Progettazione del Software

Analisi: Introduzione ad UML & UML Class Diagrams

Domenico Fabio Savo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale Antonio Ruberti

SAPIENZA Università di Roma

Le slide di questo corso sono il frutto di una rielaborazione di analogo materiale redatto da Marco Cadoli, Giuseppe De Giacomo, Maurizio Lenzerini e Domenico Lembo

Analisi

La fase di analisi

- **Cosa è l'analisi**
- Introduzione al linguaggio UML
- Il linguaggio UML per l'analisi
- Metodologia di analisi

Cos'è l'analisi

L'analisi è la fase del ciclo di sviluppo del software caratterizzata da:

INPUT: requisiti raccolti.

OUTPUT: **schema concettuale** (anche detto modello di analisi) dell'applicazione.

OBIETTIVI:

i) costruire un modello dell'applicazione che sia **completo**, **preciso** e **rigoroso** ma anche **leggibile**, indipendente da linguaggi di programmazione e **traducibile** in un programma

ii) concentrarsi su **cosa**, e non su come (indipendenza da aspetti realizzativi/tecnologici)

A cosa serve l'analisi

- Analizzare i requisiti:
  - coglie le loro implicazioni;
  - li specifica con l'obiettivo di formalizzarli e di eliminare incompletezze, inconsistenze e ambiguità.
- Crea un modello (**schema concettuale**) che sarà un riferimento per tutte le fasi successive del ciclo di vita del software.
- Verifica i requisiti con l'utente finale.
- Prende decisioni fondamentali sulla strutturazione e sulla modularizzazione del software.
- Fornisce la specifica delle funzionalità da realizzare.

Che cosa è lo schema concettuale

Lo **schema concettuale (per questo esame)** è costituito da:

Il diagramma delle classi e degli oggetti

Descrive le classi dell'applicazione e le loro proprietà; descrive anche gli oggetti particolarmente significativi.

Il diagramma degli use-case

Descrive le funzionalità fondamentali che il sistema deve realizzare, in termini di scenari di utilizzo del sistema.

I documenti di specifica

Descrivono con precisione quali condizioni devono soddisfare i programmi che realizzano il sistema.

Viene prodotto un documento di specifica per ogni classe, ed un documento di specifica per ogni use case.

Modelli e metodi per l'analisi

Orientati alle funzioni

- diagrammi funzionali
- diagrammi di flusso di controllo
- diagrammi di flusso di dati

Orientati agli oggetti

- Booch
- OOSE (Object-oriented Sooftware Engineering) (Jacobson)
- OMT (Object Modeling Technique) (Rambaugh)
- Coad-Yourdon
- **Basati sul linguaggio UML**

Analisi

La fase di analisi

- Cosa è l'analisi
- **Introduzione al linguaggio UML**
- Il linguaggio UML per l'analisi
- Metodologia di analisi

Il linguaggio UML

UML sta per **Unified Modeling Language**, perché il progetto UML nasce nel 1994 come unificazione di:

- Booch
- Rumbaugh: OMT (Object Medeling Technique)
- Jacobson: OOSE (Object-Oriented Software Engineering)

Storia:

- 1995: Versione 0.8 (Booch, Rumbaugh)
- 1996: Versione 0.9 (Booch, Rumbaugh, Jacbson)
- Versione 1.0 (BRJ + Digital, IBM, HP, ...)
- 1999, 2004: Versione 1,3, 1.4, 1.5, UML si diffonde universalmente
- 2005: Versione 2.0, nuova versione (estende la versione 1.5)

Riferimento:

- *G. Booch, J. Rumbaugh, I. Jacobson, "The unified modeling language user guide", Addison Wesley, 1999. (2° ed. 2005)*
- *http://www.uml.org/*

Diagrammi UML

- **Diagrammi strutturali:**
  - **Diagramma delle classi e degli oggetti** (*class and object diagram*)
- **Diagrammi comportamentali:**
  - **Diagramma degli use case** (*use case diagram*),
  - Diagramma degli stati e delle transizioni (state/transition diagram),
  - Interaction (Sequence e Collaboration diagram),
  - Activity diagram
- **Diagrammi architetturali:**
  - Component diagram
  - Deployment diagram

Uso di UML nella nostra metodologia

- La metodologia che illustriamo in questo corso **si basa su UML**, ma non è esattamente la metodologia usualmente associata a UML.
- Nella nostra metodologia di analisi noi useremo i seguenti diagrammi (e di questi diagrammi useremo solo le caratteristiche più importanti):
  - **Diagrammi strutturali**:
    - **Diagramma delle classi e degli oggetti**
  - **Diagrammi comportamentali**:
    - **Diagramma degli use case**
- Useremo UML con alcune limitazioni e regole precise

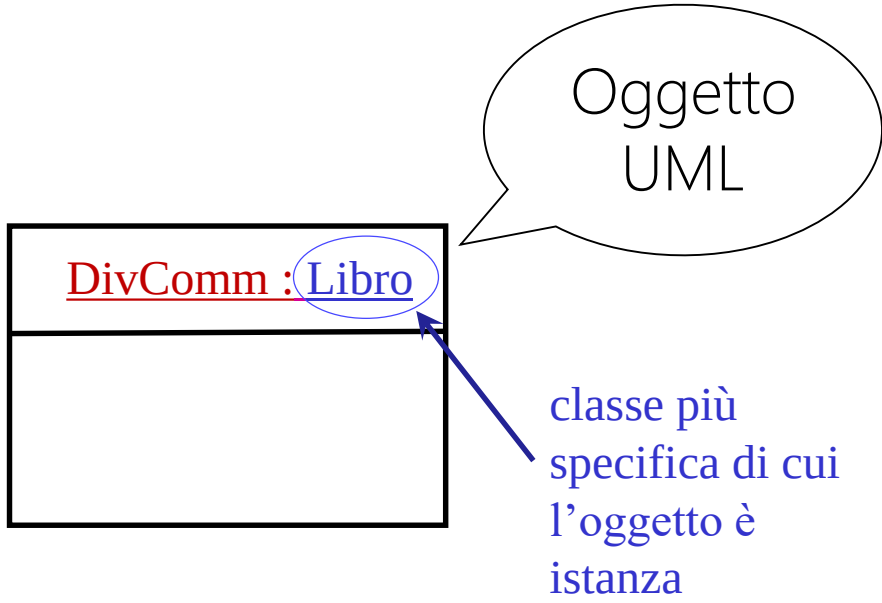
Diagramma delle classi e degli oggetti per l'analisi

- Nella fase di analisi ci si concentra sulle **classi** più che sugli **oggetti**
- Gli oggetti servono essenzialmente per descrivere elementi singoli particolarmente significativi (oltre che per scopi didattici)
- Come detto in precedenza, noi faremo riferimento solo ad un sottoinsieme dei meccanismi previsti in **UML** per descrivere il diagramma delle classi

Oggetti in UML

- Un **oggetto** in UML modella **un** elemento del dominio di analisi che
  - ha vita propria
  - è identificato univocamente mediante l'**identificatore di oggetto**
  - è istanza di una classe (la cosiddetta **classe più specifica** – vedremo che, in determinate circostanze, un oggetto è istanza di più classi, ma in ogni caso, tra le classi di cui un oggetto è istanza, esiste sempre la classe più specifica)

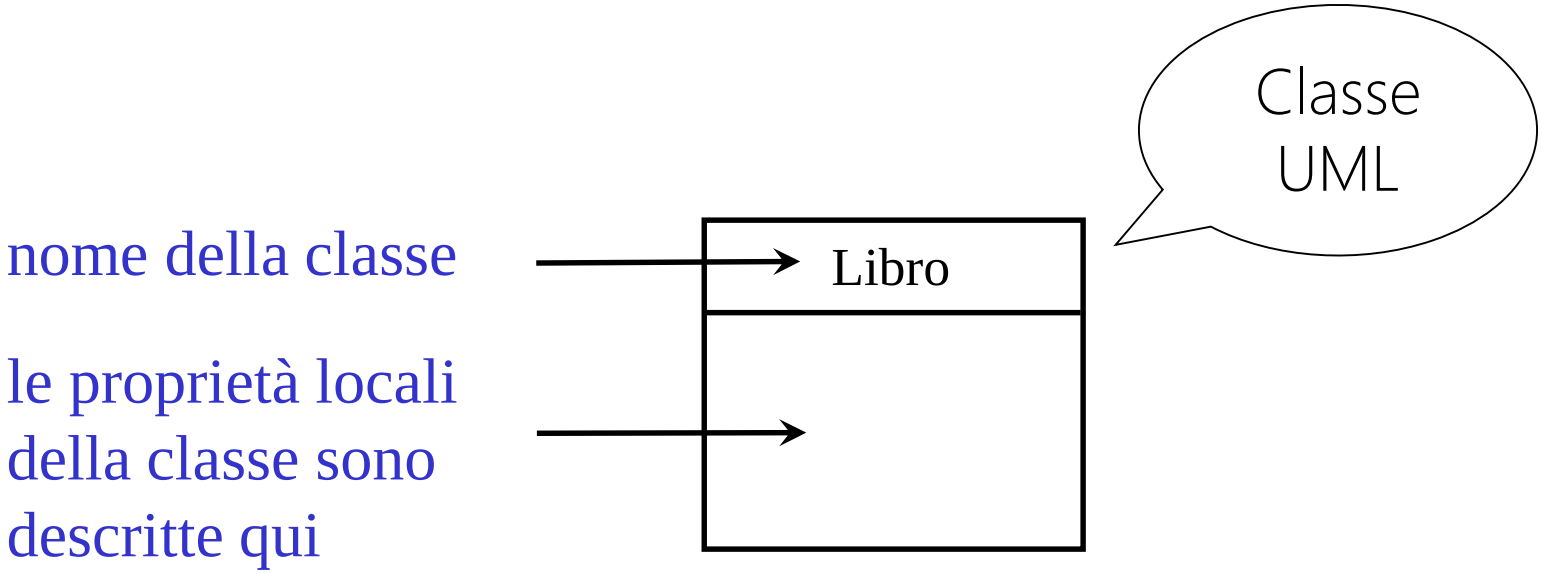
- **DivComm** è l'identificatore di oggetto
- **Libro** è la classe (più specifica) di cui l'oggetto è istanza
- Si noti la sottolineatura



Classi in UML

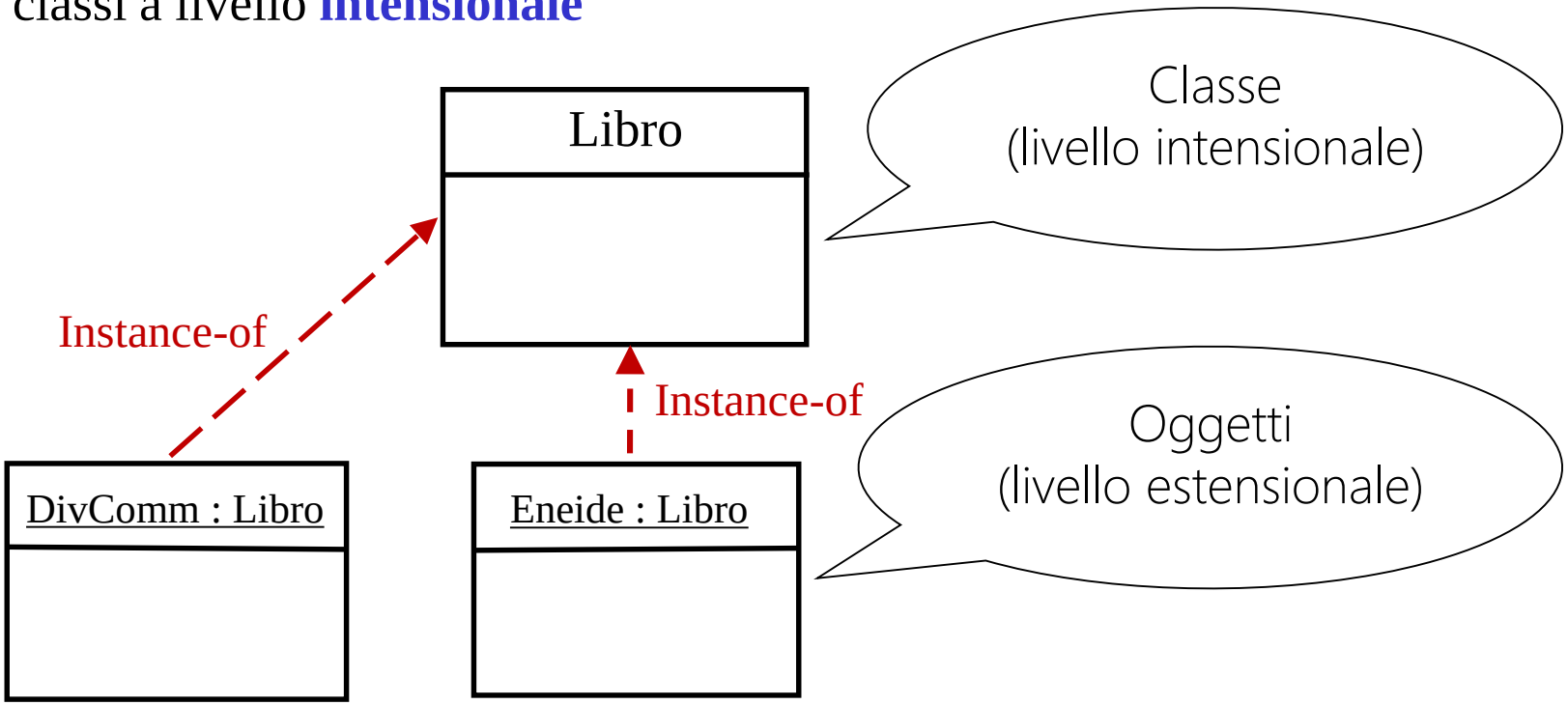
Una **classe** modella un insieme di oggetti omogenei (le **istanze** della classe) ai quali sono associate proprietà statiche e dinamiche (operazioni). Ogni **classe** è descritta da:

- un **nome**
- un **insieme di proprietà "locali"** (astrazioni delle proprietà comuni degli oggetti che sono istanze delle classi)



