesto por los siguientes elementos:
 - Clase: atributos, métodos y visibilidad.
 - Relaciones: Herencia, Composición, Agregación, Asociación y Uso.





- 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 Coche, una Cuenta Corriente, etc.).
- En UML, una clase es representada por un rectángulo que posee tres divisiones:



- Una Cuenta Corriente que posee como características:
 - Balance
 - Numero
 - ID
- Puede realizar las operaciones de:
 - Depositar
 - sacarDinero
 - y consultarBalance





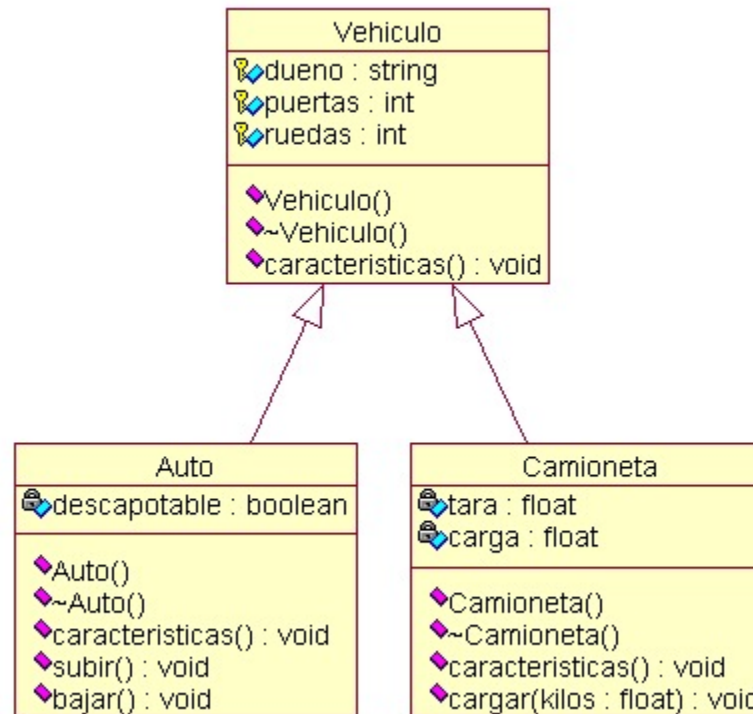
- Los llamaremos miembros de la clase. Sus accesos pueden ser de cuatro diferentes tipos que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:
 - **public** (+, ): Indica que el atributo será visible tanto dentro como fuera de la clase ya sea fuera o dentro del paquete.
 - **private** (-, ): Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos pueden manipularlo).
 - **protected** (#, ): Será accesible desde las clases que se encuentren en el mismo paquete así como en todos sus subclases. Aunque las subclases que se encuentren fuera del paquete sus objetos o instancias no los podrán manipular.
 - **Friendly** (): Indica que el atributo será accesible desde cualquier otra clase que se encuentre en el mismo paquete.

- Ahora ya definido el concepto de Clase, es necesario explicar como se pueden interrelacionar dos o más clases (cada uno con características y objetivos diferentes).
- Antes es necesario explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:
 - **uno o muchos:** 1..* (1..n)
 - **0 o muchos:** 0..* (0..n)
 - **número fijo:** m (m denota el número).

DIAGRAMA DE CLASES EN UML

HERENCIA Especialización/Generalización

- Indica que una subclase hereda los métodos y atributos especificados por una superclase, de esta forma la subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la superclase (public y protected), ejemplo:

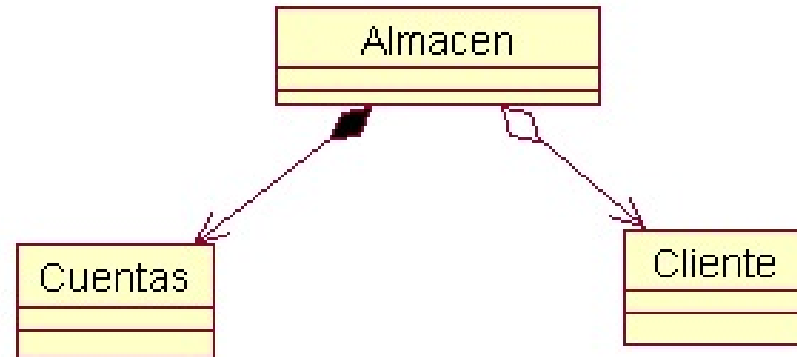




- Para modelar objetos complejos, no es suficiente con los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:
 - **Por Valor:** Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido esta condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada **Composición** (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").
 - **Por Referencia:** Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada **Agregación** (el objeto base utiliza al incluido para su funcionamiento).

