

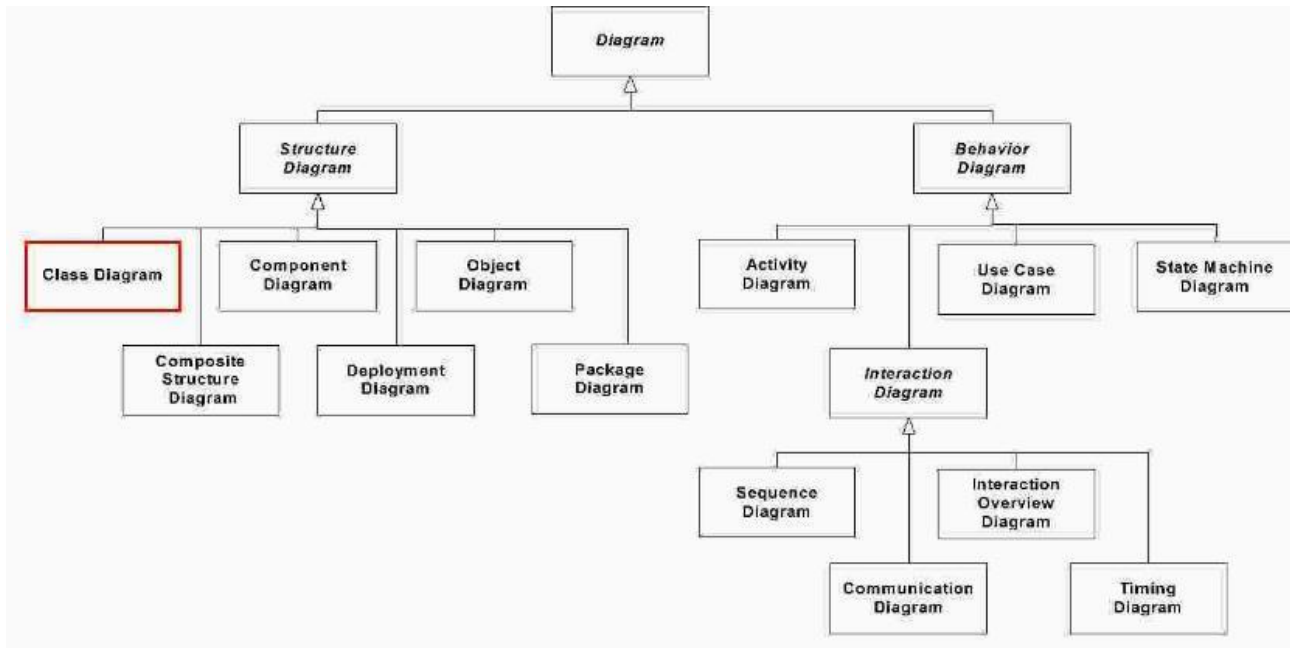
Indice

| | |
|---|---|
| 6 – UML..... | 1 |
| 6.1 – L'icona di classe in UML..... | 2 |
| 6.2 – Attributi e Operazioni..... | 3 |
| 6.3 – L'icona di Oggetto in UML | 4 |
| 6.4 – Relazioni tra classi..... | 5 |
| 6.5 – Esempio completo di diagramma delle classi..... | 8 |

6 – UML

Il linguaggio **UML** (*Unified Modeling Language*, linguaggio di modellazione unificato) è un modello grafico corredato da annotazioni di testo, usato per definire i requisiti funzionali di un sistema. Viene usato durante la specifica dei requisiti, quando viene stilato il documento dei requisiti.

Nella visione d'insieme, la UML si trova:



6.1 – L'icona di classe in UML

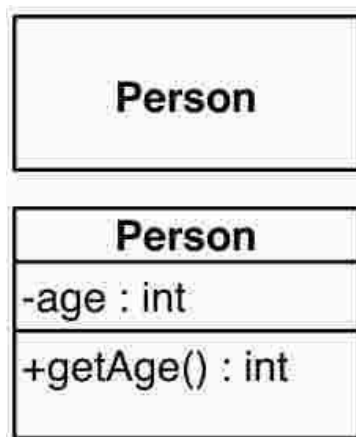
Una **classe** ha una rappresentazione grafica in forma di un rettangolo diviso in tre parti.