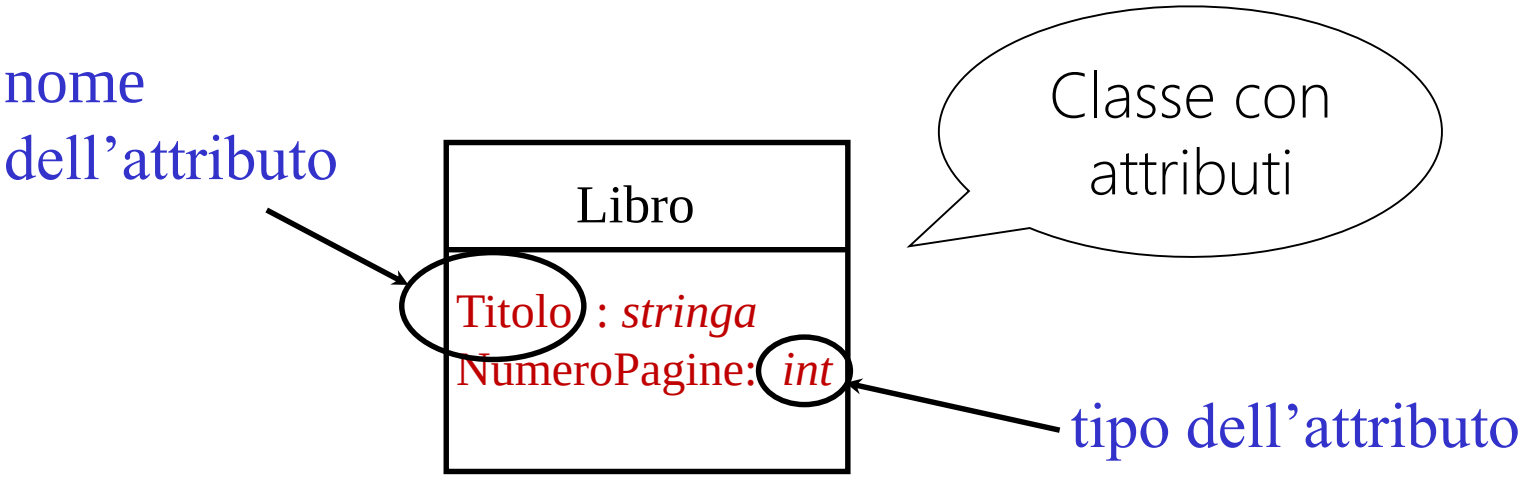
Rapporto tra classi e istanze

- Tra un **oggetto** che è **istanza** di una **classe** *C* e la classe *C* si traccia un arco **Instance-of** (l'arco in realtà non è strettamente necessario, perché la classe di cui l'oggetto è istanza è già indicata nell'oggetto)
- Ricordiamo che gli oggetti formano il livello **estensionale**, mentre le classi a livello **intensionale**



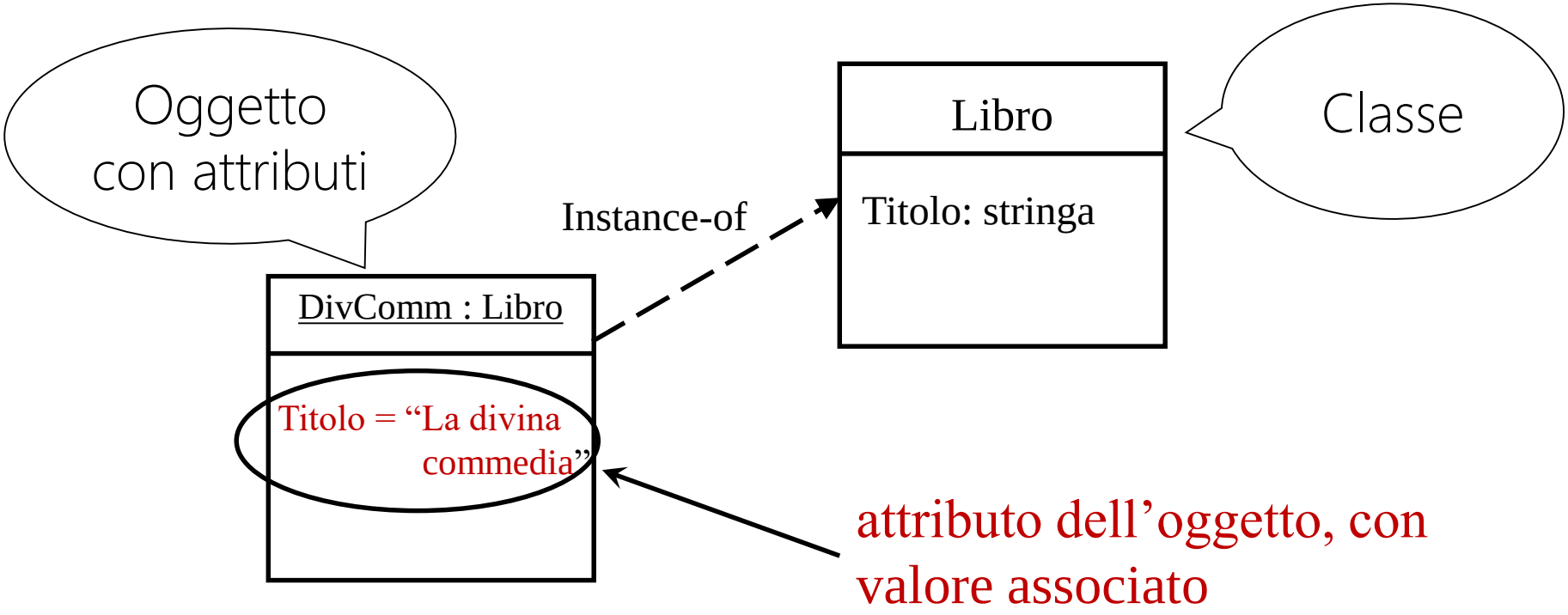
Proprietà di classi: attributi in UML

- Un **attributo** modella una proprietà **locale** della classe ed è caratterizzato da un **nome** e dal **tipo** dei valori associati (es.: stringa, intero)
- Ogni attributo di una classe stabilisce una proprietà locale **valida per tutte le istanze** della classe. Il fatto che la proprietà sia locale significa che è un proprietà **indipendente da altri oggetti**
- Formalmente, un attributo ***A*** della classe ***C*** si può considerare una **funzione** che associa un valore di tipo ***T*** ad ogni oggetto che è istanza di ***C***



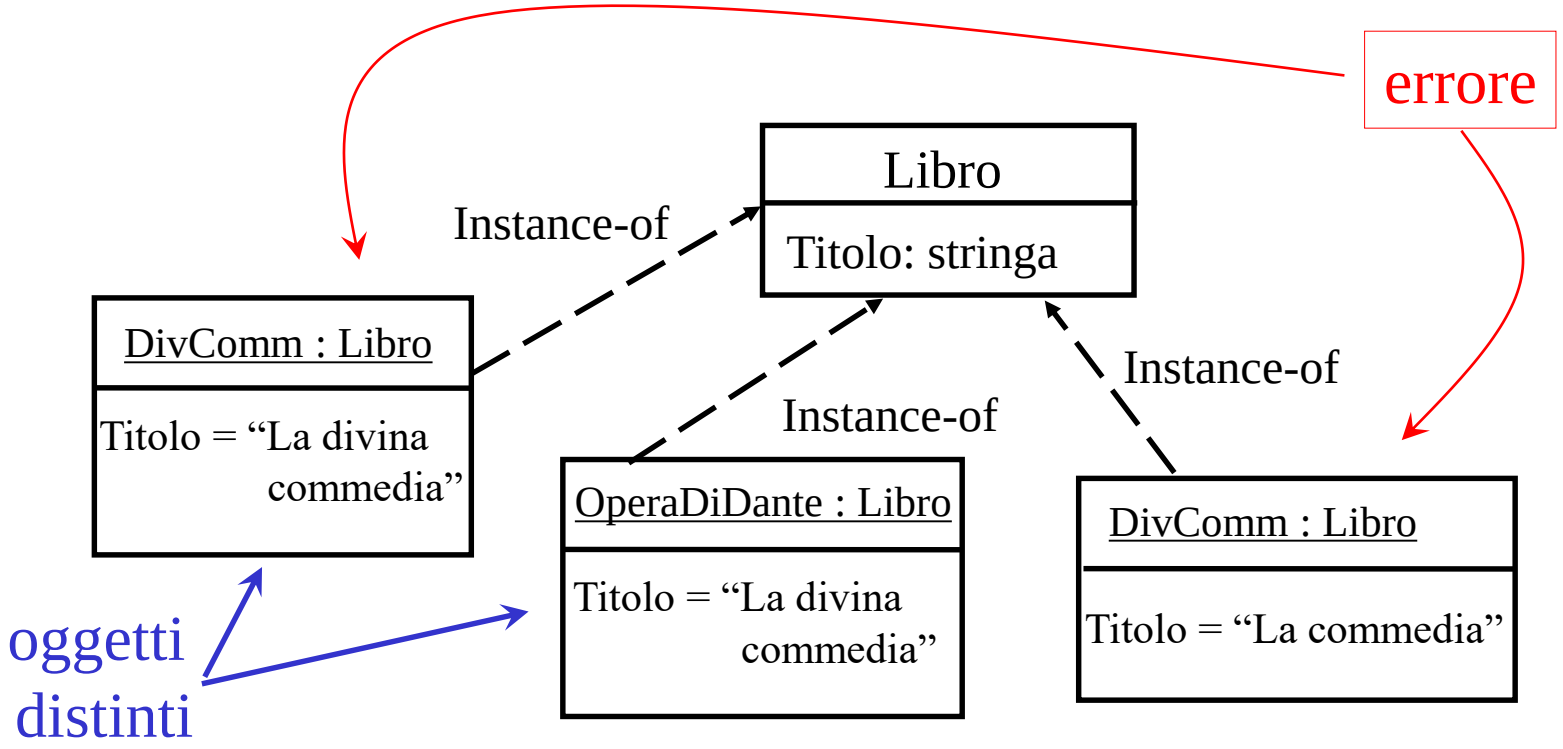
Attributi di oggetti

- Gli attributi di una classe determinano gli attributi delle sue istanze
- **Regola importante**: se una classe ***C*** ha un attributo ***A*** di tipo ***T***, **ogni** oggetto che è istanza di ***C*** ha l'attributo ***A***, con un valore associato di tipo ***T***
- **Regola importante**: un oggetto **X** non può avere un valore per un attributo non definito nella classe di cui **X** è istanza

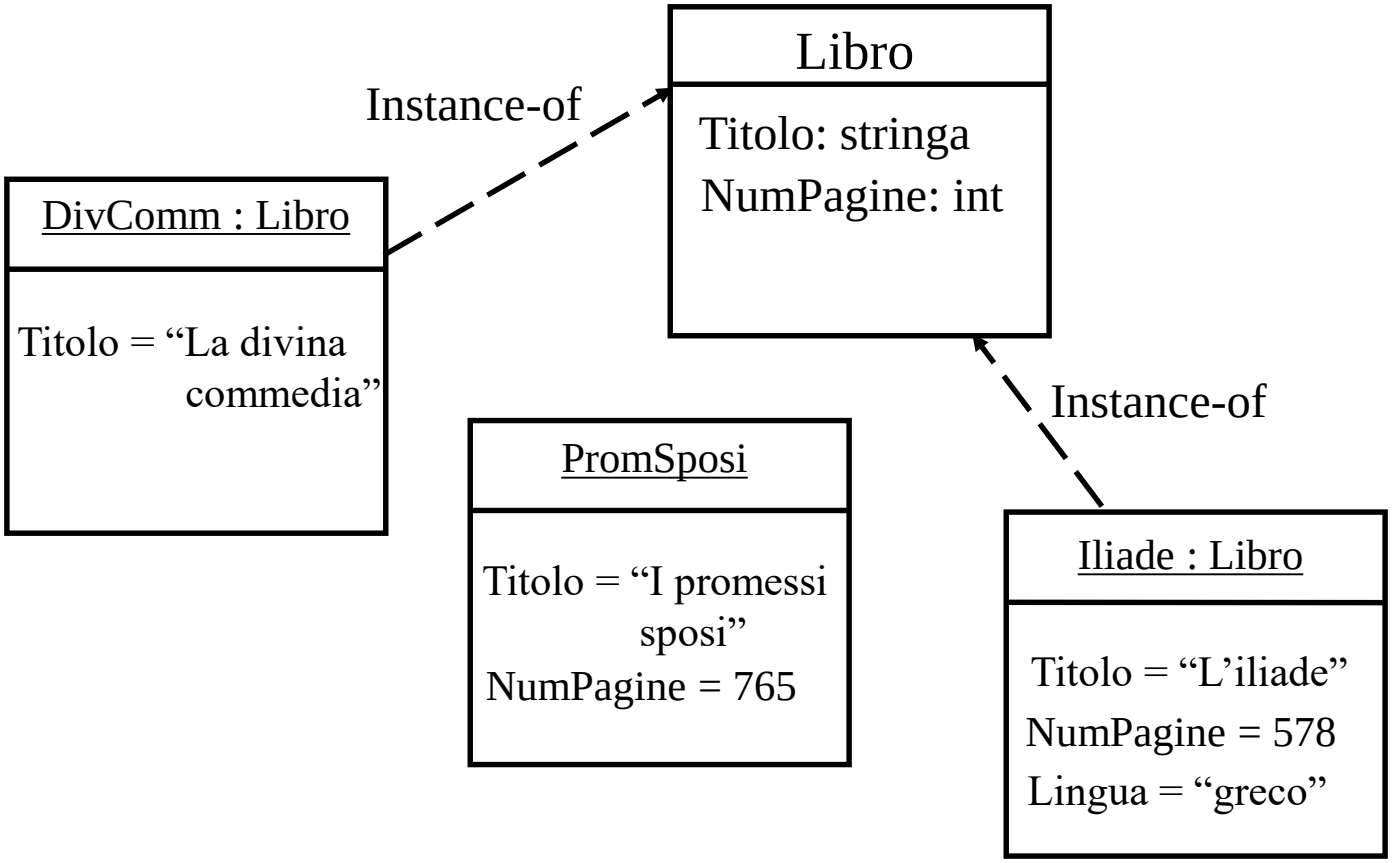


Importanza dell'identificatore di oggetto

- Due oggetti con identificatori diversi sono comunque distinti, ***anche se hanno i valori di tutti gli attributi uguali***
- Due **oggetti diversi** devono avere **identificatori diversi**, anche se possono avere gli stessi valori per tutti gli attributi



Esercizio 1



Il diagramma è corretto? Se no, quali sono gli errori?

