Azzolini Riccardo 2019-10-14

Class diagram

1 Class diagram

I **class diagram** definiscono la visione statica del sistema:

- classi;
- relazioni tra le classi:
  - *associazione* (uso);
  - *generalizzazione* (ereditarietà);
  - *aggregazione* (contenimento).

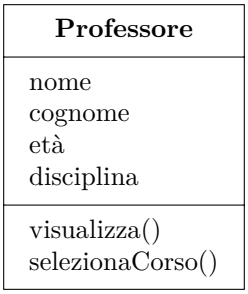
Questo è forse il modello più importante, dato che definisce il **dizionario dei dati**, cioè quali sono le entità rappresentate nel sistema e i dati associati a ciascuna di esse.

2 Classe

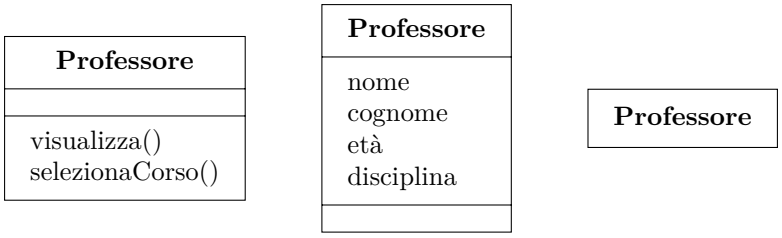
In UML, una classe è composta da tre parti:

- **nome** (che dà anche alcune altre informazioni);
- **attributi** (lo stato);
- **metodi/operazioni** (il comportamento).

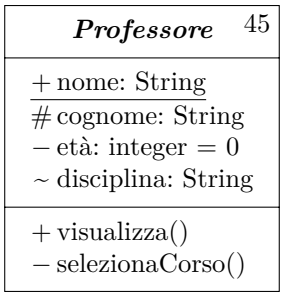
Queste tre parti si rappresentano mediante un rettangolo diviso in altrettante sezioni:



È però possibile omettere la sezione degli attributi e/o quella dei metodi (ma non il nome della classe). Quindi, anche le seguenti sono rappresentazioni valide della stessa classe:



Un esempio di rappresentazione ancora più completa, invece, è:



- Il numero 45 nell'angolo indica la **molteplicità** della classe, cioè quante sue istanze esistono nel sistema. In questo caso è data una quantità esatta, ma si possono anche specificare vincoli più flessibili, definendo un insieme/range di numeri di istanze consentiti (ad esempio 0..5).
- Il nome scritto in corsivo indica che la classe è astratta.
- La sottolineatura indica che l'attributo "nome" è statico.
- È specificato il tipo di ciascun attributo. Si possono indicare in modo simile anche i tipi di eventuali parametri e i tipi restituiti dei metodi.
- Per l'attributo "età" è specificato un valore iniziale.
- I simboli che precedono i nomi di attributi e metodi ne indicano la visibilità:

Simbolo	Visibilità
+	pubblica
#	protetta
−	privata
~	package

3 Principio dell'elisione

In UML, spesso *non è necessario* che un certo costrutto sia completo. Un elemento della sintassi che viene omesso indica infatti un'informazione non specificata. Ciò può essere utile per vari motivi:

- perché l'informazione, almeno per il momento, non è nota (ad esempio, se essa è relativa un aspetto del sistema la cui progettazione non è ancora stata completata);
- perché si vuole analizzare il sistema a un livello di astrazione per il quale l'informazione non è rilevante.

4 Attributi

Un **attributo** è una caratteristica della classe:

- non ha identità propria, cioè rappresenta un valore "semplice" appartenente alle istanze della classe, e non un oggetto complesso;
- deve essere definito in modo preciso (ma vale il principio di elisione).

In pratica, gli attributi sono "nomi che non sono diventati classi". Ad esempio, per memorizzare nome e cognome di una Persona si potrebbe definire un'apposita classe Nominativo (utile soprattutto se questa venisse poi riutilizzata da altre classi), oppure, se si ritenesse che ciò non sia necessario, si potrebbero semplicemente aggiungere due attributi separati alla classe Persona.

La scelta degli attributi dipende strettamente dal dominio applicativo. Ad esempio, i dati di interesse di una Persona potrebbero essere:

- in ambito bancario: nome, cognome, codice fiscale, numero del conto, ecc.
- in ambito medico: nome, cognome, allergie, peso, altezza, ecc.

Ci sono però dei criteri generali da rispettare: ad esempio, i corsi scelti da uno studente *non* devono essere un attributo, perché non dipendono esclusivamente dallo studente, ma anche dall'elenco dei corsi disponibili.