- Ejemplo, En donde se destaca que:

- Un Almacén posee Clientes y Cuentas (los rombos van en el objeto que posee las referencias).
- Cuando se destruye el Objeto Almacén también son destruidos los objetos Cuenta asociados, en cambio no son afectados los objetos Cliente asociados.
- La composición (por Valor) se destaca por un rombo relleno.
- La agregación (por Referencia) se destaca por un rombo transparente.
- La flecha en este tipo de relación indica la navegabilidad del objeto referenciado. Cuando no existe este tipo de particularidad la flecha se elimina.





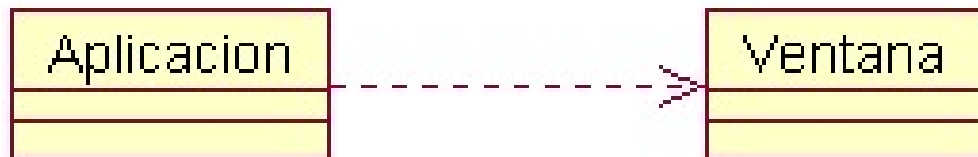
- La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.
- Ejemplo:



- Un cliente puede tener asociadas muchas Ordenes de Compra, en cambio una orden de compra solo puede tener asociado un cliente.

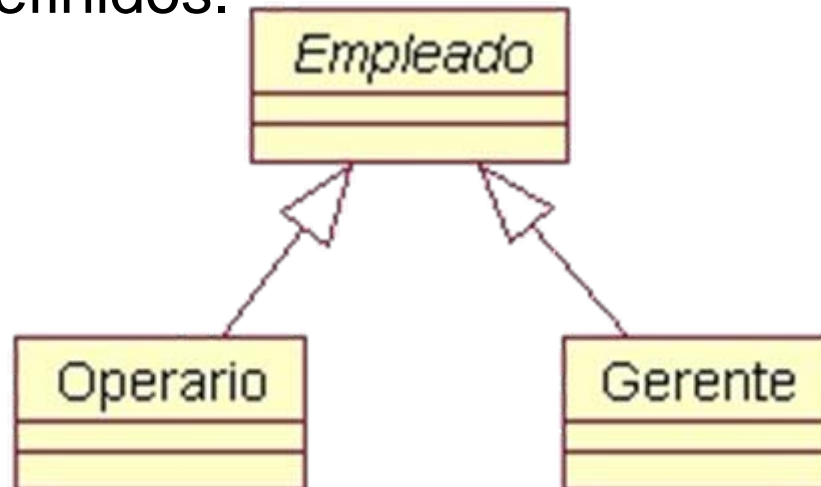


- Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro Objeto/clase). Se denota por una flecha punteada.
- El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo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

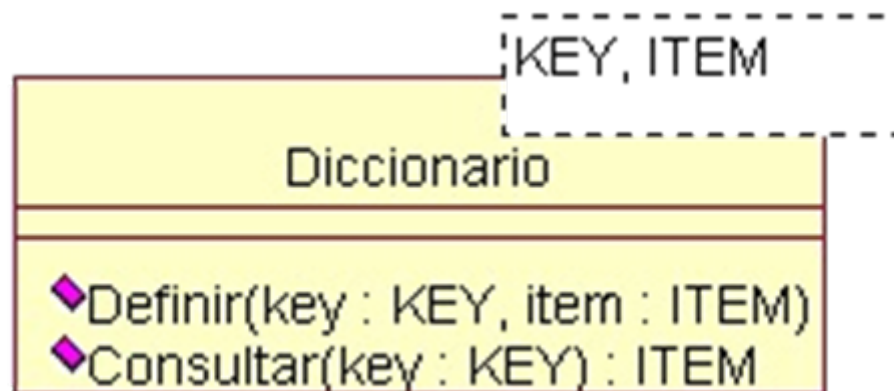


- Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

- Una **clase abstracta** se denota con el nombre de la clase y de los métodos con letra cursiva. Esto indica que la clase definida no puede ser instanciada pues posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). La única forma de utilizarla es definiendo subclases, que implementan los métodos abstractos definidos.