Esercizio 1 (errori presenti nel diagramma)

Esercizio 1 (diagramma corretto)

Osservazioni sulle classi

- Per tutto ciò che è stato detto finora, due classi diverse non possono avere istanze comuni. In altre parole, classi diverse modellano insiemi disgiunti (*torneremo su questo punto quando introdurremo la* ***generalizzazione***)
- Si noti la **distinzione tra oggetti** (istanze di classi) **e valori** (di un certo tipo):
  - un oggetto è istanza di una **classe** ed ha vita propria
  - un valore è un elemento di un **tipo**, ed ha senso solo se associato ad un oggetto tramite un attributo
- Il livello intensionale **determina** come è strutturato il livello estensionale

Intermezzo: relazione matematica (1)

Se S_1 e S_2 sono due insiemi, una relazione R tra S_1 e S_2 è un sottoinsieme del prodotto cartesiano tra S_1 e S_2

$$R \subseteq S_1 \times S_2$$

Il prodotto cartesiano $S_1 \times S_2$ tra S_1 e S_2 è l'insieme di tutte le coppie $\langle x,y \rangle$ tali che $x \in S_1$ e $y \in S_2$

Ovviamente, la nozione si estende banalmente a relazioni tra ***n*** insiemi:

$$R \subseteq S_1 \times S_2 \times \ldots \times S_n$$

Intermezzo: relazione matematica (2)

- Consideriamo gli insiemi $S_1 = \{1, 2, 3\}$ e $S_2 = \{a, b\}$
- Prodotto cartesiano:

  $S_1 \times S_2 = \{ \langle 1,a \rangle, \langle 1,b \rangle, \langle 2,a \rangle, \langle 2,b \rangle, \langle 3,a \rangle, \langle 3,b \rangle \}$

- Esempio di relazione tra S_1 e S_2 :

  $R \subseteq S_1 \times S_2 = \{ \langle 1,a \rangle, \langle 1,b \rangle, \langle 2,a \rangle \}$

- Si noti come una relazione R selezioni un sottoinsieme degli elementi del prodotto cartesiano, quelli che sono significativi rispetto al significato della relazione R

Intermezzo: relazione matematica (3)

Consideriamo i seguenti insiemi:

studente = {Paolo, Anna, Lucia}
esame = {Analisi, Geometria}

Esempio di relazione "sostenuto" tra "studente" ed "esame":

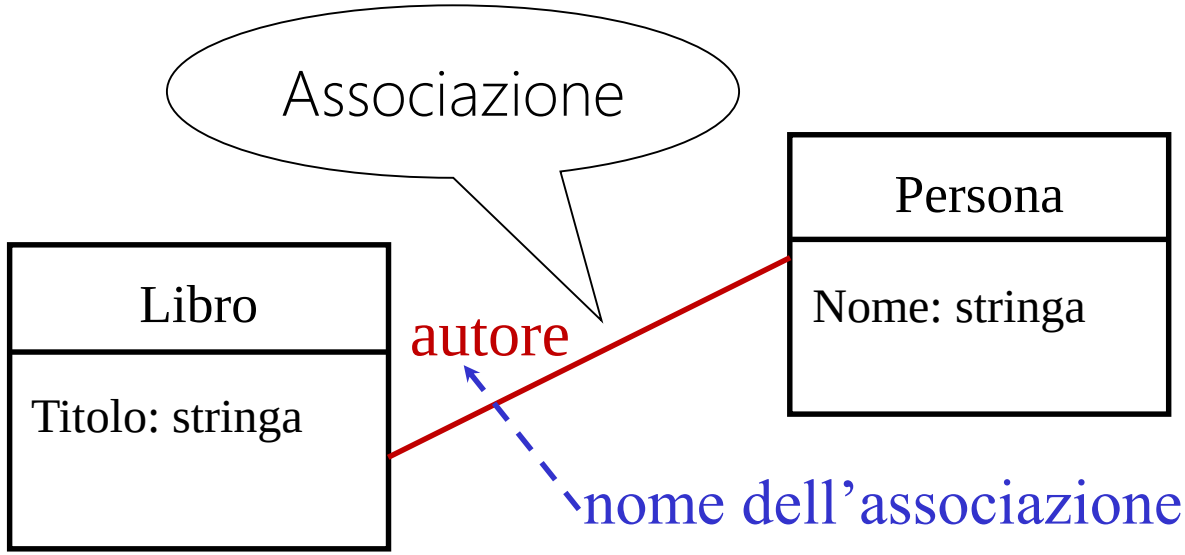
sostenuto = {<Paolo,Analisi>, <Anna,Analisi>, <Anna,Geometria>}

Si noti come, tra tutte le coppie del prodotto cartesiano tra "studente" ed "esame", la relazione "sostenuto" seleziona un insieme di coppie, e cioè solo le coppie **<*x*,*y*>** tali che lo studente *x* ha sostenuto l'esame *y*

Proprietà di classi: associazioni in UML

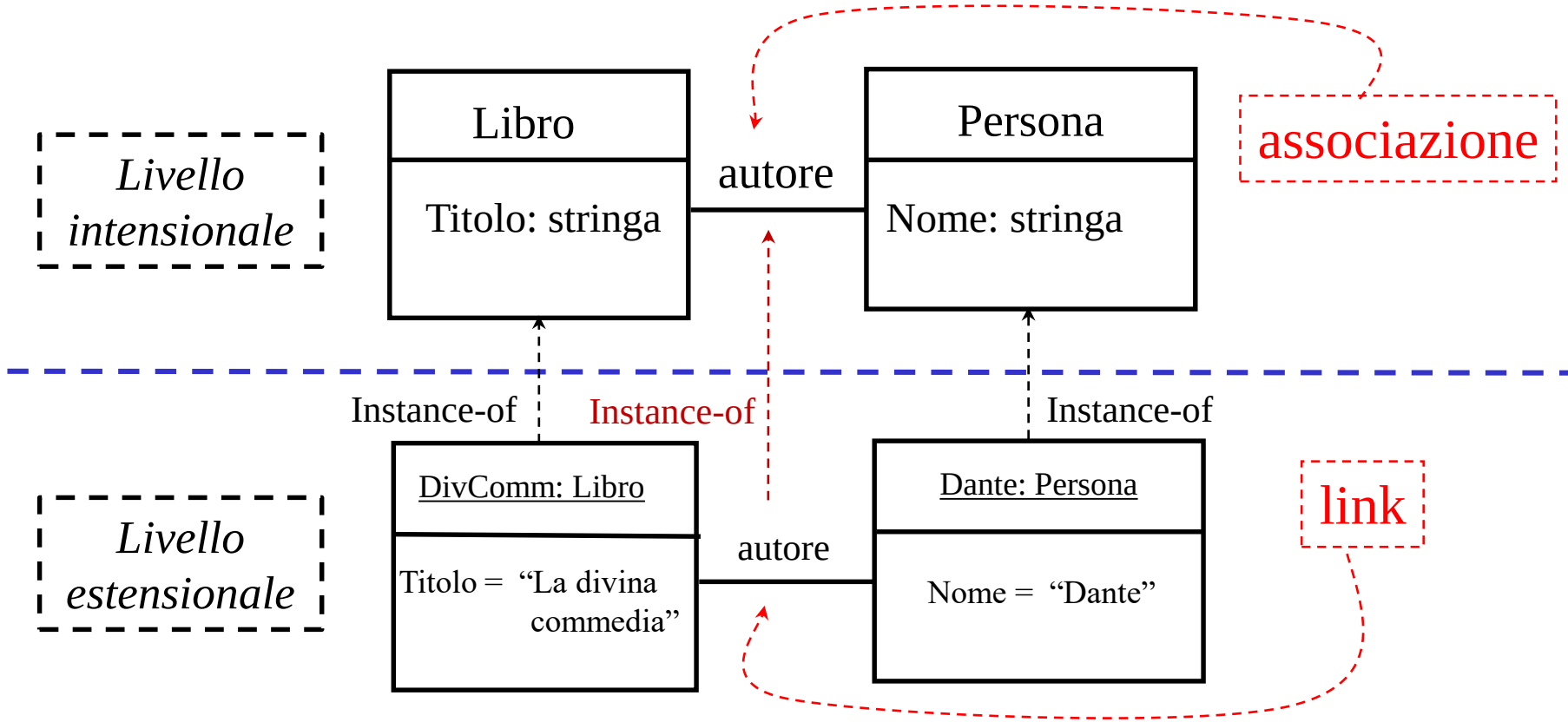
- Per il momento, ci limitiamo a discutere associazioni tra due classi (ma le associazioni possono coinvolgere anche N classi)
- Una **associazione** (o relazione) tra una classe C_1 ed una classe C_2 modella una relazione matematica tra l'insieme delle istanze di C_1 e l'insieme delle istanze di C_2
- Gli attributi modellano proprietà locali di una classe, le associazioni invece modellano **proprietà che coinvolgono altre classi**. Una associazione tra due classi modella una proprietà di **entrambe le classi**

Nota: "autore" è una proprietà sia di libro sia di persona



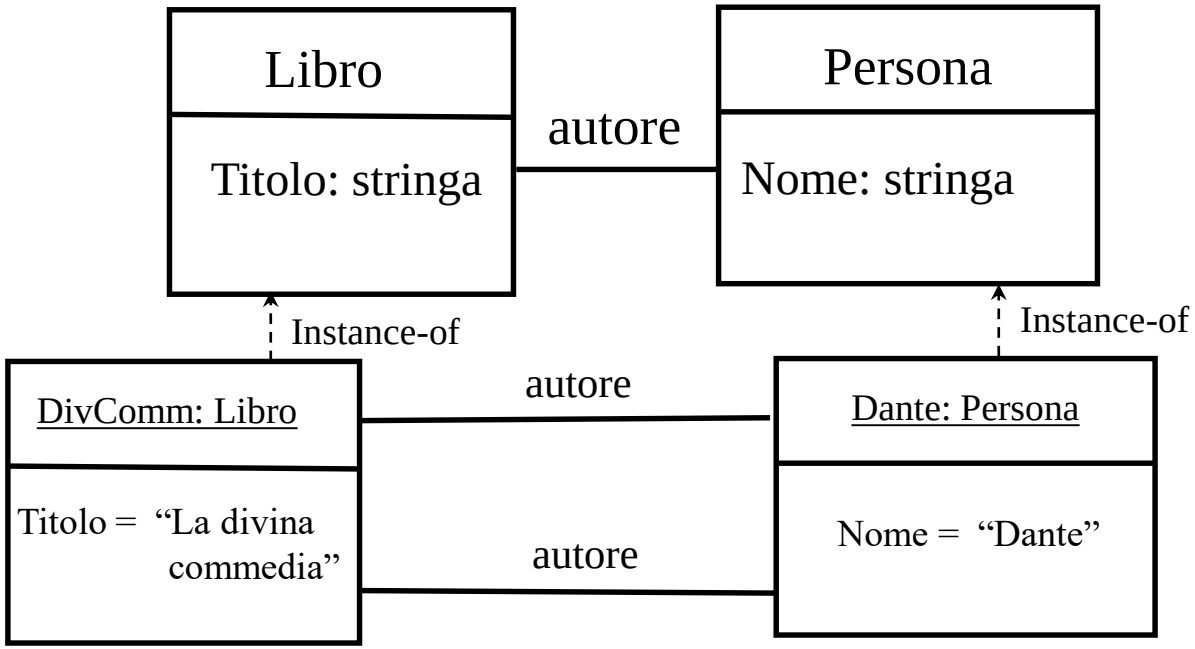
Istanze di associazioni: Link (1)

Le istanze di associazioni si chiamano **link**: se *A* è una associazione tra le classi C_1 e C_2, una istanza di *A* è un link tra due oggetti (in altre parole, una **coppia**), uno della classe C_1 e l'altro della classe C_2



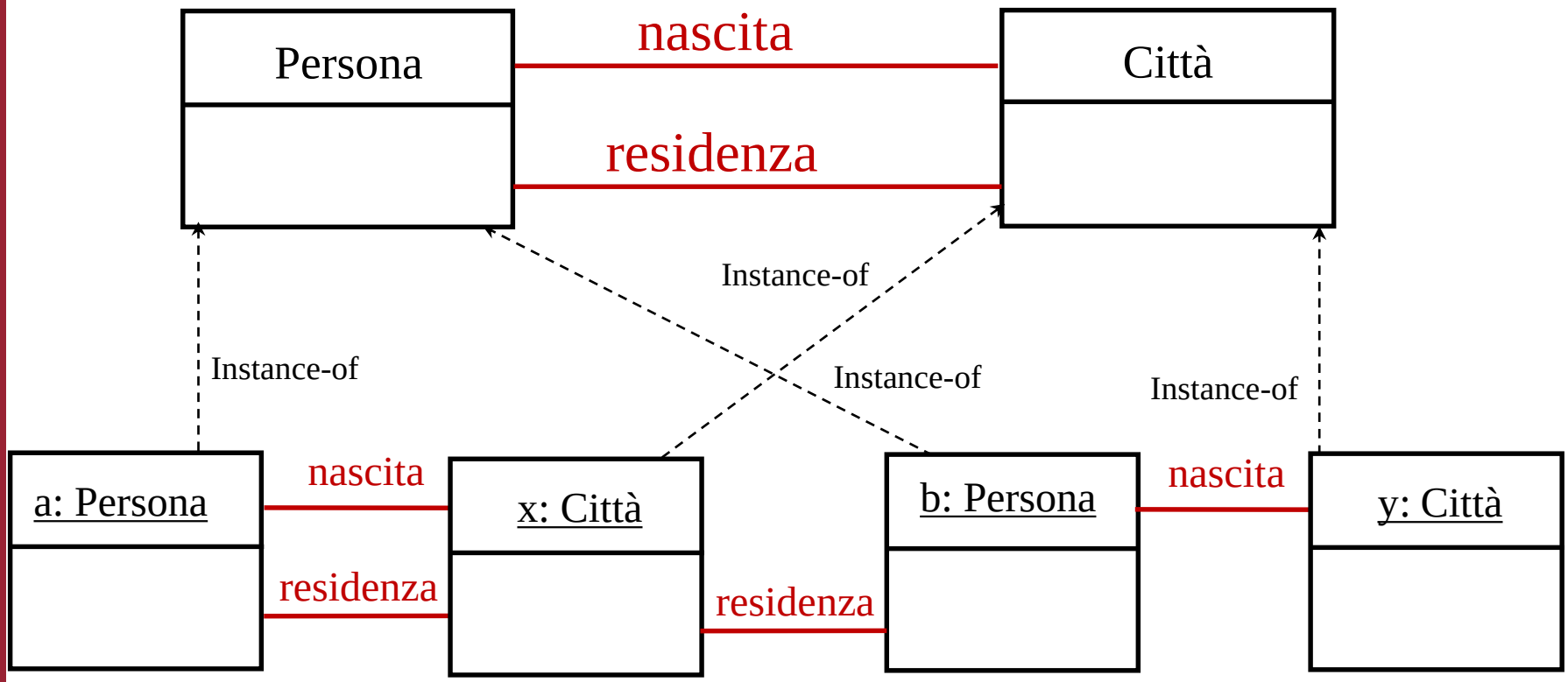
Istanze di associazioni: Link (2)

- Come gli oggetti sono istanze delle classi, così i link sono **istanze delle associazioni** (gli archi *instance-of* non sono necessari)
- Al contrario degli oggetti, però, i link non hanno identificatori espliciti: **un link è implicitamente identificato dalla coppia (o in generale dalla ennupla) di oggetti che esso rappresenta**
- Ciò implica, ad esempio, che il seguente diagramma è **errato**:



Più associazioni tra due classi

Ovviamente, tra le stesse due classi possono essere definite più associazioni



Ancora sulle associazioni in UML

Attenzione: una relazione *R* tra C_1 e C_2 non dice nulla sul numero di link di *R* che coinvolgono due istanze delle classi C_1 e C_2. Ad esempio, dato questo diagramma:

Una istanza di Persona può essere legata a **zero, una, o più** istanze di Città da link di tipo "nascita"

Vedremo successivamente come si possono specificare condizioni sul numero di link che coinvolgono un oggetto in UML (ad esempio per imporre che ogni istanza deve avere esattamente un link di tipo "nascita" con una istanza di Città)

Esercizio 2

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare gli studenti (con nome, cognome, numero di matricola, età), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame. Di ogni corso di laurea interessa il codice e il nome. Di ogni corso interessa il nome e la disciplina a cui appartiene (ad esempio: matematica, fisica, informatica, ecc.).