5 Identità degli oggetti

Gli oggetti hanno una loro identità, quindi non è necessario aggiungerla mediante un attributo "id" (come si farebbe, ad esempio, nella progettazione di un database relazionale).

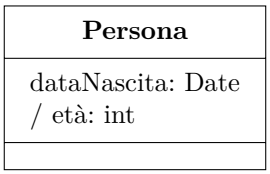
Se, però, l'entità modellata ha un identificatore che esiste nel mondo reale (il codice fiscale di una persona, la targa di un'auto, il numero di matricola di uno studente,

ecc.) ed è rilevante per l'applicazione, allora tale identificatore deve esistere anche come attributo.

6 Attributi derivati

Un attributo si dice **derivato** se è calcolato, o per lo meno calcolabile, in base ai valori di altri attributi.

In UML, un attributo derivato si indica scrivendo il simbolo / prima del suo nome, e riportando la regola per calcolarlo tra parentesi graffe all'esterno della classe (che è, in generale, la notazione usata per i *vincoli*):



{età = oggi − dataNascita}

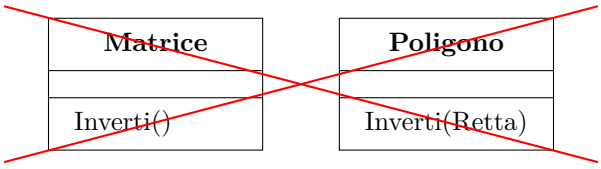
Gli attributi derivati si usano solitamente per dati che variano frequentemente e devono essere sempre aggiornati.

7 Operazioni

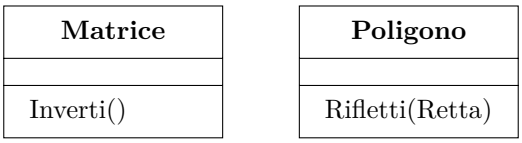
Un'**operazione** è una funzione o una trasformazione applicabile a un'istanza di una classe.

Possono essere definite operazioni con lo stesso nome in classi diverse: tecnicamente, esse sono indipendenti (a meno che le classi non siano legate da una relazione di ereditarietà: in tal caso, l'operazione nella sottoclasse effettua l'override dell'operazione corrispondente nella superclasse), ma è fortemente consigliato che siano almeno di natura simile.

Ad esempio, non è corretto usare il nome Inverti per rappresentare sia l'inversione di una matrice che la riflessione di un poligono rispetto a una retta:



Invece, è meglio chiamare le due operazioni rispettivamente Inverti e Rifletti:



8 Relazioni

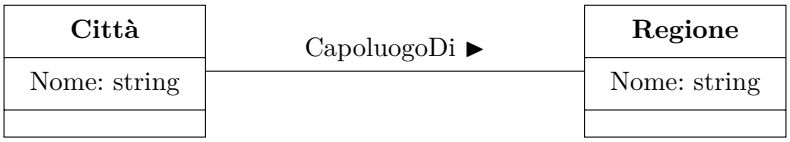
Le principali relazioni che possono esistere tra le classi sono:

- associazione (semplice, aggregazione, composizione);
- generalizzazione;
- dipendenza;
- raffinamento.

9 Associazioni

Un'**associazione** definisce un *canale di comunicazione bidirezionale* tra due classi. Le istanze delle associazioni si dicono **link**: mentre un'associazione connette due classi, un link connette due oggetti che sono istanze di classi associate.

Un'associazione deve avere un *nome* (solitamente un verbo), che ne specifica il significato. Opzionalmente, si può anche aggiungere un triangolo di fianco al nome, per indicare il verso di lettura.



9.1 Molteplicità

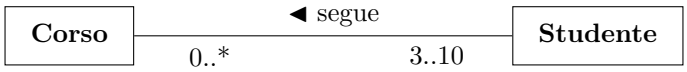
La molteplicità di un'associazione specifica:

- se l'associazione è obbligatoria oppure no;
- il numero minimo e massimo di oggetti che possono essere relazionati a un altro oggetto.

Alcuni esempi di valori di molteplicità sono:

- 0..1: zero oppure uno oggetti relazionati;
- 1: esattamente un oggetto relazionato;
- 0..*: zero o più, cioè qualsiasi numero di oggetti relazionati (si può scrivere anche solo come *);
- 0, 3..5, 7..*: zero, oppure da 3 a 5, oppure 7 o più oggetti relazionati.