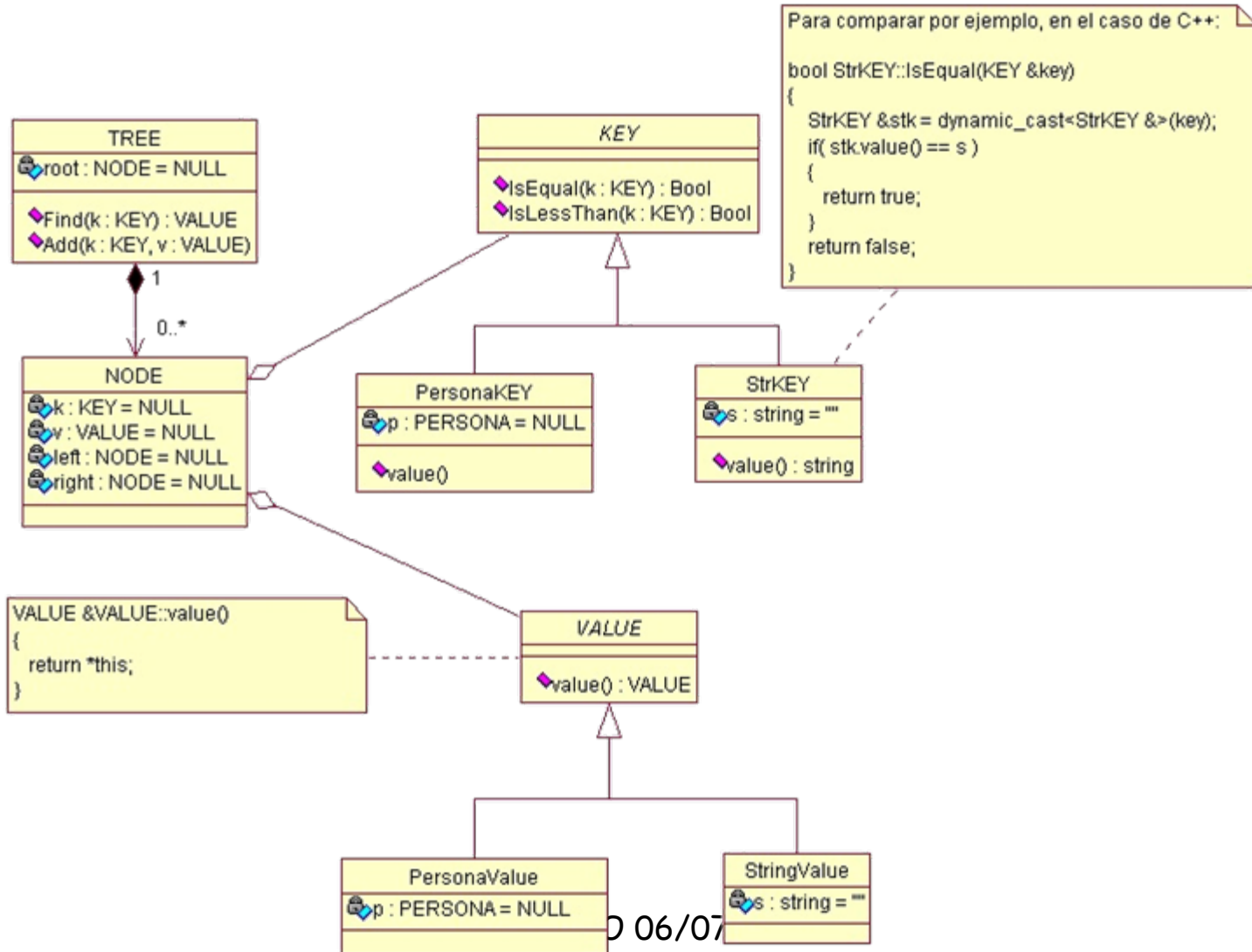
- Una **clase parametrizada** se denota con un subcuadro en el extremo superior de la clase, en donde se especifican los parámetros que deben ser pasados a la clase para que esta pueda ser instanciada. El ejemplo más típico es el caso de un Diccionario en donde una llave o palabra tiene asociado un significado, pero en este caso las llaves y elementos pueden ser genéricos.



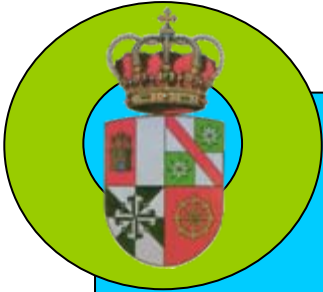
- Supongamos que tenemos el caso de un diccionario implementado mediante un árbol binario, en donde cada nodo posee:
 - **key**: Variable por la cual se realiza la búsqueda, puede ser genérica.
 - **item**: Contenido a almacenar en el diccionario asociado a "key", cuyo tipo también puede ser genérico.
- Para este caso particular hemos definido un Diccionario para almacenar String y Personas, las cuales pueden funcionar como llaves o como item, solo se mostrarán las relaciones para la implementación del Diccionario:

DIAGRAMA DE CLASES EN UML

EJEMPLO: DICCIONARIO



- Se desea realizar un sistema informático para gestionar las bibliotecas de la universidad de Castilla – La Mancha.
- Un diagrama de secuencia típico es el que nos encontramos a continuación:
- Las personas son alumnos y profesores de la universidad. Los alumnos tienen restricciones a la hora de sacar los recursos de la biblioteca no así los profesores.
- Los recursos pueden ser de tres tipos diferentes libros, revistas y medios electrónicos.
- Los libros estarán formados por capítulos que podrán ser considerados como otro recurso. De la misma forma también las revistas estarán formados por artículos que también tendrán carácter de recurso.
- Los bibliotecarios serán los encargados de la gestión de entrada y salida de los recursos de la biblioteca.



DIAGRAMAS DE CLASE EN UML

Diseño y Programación Orientada a Objetos

<http://www.dsi.uclm.es/asignaturas/42579>

Curso 06/07