Le **classi** possono avere fino a **3 slot**:

- **Nome** e l'eventuale **stereotipo** (**obbligatorio** l'uso del UpperCamelCase);
- **Attributi** (opzionale l'uso del lowerCamelCase);
- **Operazioni** (opzionale l'uso del lowerCamelCase).

La stessa classe può apparire con diverse quantità di ornamenti in diagrammi diversi.

Un **esempio grafico** di una classe in UML:



6.2 – Attributi e Operazioni

C'è differenza tra **attributo** e **operazioni**.

L'**attributo** è strutturato nel seguente modo:

visibilità nome molteplicità: tipo = valoreIniziale

Al contrario, un'**operazione** è strutturata nel seguente modo e ritorna un valore:

*visibilità nome(nomeParametro: tipoParametro, ...):
tipeRestituito*

In entrambi **solo il nome è obbligatorio**.

Le **classi di analisi** solitamente contengono solo quelli più importanti, ovvero quelli che risultano evidenti dall'analisi del dominio, e spesso viene specificato solo il nome.

Assegnare un valore iniziale in una classe di analisi può evidenziare i vincoli di un problema.

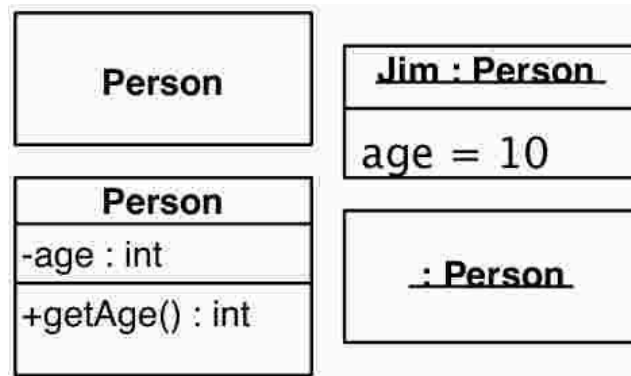
Le **classi di progettazione** forniscono una specifica completa (implementabile) della classe e dei suoi attributi.

Qui di seguito vengono elencati alcuni **tipi di visibilità**:

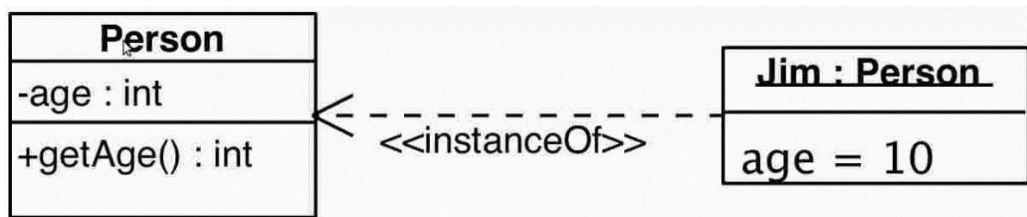
| | | |
|---|------------------|---|
| + | <i>public</i> | Ogni elemento che può accedere alla classe può anche accedere a ogni suo membro con visibilità pubblica. |
| – | <i>private</i> | Solo le operazioni della classe possono accedere ai membri con visibilità privata. |
| # | <i>protected</i> | Solo le operazioni appartenenti alla classe o ai suoi discendenti possono accedere ai membri con visibilità protetta. |
| ~ | <i>package</i> | Ogni elemento nello stesso package della classe (o suo sottopackage annidato) può accedere ai membri della classe con visibilità package. |

6.3 – L'icona di Oggetto in UML

Le istanze delle classi (**oggetti**) hanno una notazione molto simile a quella delle classi. Il titolo degli oggetti è sottolineato così da distinguerle dalle classi (come in figura) e sono formate con *nome: classe*, in cui il *nome* è opzionale.