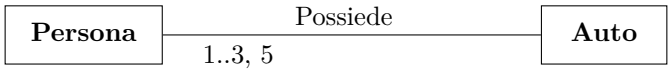
Nei diagrammi, essa si legge "andando verso l'altra classe". Ad esempio, il diagramma



indica che:

- uno studente segue 0 o più corsi;
- un corso è seguito da 3–10 studenti.

La molteplicità può non essere specificata a uno degli estremi dell'associazione, o anche a entrambi: in questo caso, *non* si ha un valore di default (es. 1), ma vale invece il principio di elisione. Ad esempio, il diagramma

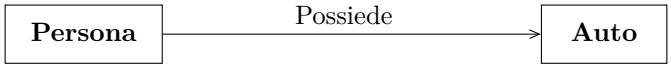


significa che una persona possiede un *numero non specificato* di auto, mentre un'auto è posseduta da 1, 2, 3 o 5 (ma non 4) persone.

9.2 Navigabilità

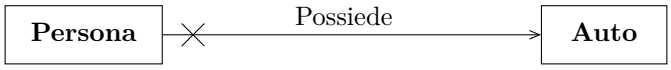
Normalmente, un'associazione è **navigabile** in entrambe le direzioni, cioè dagli oggetti di una classe è possibile risalire agli oggetti relazionati dell'altra, e viceversa.

Per specificare che invece l'associazione è navigabile in una sola direzione, si usa una freccia orientata in tale direzione. Ad esempio, nel diagramma



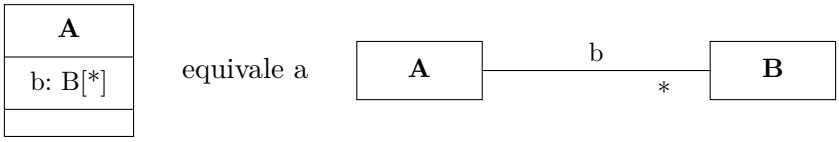
si può risalire da una Persona alle Auto che Possiede, ma non da un'Auto ai suoi proprietari.

Si può anche usare una croce per indicare esplicitamente la non navigabilità nell'altro verso:



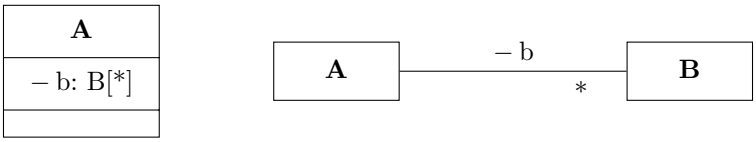
9.3 Notazione degli attributi

Un'associazione può essere rappresentata anche con la notazione degli attributi, specificando come tipo la classe associata, con la molteplicità tra parentesi graffe:



Ciò è utile quando non si vuole mostrare esplicitamente l'altra classe che partecipa all'associazione.

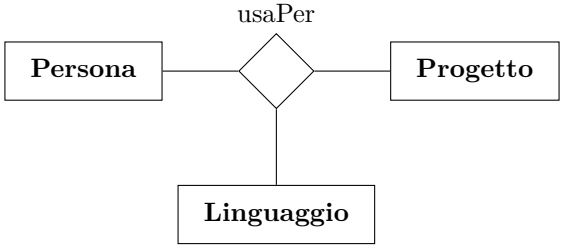
In entrambe le notazioni, può opzionalmente essere specificata la visibilità. Ad esempio:



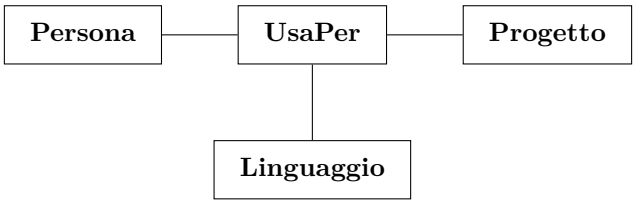
9.4 Associazioni n-arie

In UML è possibile rappresentare anche associazioni tra più di due classi.

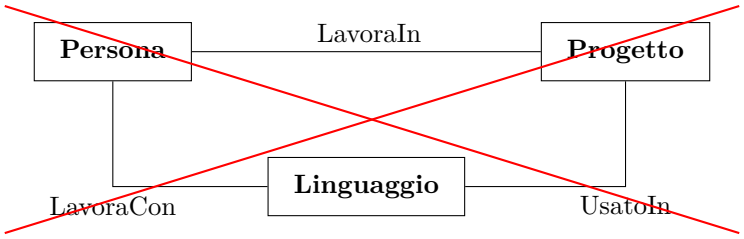
Ad esempio, per indicare che una determinata Persona usa un certo Linguaggio di programmazione per lo sviluppo di uno specifico Progetto:



Solitamente, però, si introduce invece una classe che rappresenta l'associazione: in questo modo, l'associazione n-aria si riduce a un insieme di associazioni binarie:



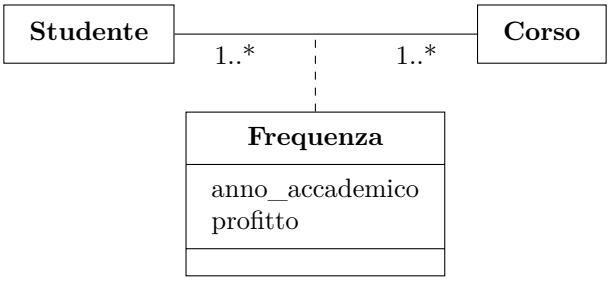
È invece sbagliato sostituire un'associazione n-aria con un insieme di associazioni binarie *senza introdurre un'apposita classe*:



in questo caso, ad esempio, si è perso il legame tra una specifica Persona, uno specifico Linguaggio e uno specifico Progetto.

9.5 Attributi delle associazioni

In molti casi, alcune proprietà sono proprie dell'associazione, piuttosto che delle classi coinvolte. Queste si rappresentano mediante un'apposita classe, chiamata **classe associazione**, che è collegata all'associazione da una linea tratteggiata. Ad esempio:

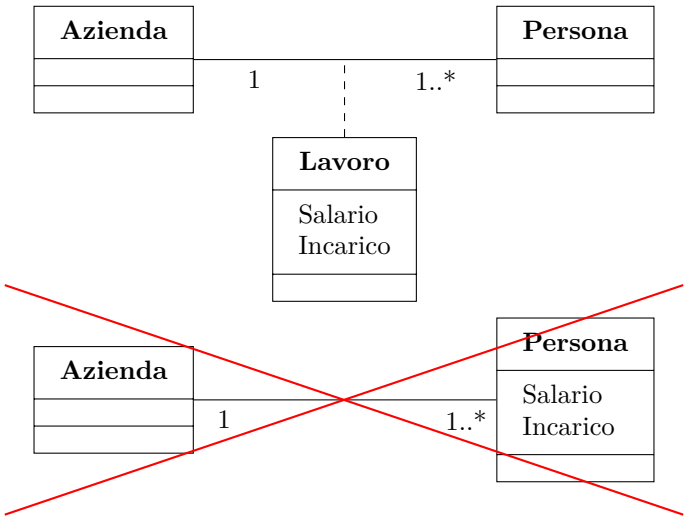


Infatti, l'anno accademico e il profitto:

- non sono attributi dello studente, perché cambiano da corso a corso;

- non sono attributi del corso, perché ogni corso è frequentato da studenti diversi, in anni diversi e con profitti diversi.

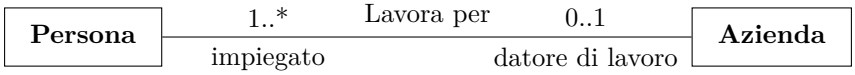
Se l'associazione è uno a molti, i suoi attributi si possono modellare come attributi della classe corrispondente ai "molti", ma ciò non è consigliato, anche perché diventerebbe scorretto se si cambiassero semplicemente le molteplicità.



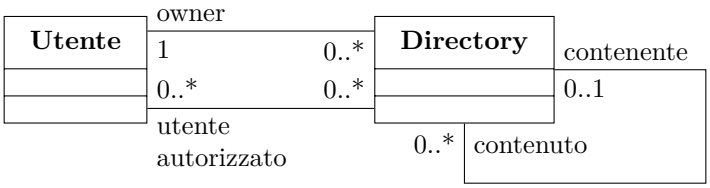
Anche le associazioni n-arie possono avere attributi, con la stessa notazione usata per le associazioni binarie.

9.6 Ruolo

Una classe può partecipare a un'associazione con un **ruolo** specifico, che può essere indicato.



In generale, specificare i ruoli, anche se consigliabile, è opzionale, ma diventa particolarmente utile quando ci sono associazioni multiple tra le stesse classi, e (soprattutto) quando una classe è associata con sé stessa (*associazione riflessiva*: oggetti diversi della stessa classe possono essere in relazione tra loro). In quest'ultimo caso, i ruolo diventano obbligatori, altrimenti il significato dell'associazione sarebbe ambiguo.



9.7 Implementazione

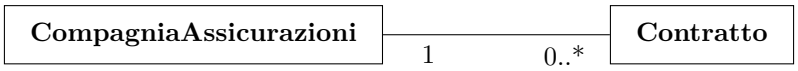
In generale, *non esiste* un mapping diretto dalle associazioni a un costrutto di un linguaggio di programmazione.