Esercizio 2 (soluzione)

Esercizio 2 (esempi di istanza)

Esercizio 3

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare le persone (con nome, cognome, età), le città di nascita (con nome, numero di abitanti, e sindaco), e le regioni in cui si trovano le città (con nome, anno di istituzione, e presidente).

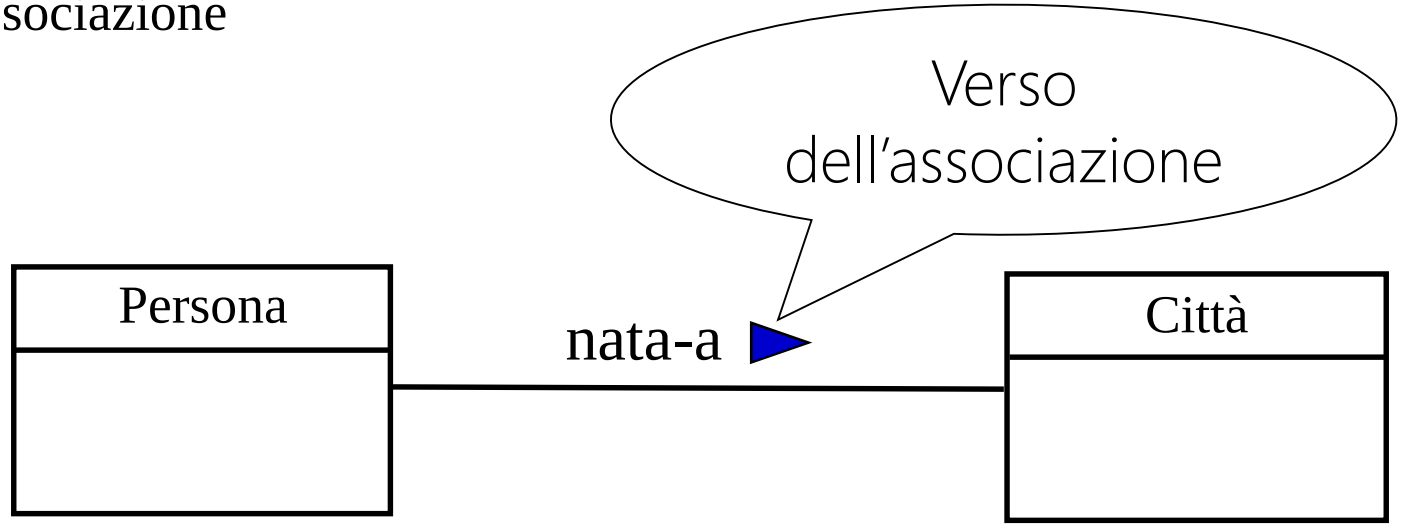
Esercizio 3 (soluzione)

Possibile errore nell'esercizio 3

Possibile errore nell'esercizio 3

Nomi di associazioni (1)

- Alcune volte è interessante specificare un verso per il nome della associazione

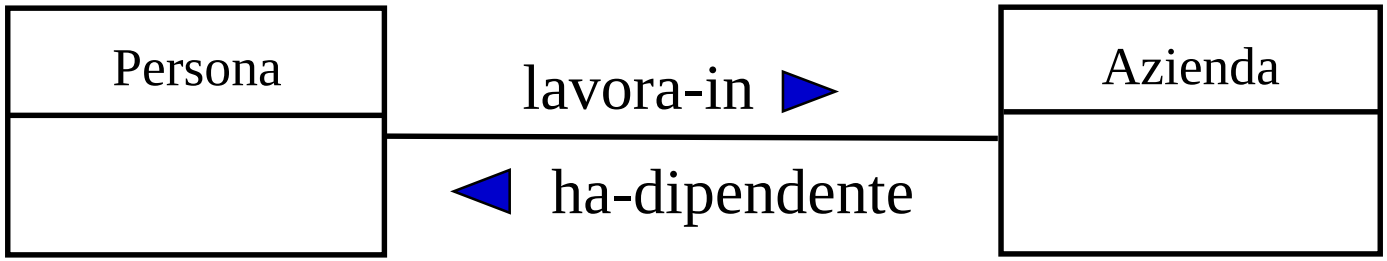


- Attenzione: la notazione riportata sopra non significa che l'associazione è navigabile (attraversabile) solo in un verso

- In altre parole, il verso non è una caratteristica del significato della associazione, ma dice semplicemente che il nome scelto per la associazione evoca un verso (nell'esempio, il verso è dalla Persona alla Città)

Nomi di associazioni (2)

- È anche possibile assegnare due nomi con i relativi versi alla stessa associazione

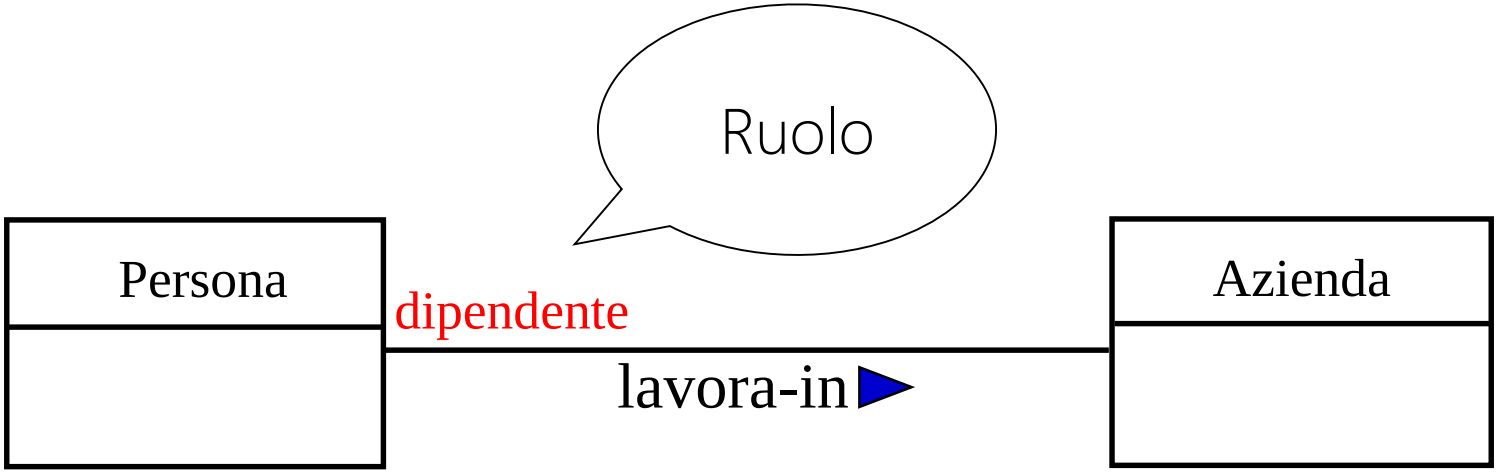


- Osserviamo ancora che le frecce che simboleggiano il verso non aggiungono nulla al significato della associazione (che formalmente si può considerare sempre una relazione matematica), ma aiutano ad interpretare il senso dei nomi scelti per l'associazione

- Le frecce che simboleggiano il verso si indicano anche nel link che sono istanze delle associazioni

Ruoli nelle associazioni

- È possibile aggiungere alla associazione una informazione che specifica il ruolo che una classe gioca nella associazione



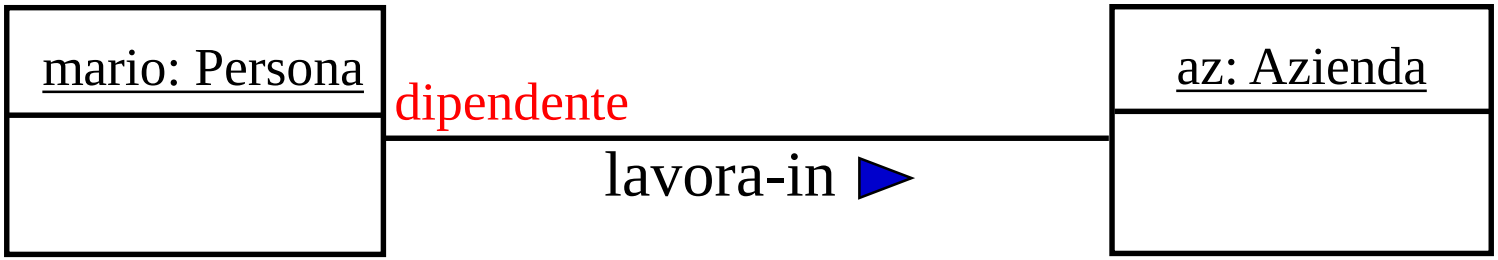
- Il ruolo si indica con un nome posizionato lungo la linea che rappresenta l'associazione, vicino alla classe alla quale si riferisce

- Nell'esempio, dipendente è il ruolo che la persona gioca nell'associazione "lavora-in" con Azienda

Ruoli nei link

Se nella associazione *A* è indicato il ruolo rivestito dalla classe *C*, tale ruolo sarà indicato (vicino alla corrispondente istanza di *C*) in ogni link che è istanza di *A*

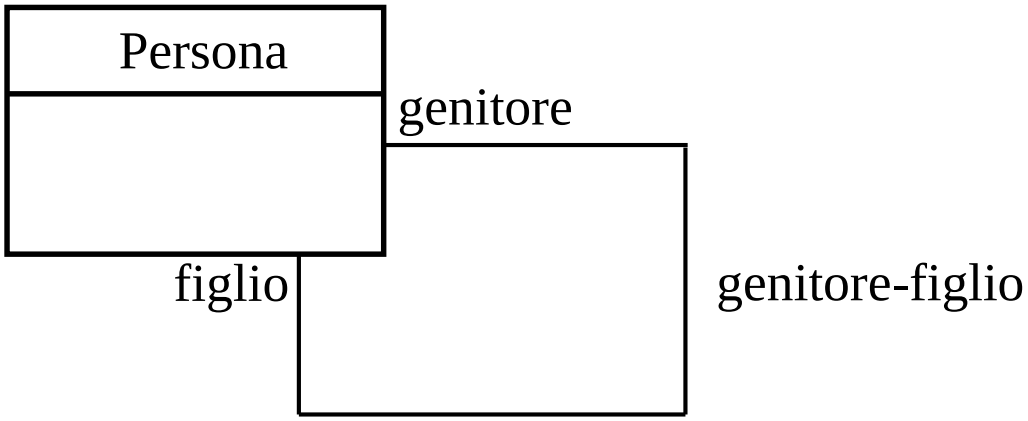
Esempio:



Ancora sui ruoli

- Analogamente al verso, il ruolo è generalmente opzionale, e non aggiunge nulla al significato del'associazione
- L'unico caso in cui il ruolo è **obbligatorio** è quello in cui l'associazione insiste più volte sulla stessa classe, e rappresenta una relazione **non simmetrica**

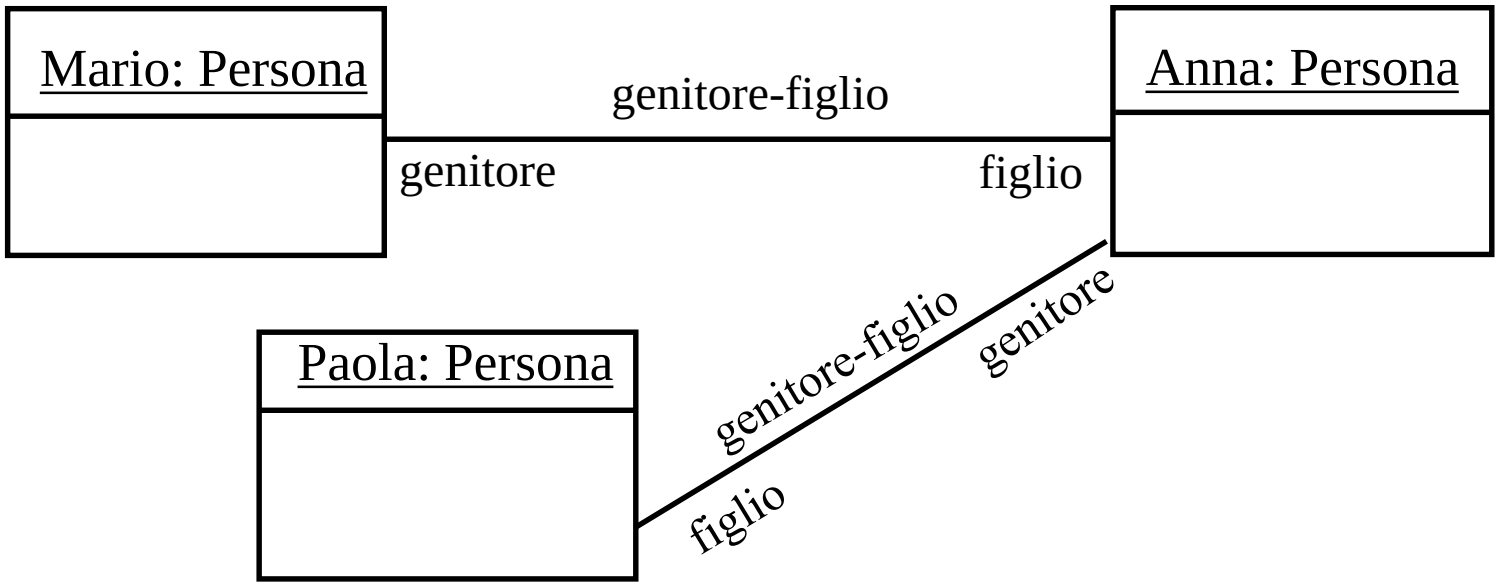
Esempio:



- Se non fossero indicati i ruoli nell'associazione "genitore-figlio", non sapremmo interpretare correttamente i link che sono istanze dell'associazione

Osservazioni sui ruoli

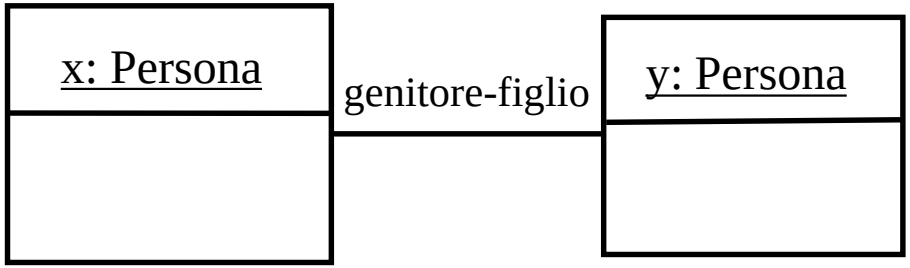
Esempio della necessità dei ruoli nei link:



Se non fossero indicati i ruoli nell'associazione "genitore-figlio", non sapremmo interpretare correttamente i link che sono istanze dell'associazione

Importanza dei ruoli

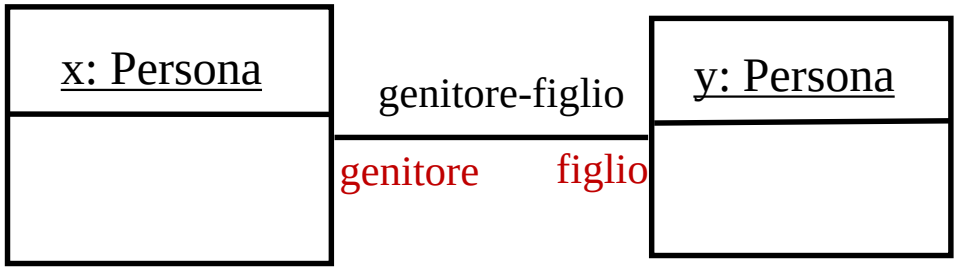
Senza ruoli non è chiaro chi è il genitore e chi il figlio. In altre parole, dalla coppia <*x*,*y*> in genitore-figlio non si evincono i ruoli di *x* e *y*



Con i ruoli, non c'è più ambiguità. Formalmente, il link è ora una coppia etichettata:

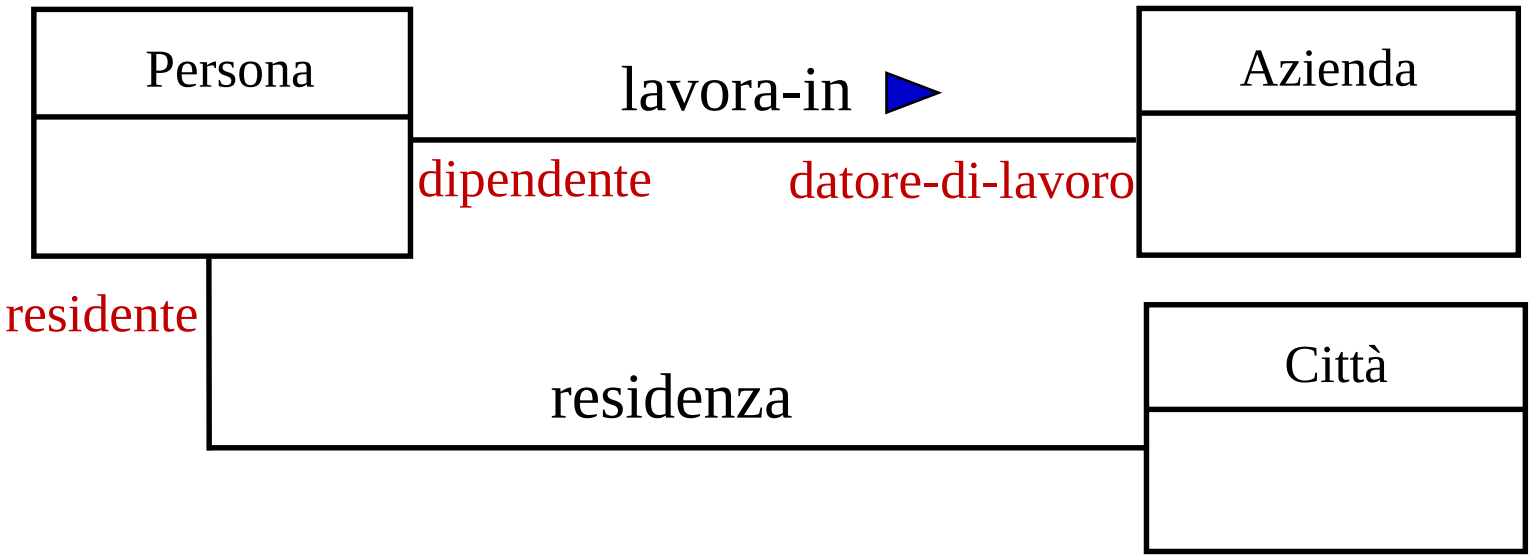
<genitore:x, figlio:y>

dalla quale si evincono i ruoli di *x* e *y*



Importanza dei ruoli

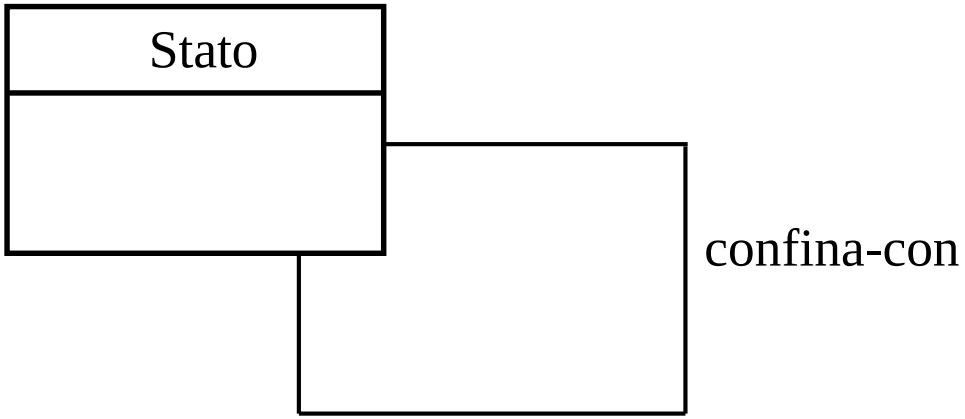
Anche nei casi in cui non è strettamente necessario, il ruolo può essere utile per aumentare la leggibilità del diagramma



Casi di ruoli non significativi

- Ci sono casi in cui l'associazione insiste più volte sulla stessa classe, ma il ruolo non è significativo, in particolare quando l'associazione rappresenta una **relazione simmetrica**

Esempio:



Esercizio 4

Considerare le seguenti relazioni fra persone:

1. Essere amico
2. Essere coniuge
3. Essere collega di lavoro
4. Essere un superiore (nell'ambito del lavoro)

Per quali fra esse prevedereste dei ruoli?

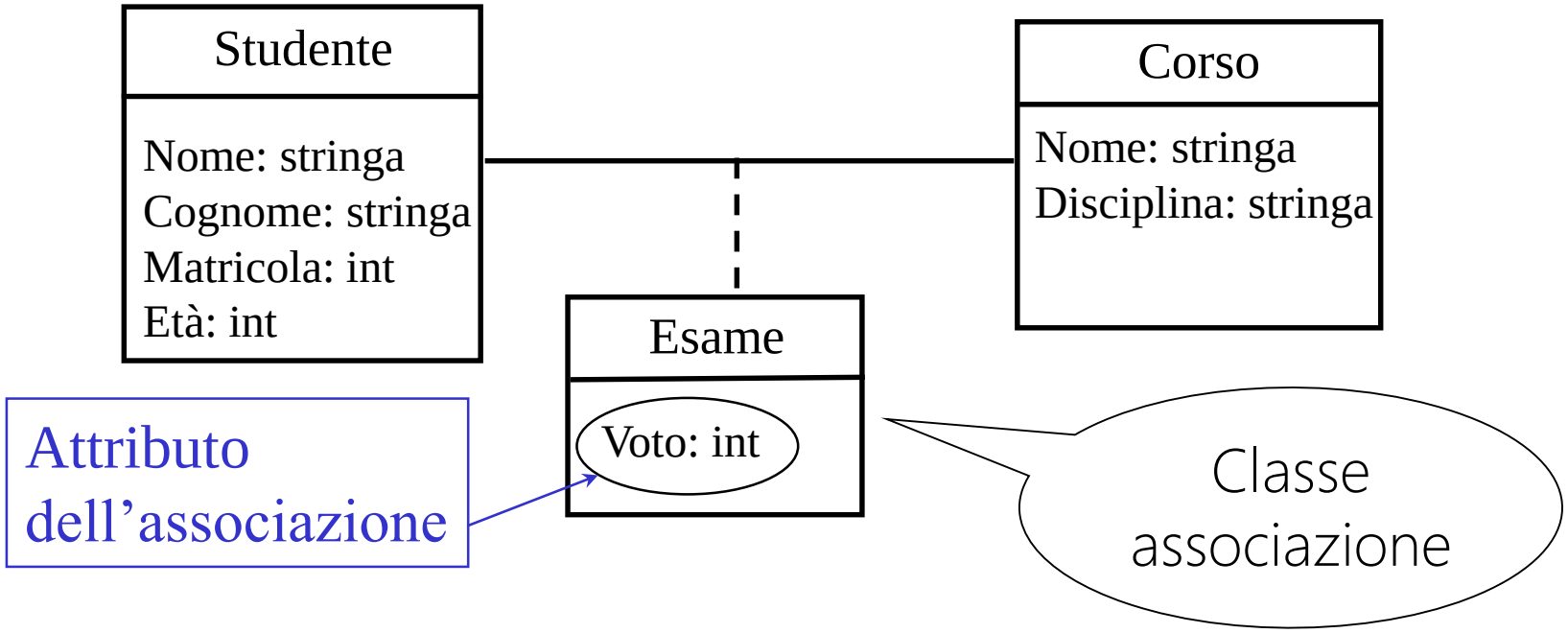
In tal caso, quali nomi sceglieresti?

Attributi di associazione

Analogamente alle classi, anche **le associazioni possono avere attributi**.

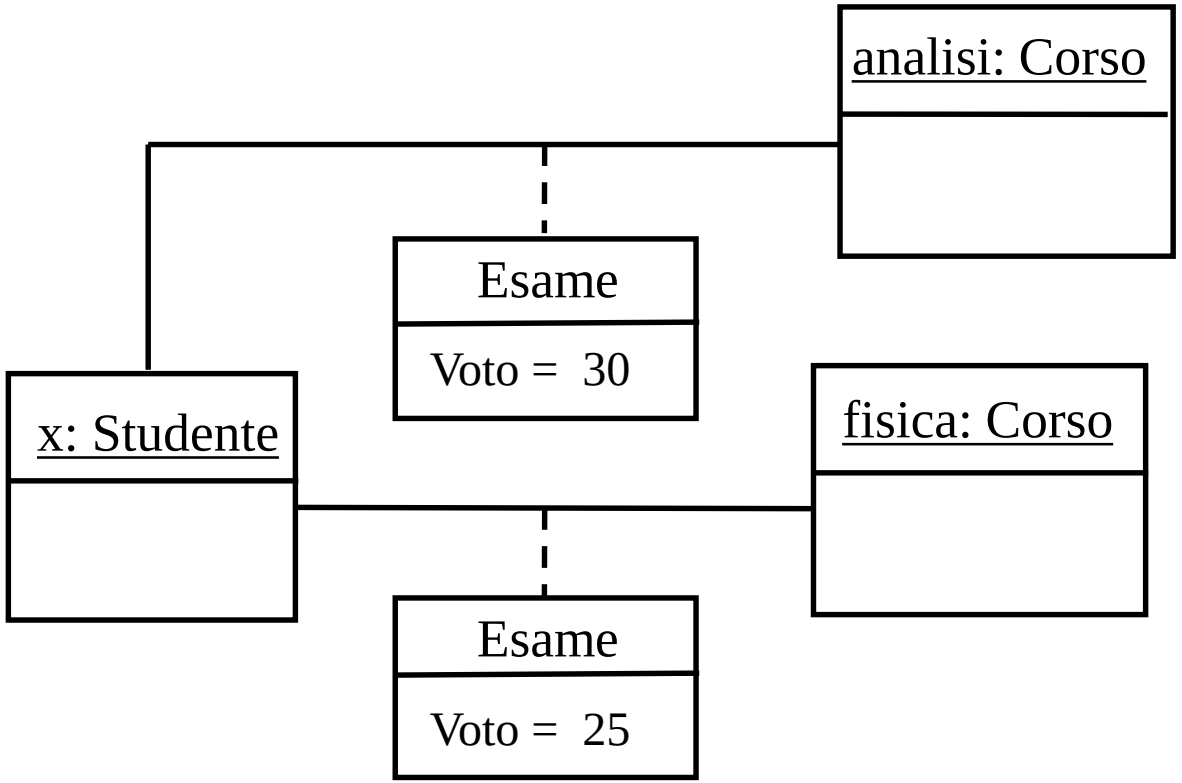
Formalmente, un attributo di una associazione è una **funzione** che associa ad ogni link che è istanza dell'associazione un valore di un determinato tipo

Esempio: voto non è una proprietà ne di Studente, ne di Corso, ma della associazione Esame tra Studente e Corso



Attributi nei link (1)