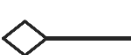

Tuttavia, nonostante il modo per distinguerle, è possibile creare una relazione tra Classe e Oggetto tramite una dipendenza con stereotipo chiamata *instanceOf*:



6.4 – Relazioni tra classi

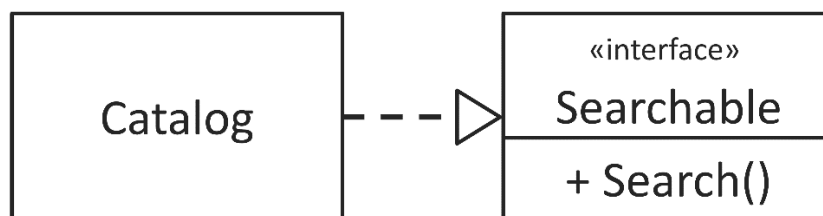
La linea rappresentata nell'esempio precedente, indica una relazione di tipo "dipendenza". Esistono altre **relazioni grafiche**: **generalizzazione**, **realizzazione**, **associazione**, **dipendenza**, **aggregazione**, **composizione**.

Vengono rappresentate così:

| | |
|------------------|--|
| Generalizzazione |  |
| Realizzazione |  |
| Associazione |  |
| Dipendenza |  |
| Aggregazione |  |
| Composizione |  |

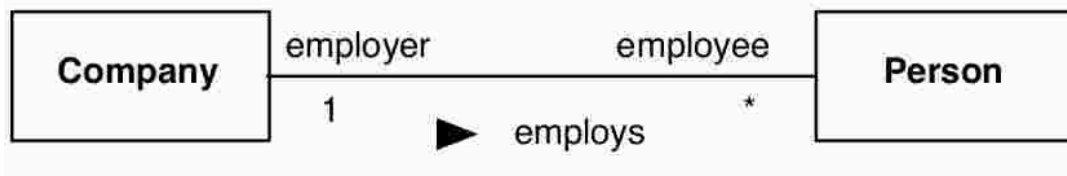
La **generalizzazione** è una relazione tassonomica tra un elemento più generale e uno che lo specifica. In questo caso, la freccia parte dall'elemento specifico e punta verso quello più generale. Una relazione di questo tipo implica l'ereditarietà in UML. Infine, tra tutte le relazioni, questa è considerata la **più forte e vincolante**.

La **realizzazione** rappresenta una relazione semantica in cui il fornitore presenta una specifica, mentre il cliente la realizza (implementandola ed eseguendola). L'esempio classico di realizzazione è quello in cui il fornitore è un'interfaccia, e il cliente è la classe che la implementa:



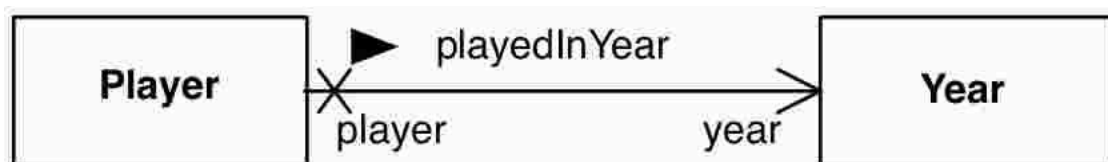
L'**associazione** si tratta del tipo di relazione più generico. Infatti, indica solo l'esistenza di collegamenti, chiamati *link*, tra le istanze delle classi. Essa rappresenta l'abilità di un'istanza di mandare messaggi a un'altra istanza. Ha la possibilità di coinvolgere più di due classi e la stessa classe più di una volta. Tra tutte le relazioni, è anche la **più flessibile** e la **meno vincolante**.

È opzionale, ma è possibile aggiungere un **nome**, un **triangolo direzionale** per aumentare la leggibilità e specificare la direzione in cui leggere l'associazione, i **ruoli**, la **molteplicità**.



La **navigabilità** è una tecnica usata con l'associazione per migliorare la comprensione. La freccia è il simbolo di questa tecnica, mentre la croce indica assenza di navigabilità. Vengono specificati i nomi di quali istanze sono associati e la mancanza di entrambe (freccia e croce) indica che la navigabilità non è stata specificata.