Un possibile modo per implementarle è usare attributi che contengono riferimenti (puntatori) alle istanze delle classi associate. Il riferimento a più istanze può essere realizzato con liste, array, ecc. A seconda della navigabilità, tali riferimenti possono essere presenti in una sola delle classi o in entrambe (in quest'ultimo caso, bisogna fare attenzione ad aggiornare sempre i riferimenti di entrambe le classi). Un altro modo è usare una classe che rappresenta l'associazione: essa contiene dei riferimenti alle istanze delle classi associate, più eventuali attributi dell'associazione stessa.

Comunque, la realizzazione delle associazioni è un dettaglio implementativo, che a questo livello di astrazione *non* interessa. In un class diagram, le associazioni devono infatti essere rappresentate come tali, e non "nascoste" nelle classi (ad esempio, come attributi puntatori): la tecnica usata per rappresentarle potrà essere stabilito in seguito.

9.7.1 Esempio

Un esempio di implementazione con riferimenti e navigabilità bidirezionale, realizzato in Java, è:




```java
public class CompagniaAssicurazioni {
    private List<Contratto> contratti;
    // ...
}

public class Contratto {
    private CompagniaAssicurazioni compagnia;
    // ...
}
```

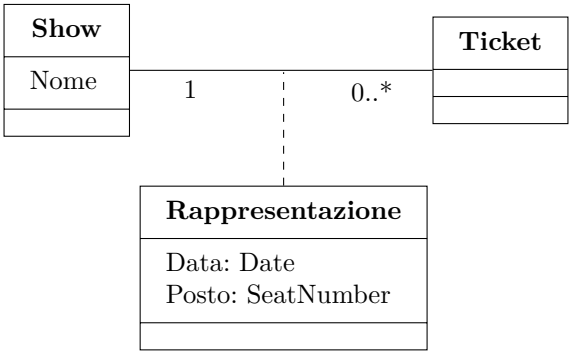
9.8 Qualificatori

Una classe che partecipa a un'associazione uno a molti o molti a molti può avere un **qualificatore**, composto da alcuni attributi dell'associazione, detti *attributi del qualificatore*, che distinguono un oggetto associato tra i molti. In alcuni casi, l'uso di un qualificatore può abbassare la molteplicità, perché vengono specificati in modo più preciso i vincoli sugli oggetti coinvolti.

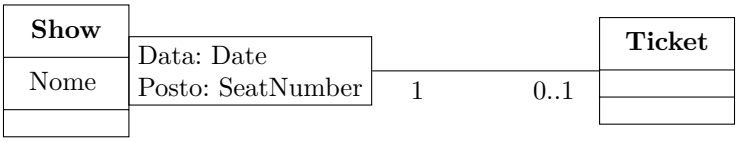
Il qualificatore si posiziona sul lato dell'associazione opposto rispetto a quello dei "molti" oggetti che permette di distinguere.

9.8.1 Esempio 1

Si consideri l'associazione Rappresentazione tra uno spettacolo (Show) e un numero qualsiasi di biglietti (Ticket):



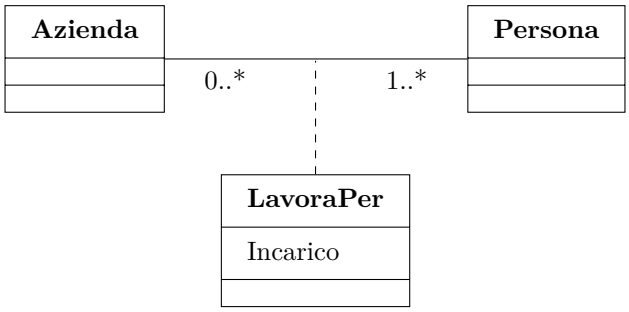
Con un qualificatore, è possibile specificare che, tra i molti biglietti associati a uno spettacolo, per una determinata data e un determinato posto può esistere al massimo un singolo biglietto:



In altre parole, questo qualificatore specifica che gli attributi Data e Posto permettono di "selezionare" un singolo biglietto (se esiste) per un certo spettacolo. Si è quindi ridotta la molteplicità (da 0..* a 0..1).

9.8.2 Esempio 2

Si consideri l'associazione:



In questo caso, l'uso dell'attributo Incarico come qualificatore non abbassa la molteplicità, perché:

- una persona può svolgere lo stesso incarico per più aziende;
- per un'azienda possono lavorare più persone che svolgono lo stesso incarico.