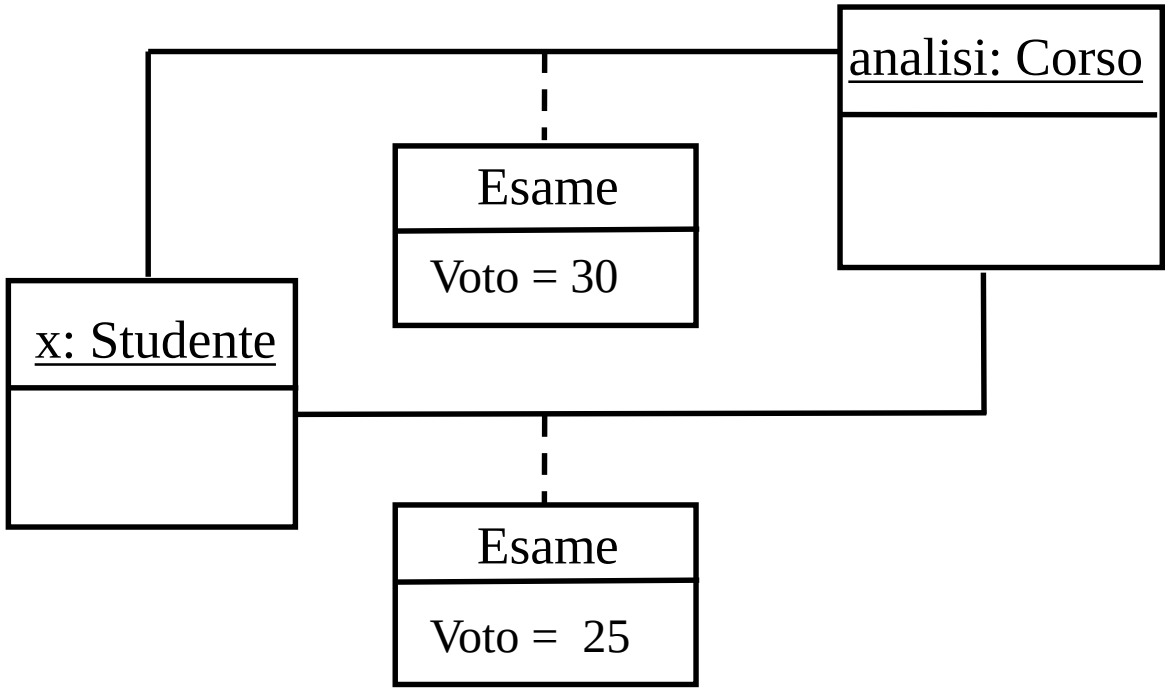
Ovviamente, se una associazione ha un attributo, ogni link che è istanza dell'associazione avrà un valore per quell'attributo



Attributi nei link (2)

Attenzione: con la presenza degli attributi, il significato dell'associazione (relazione matematica) non cambia.

Quindi questo è un errore:



Esercizio 5

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare le persone (con nome, cognome, età), le città di nascita (con nome, numero di abitanti, e sindaco, compresa l'indicazione dell'anno di elezione), e le regioni in cui si trovano le città (con nome, anno di istituzione, e presidente, con l'anno di elezione e lista politica in cui è stato eletto).

Esercizio 5 (soluzione)

Esercizio 6

Con riferimento al diagramma delle classi precedente, è possibile:

1. Rappresentare che Gianni è nato a Bari ed è sindaco di Roma, eletto nel 2008?
2. Inoltre, che Francesco è nato a Roma ed è sindaco di Roma, eletto nel 1992?
3. Inoltre, che Walter è nato a Roma ed è sindaco di Roma, eletto nel 1999?
4. Inoltre, che Walter è stato eletto nuovamente sindaco di Roma nel 2003?

Esercizio 7

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare i clienti che prenotano presso un locale. Dei clienti interessa il nome e il cognome. Del locale la via e il nome. Della prenotazione interessa giorno ed ora, ed il numero di posti da prenotare.

Esercizio 7 (soluzione 1)

Esercizio 7 (soluzione 2)

Esercizio 8

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Gli impiegati afferiscono ai dipartimenti da una certa data. Degli impiegati interessa il nome, l'età, e lo stipendio. Dei dipartimenti interessa il nome e il numero di telefono. Ogni dipartimento ha un direttore. Gli impiegati partecipano ai progetti. Dei progetti interessa il nome e il budget.

Esercizio 8 (soluzione)

Esercizio 9

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Degli studenti interessa il numero di matricola, la data di nascita, la città e la regione di nascita, la facoltà in cui è iscritto (con l'anno di iscrizione), e i corsi superati.

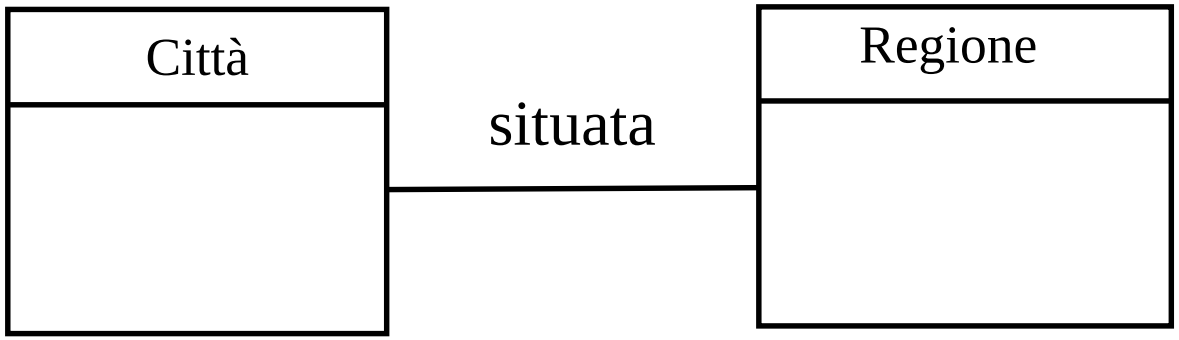
Dei professori interessa il nome, la data di nascita, la città di nascita e il corso insegnato.

Delle facoltà interessa il nome ed il tipo (scientifica, letteraria, ecc..).

Dei corsi interessa il codice, il numero di ore di lezione, e la facoltà a cui appartiene.

Esercizio 9 (soluzione)

Significato delle associazioni

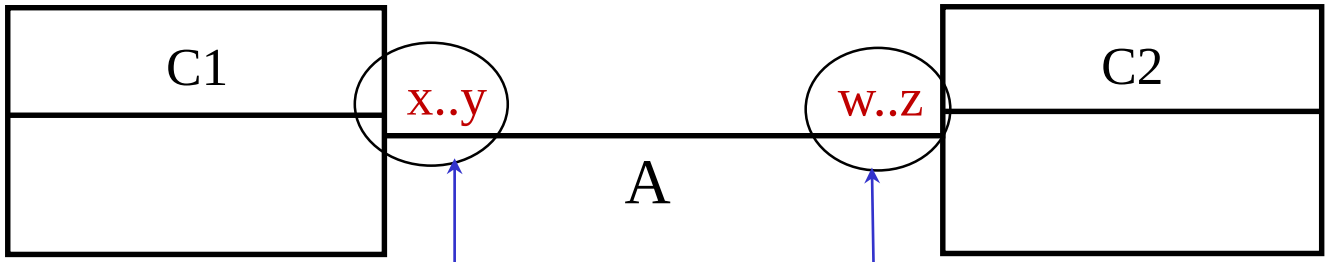


Quali delle seguenti cose ci dice questo diagramma delle classi?

1. Ogni città è situata in una regione
2. Ogni regione ha delle città in essa situate
3. Alcune città sono situate in una regione
4. Alcune regioni hanno città situate in esse
5. Nessuna delle precedenti (specificare)

Molteplicità delle associazioni

- Per specificare con maggiore precisione il significato delle **associazioni binarie** (non ennarie – vedi dopo) si possono definire i **vincoli di molteplicità** (*o semplicemente molteplicità*) delle associazioni



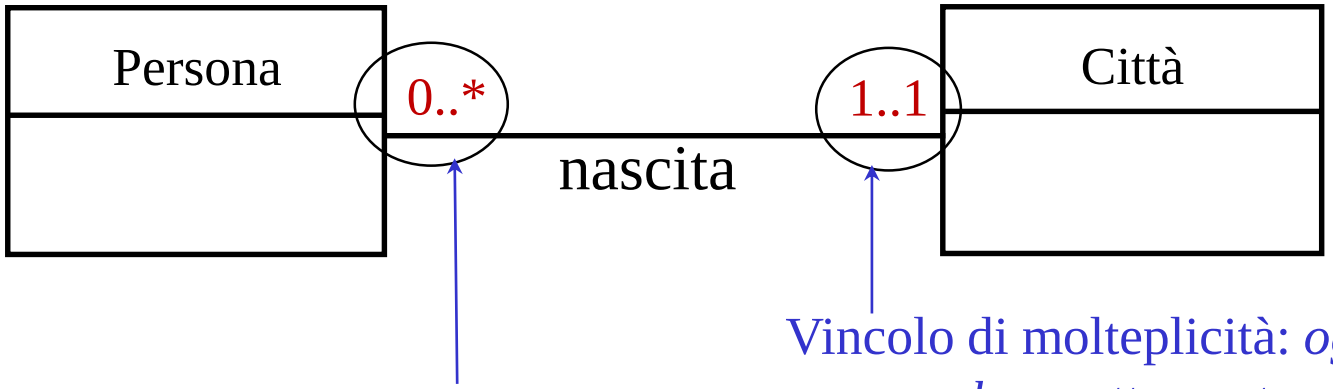
Ogni istanza di **C2** è legata ad **almeno x e al massimo y** istanze di C1 mediante l'associazione A

Ogni istanza di **C1** è legata ad **almeno w e al massimo z** istanze di C2 mediante l'associazione A

Molteplicità delle associazioni (esempio)

Esempio:

ogni istanza di Persona deve essere legata ad **esattamente** una istanza (cioè ad almeno una e al massimo una) istanza di Città da link della associazione "nascita"



Non ci sono vincoli di molteplicità per le città: *in una Città può essere nato un numero qualunque di persone*

Vincolo di molteplicità: *ogni persona ha esattamente una città di nascita*

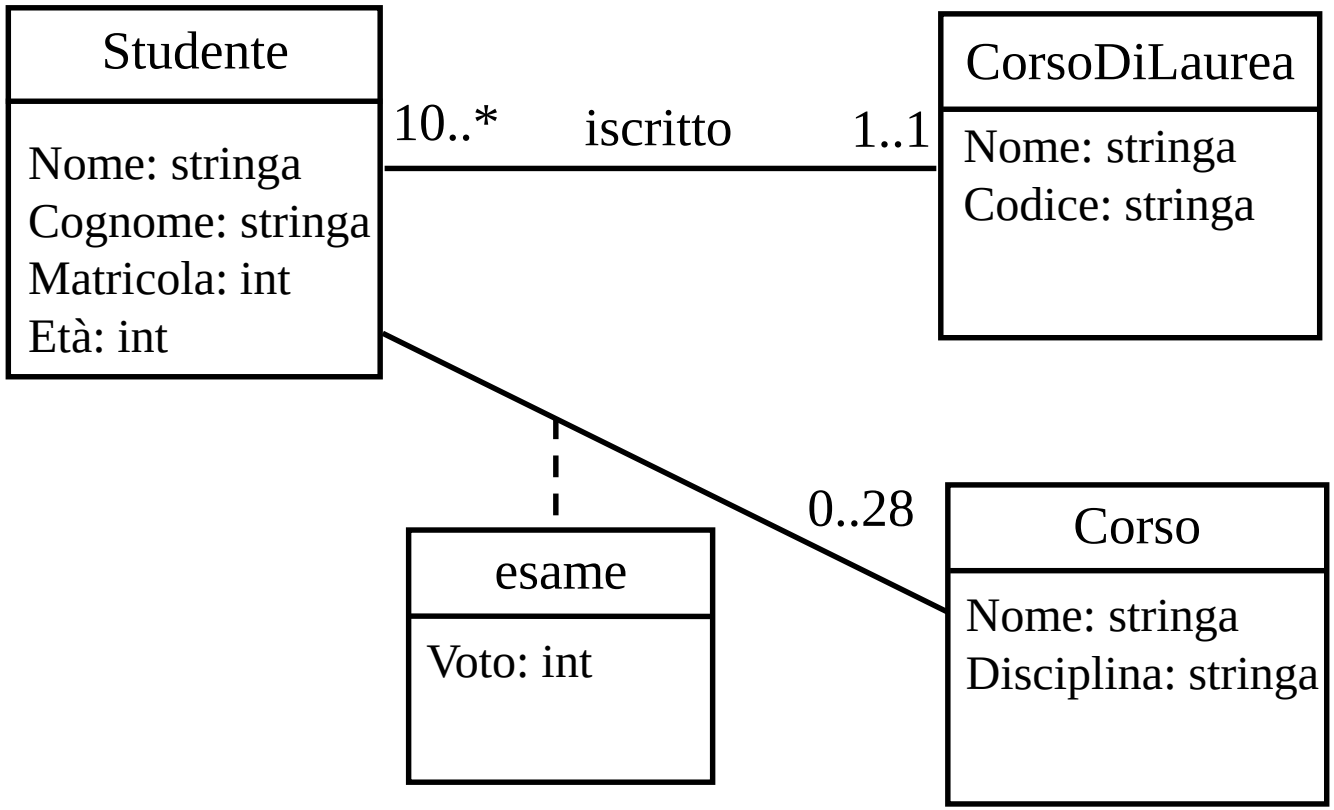
Molteplicità delle associazioni: notazione

Le molteplicità si definiscono solo per le associazioni binarie

Possibili molteplicità:

0 .. *	(nessun vincolo, si può evitare di indicare)
0 .. 1	(nessun limite per il minimo, e al massimo una)
1 .. *	(al minimo una, e nessun limite per il massimo)
1 .. 1	(esattamente una)
0 .. y	(nessun limite per il minimo, e al massimo y, con y intero > 0)
x .. *	(al minimo x, con x intero > 0, e nessun limite per il massimo)
x .. y	(al minimo x e al massimo y, con x ,y interi, x >0 e y > x)