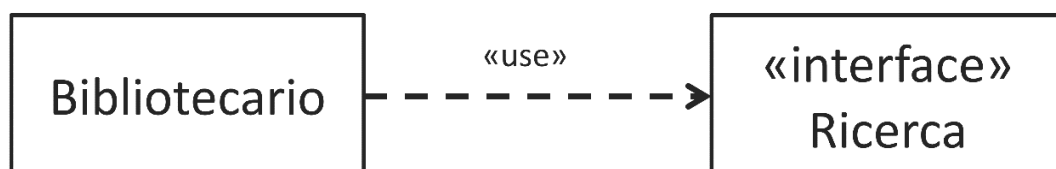
Per esempio, un oggetto di tipo *Player* sa in quali anni ha giocato, un oggetto di tipo *Year* non sa quali giocatori giocarono quell'anno:



Nella pratica non si tende a specificare la navigabilità in ogni estremo di ogni relazione. Solitamente non si usano le croci, l'assenza di frecce indica navigabilità in entrambe le direzioni e una freccia indica navigabilità in quella direzione e assenza di navigabilità nell'altra.

La **dipendenza** è una forma più debole di relazione: tra un cliente ed un fornitore di servizio (esempio tra classi e operazioni). Esistono due **ruoli**: il *supplier* (fornitore) e il *client* (cliente); entrambi possono essere insiemi di elementi. La freccia va dal cliente verso il fornitore e può essere indicato anche il tipo di dipendenza tra i due.

Una dipendenza significa che il cliente richiede il fornitore per la propria specifica o implementazione. Il cliente dipende strutturalmente o semanticamente dal fornitore, e se la specifica del fornitore cambia, può cambiare anche quella del cliente.

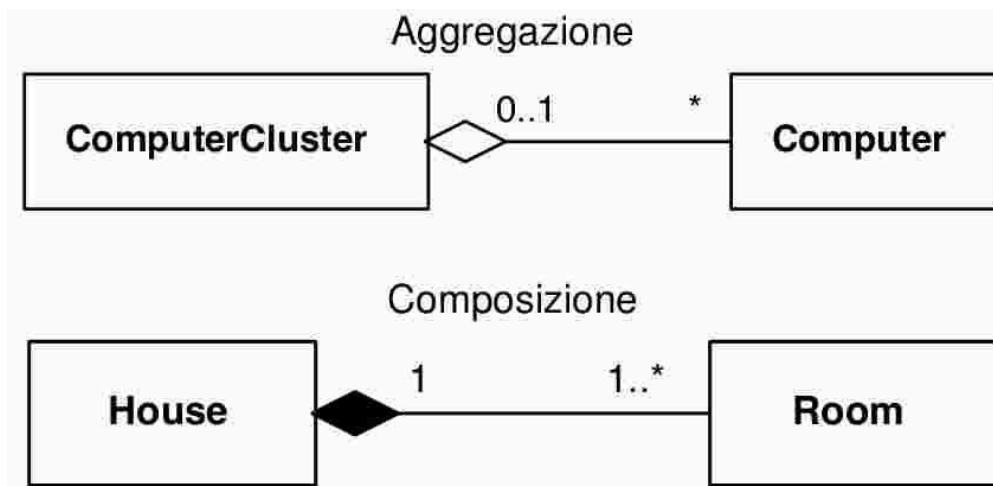


L'**aggregazione** e la **composizione** si trattano di particolari forme di associazione che rappresentano la relazione *whole-part* (tutto-parte) tra un aggregato e le sue parti.

- **Aggregazione**: relazione poco forte.
 - Le parti esistono anche senza il tutto;
 - Le parti possono appartenere a più aggregazioni.
 - Per esempio, i computer e il loro [cluster](#).
- **Composizione**: relazione molto forte.
 - Le parti dipendono dal tutto e non possono esistere al di fuori di esso.
 - Per esempio, le stanze e la casa.

Non è sempre semplice capire quale delle due modelli meglio una situazione, dipende dal contesto.

Le loro **rappresentazioni grafiche** sono rappresentate da linee con un diamante (pieno o vuoto) vicino alla classe che rappresenta l'intero:



Infine, è possibile eseguire una **combinazione** con **aggregazione** o **composizione** con le altre notazioni per le associazioni.

6.5 – Esempio completo di diagramma delle classi