Esempi di molteplicità



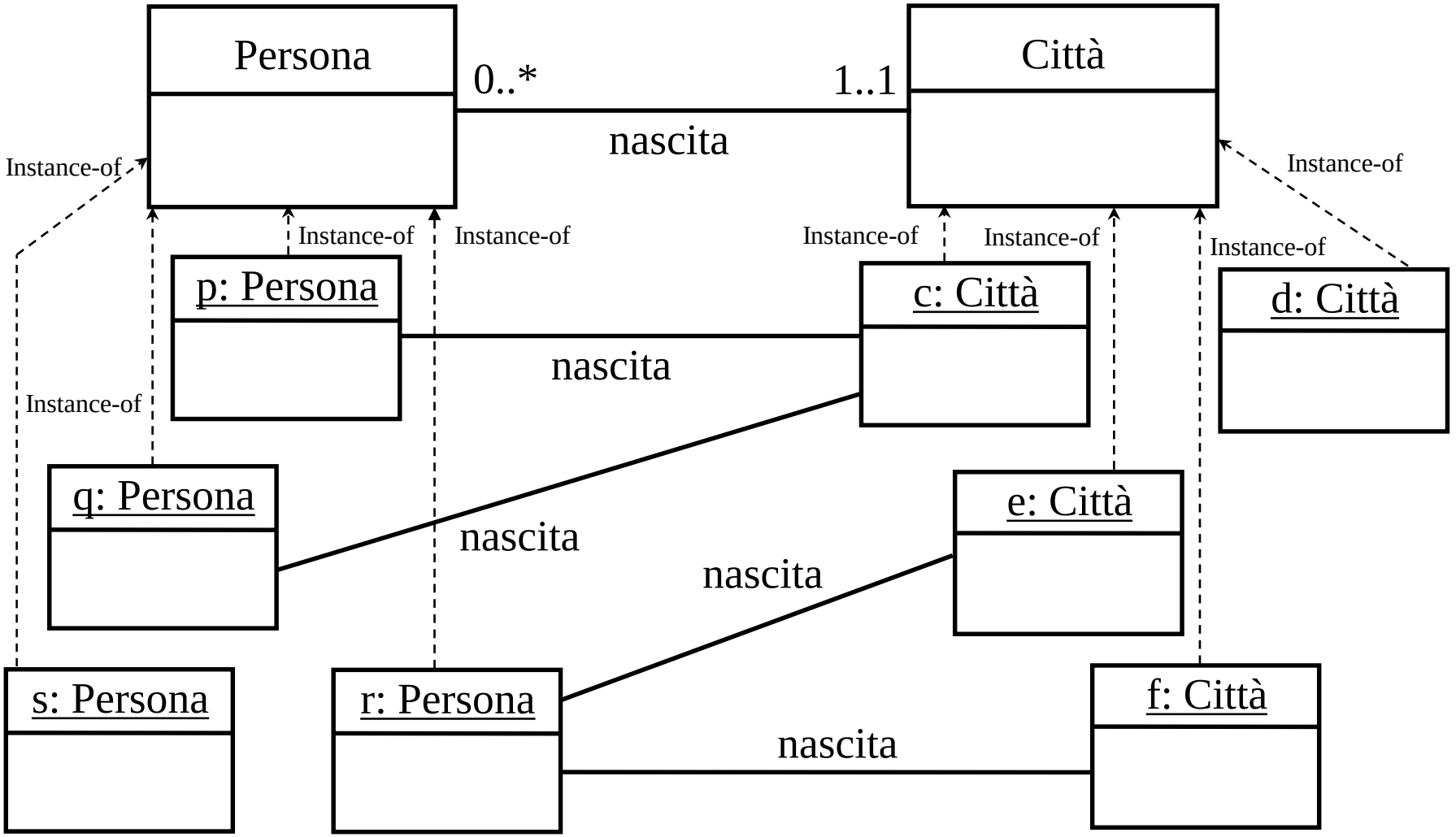
- Ogni studente è iscritto ad un corso di laurea
- Ogni corso di laurea ha almeno 10 iscritti
- Ogni studente può sostenere al massimo 28 esami

Esercizio 10

Con riferimento al diagramma delle classi precedente, è possibile:

1. Che uno studente non sia iscritto ad alcun corso di laurea?
2. Che uno studente sia iscritto a due corsi di laurea differenti?
3. Che ci siano esattamente 11 studenti, tutti iscritti allo stesso corso di laurea?
4. Che ci siano esattamente 3 studenti?
5. Che ci siano esattamente 10 studenti?

Esercizio 11: individuare gli errori



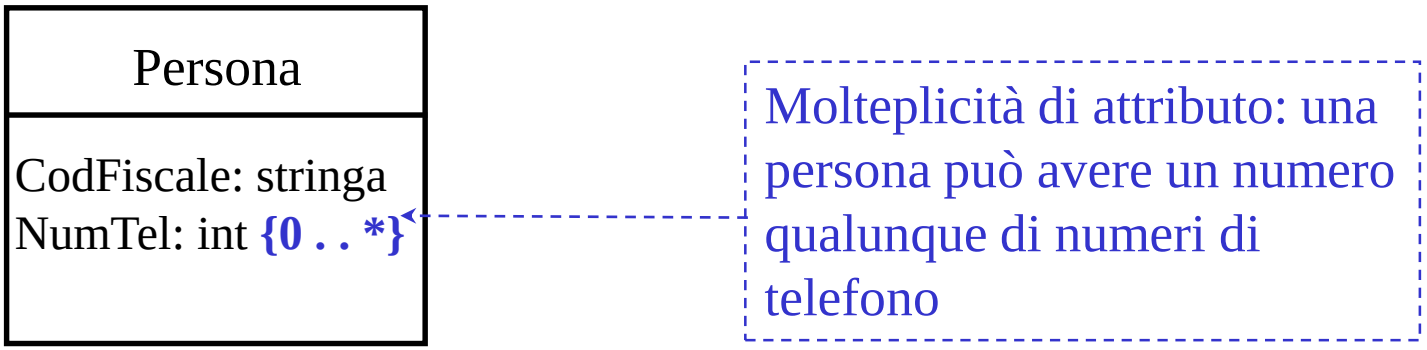
Esercizio 11 (soluzione)

Molteplicità di attributi

Si possono specificare anche le molteplicità degli attributi.

Se le molteplicità di un attributo *B* di tipo *T* di una classe *C* non vengono indicate, vuol dire che *B* associa ad ogni istanza di *C* esattamente un valore di *T* (come detto prima), che è equivalente a dire che la molteplicità è **1..1**

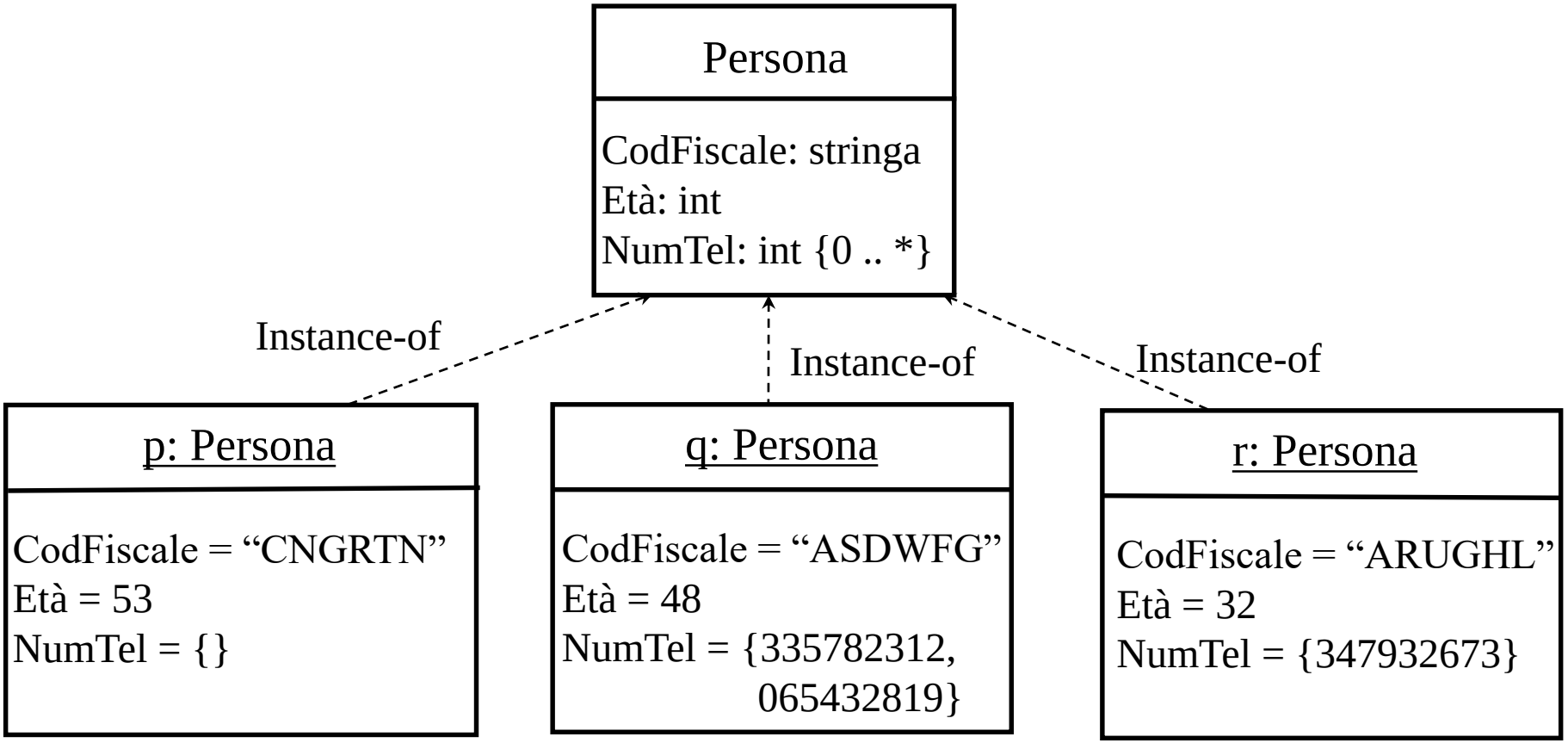
Se un attributo *B* di tipo *T* di una classe *C* ha molteplicità ***x .. y***, allora *B* associa ad ogni istanza di *C* al minimo *x* e al massimo *y* valori di tipo *T* ed indicato come in figura



Un attributo di tipo *T* della classe *C* con molteplicità diversa da {1..1} si dice **multivalore**, e formalmente non è una funzione totale, ma una relazione tra la classe *C* ed il tipo *T*

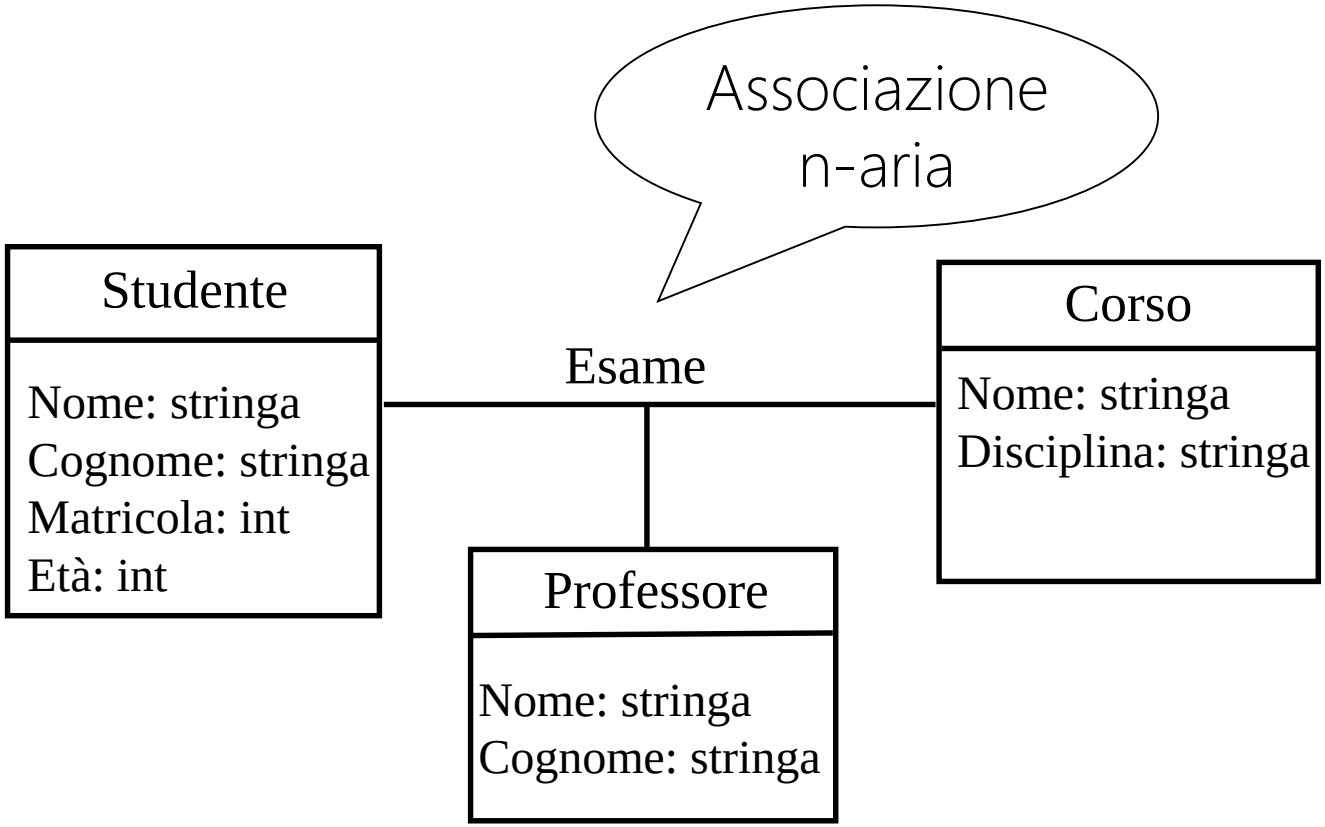
Attributi multivalore nelle istanze

Nelle istanze, il valore di un attributo multivalore si indica mediante un insieme



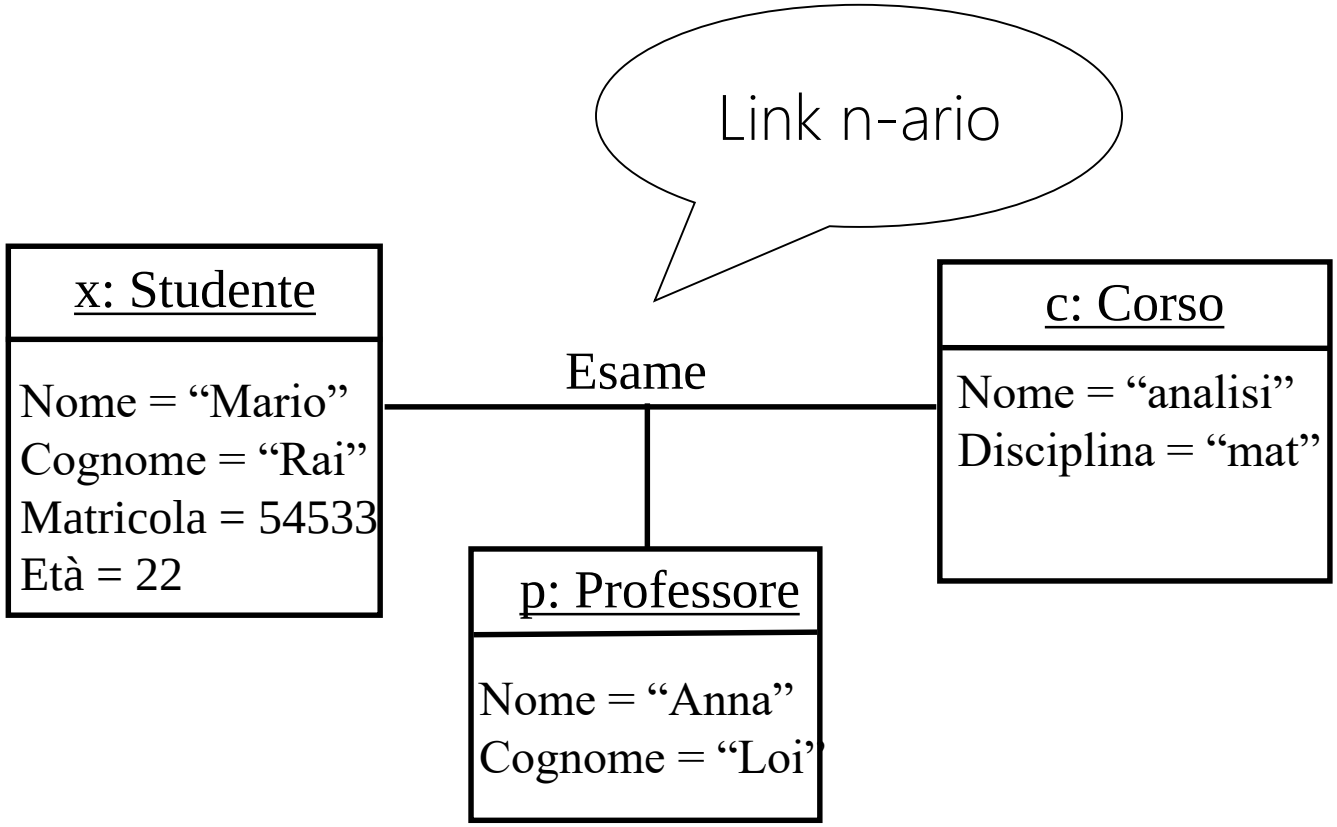
Associazioni n-arie

Una associazione può essere definita su tre o più classi. In tale caso l'associazione si dice **n-aria**, e modella una **relazione matematica tra n insiemi**



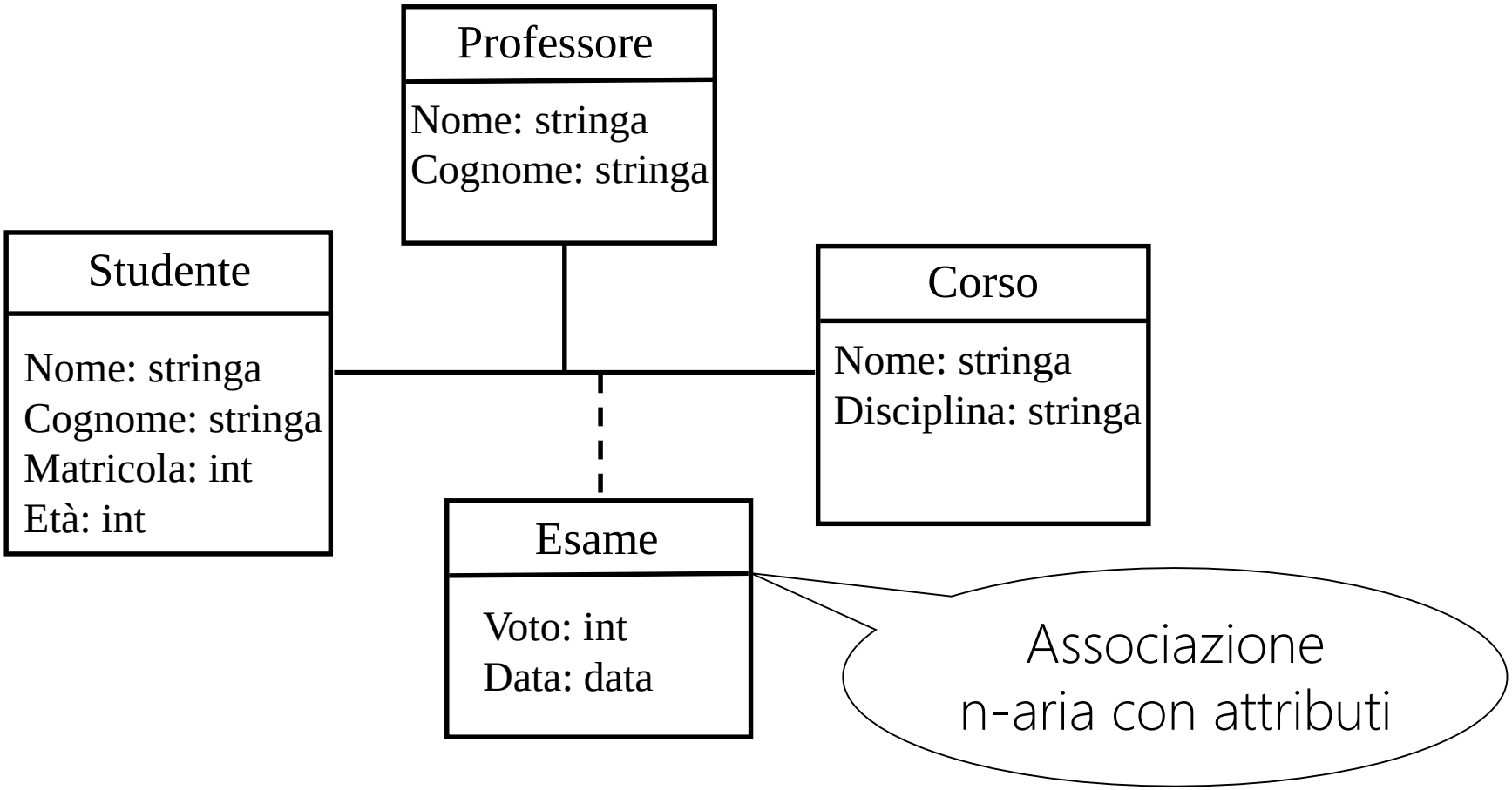
Istanze di associazioni n-arie: link n-ari

Ogni istanza di una associazione n-aria è un **link n-ario**, cioè che coinvolge **n oggetti** (***è una ennupla***)



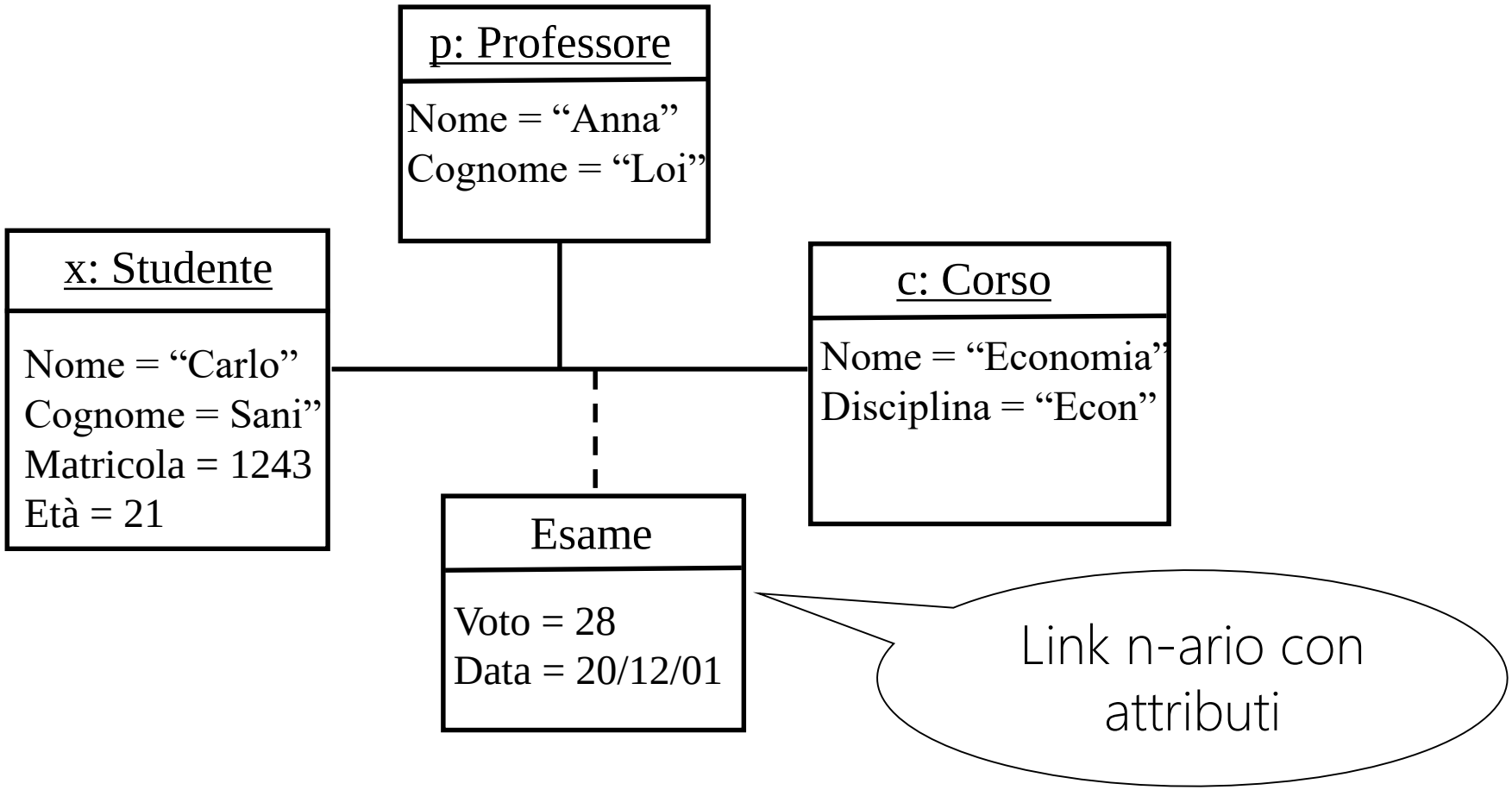
Associazioni n-arie con attributi

Ovviamente, anche le associazioni n-arie possono avere attributi



Link n-ari con attributi

I link che sono istanze di associazioni n-arie con attributi, hanno un valore per ogni attributo



Associazioni n-arie e molteplicità

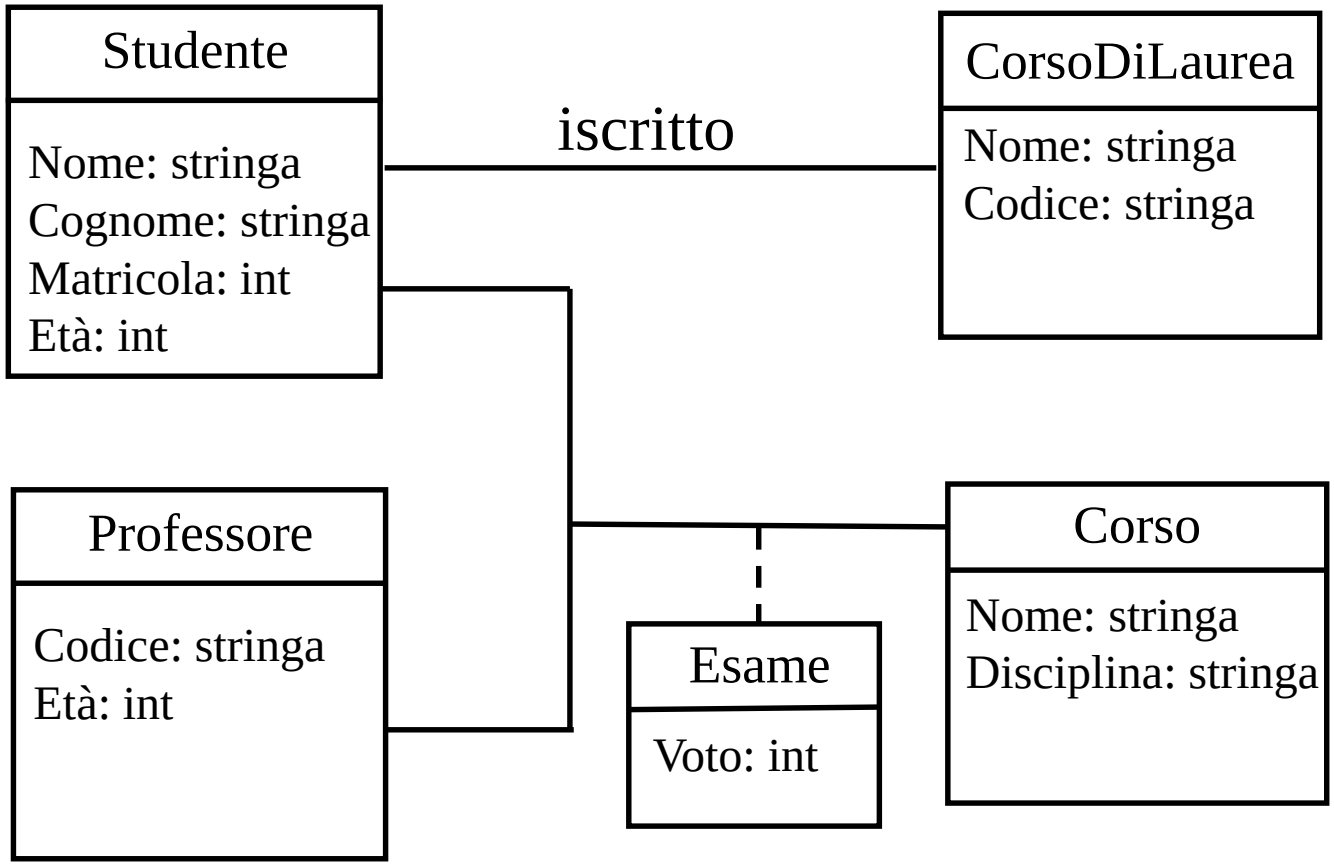
- Ci sono diversi vincoli di molteplicità che sarebbe di interesse esprimere su associazioni n-arie (*vedi Corso di Basi di Dati*)…
- …tuttavia noi in questo corso non li studieremo in modo specifico, ne considereremo la notazione per esprimerli in UML.
- Qualora avessimo bisogno di specificare un vincolo di molteplicità lo faremo in linguaggio testuale con un commento.

Esempio

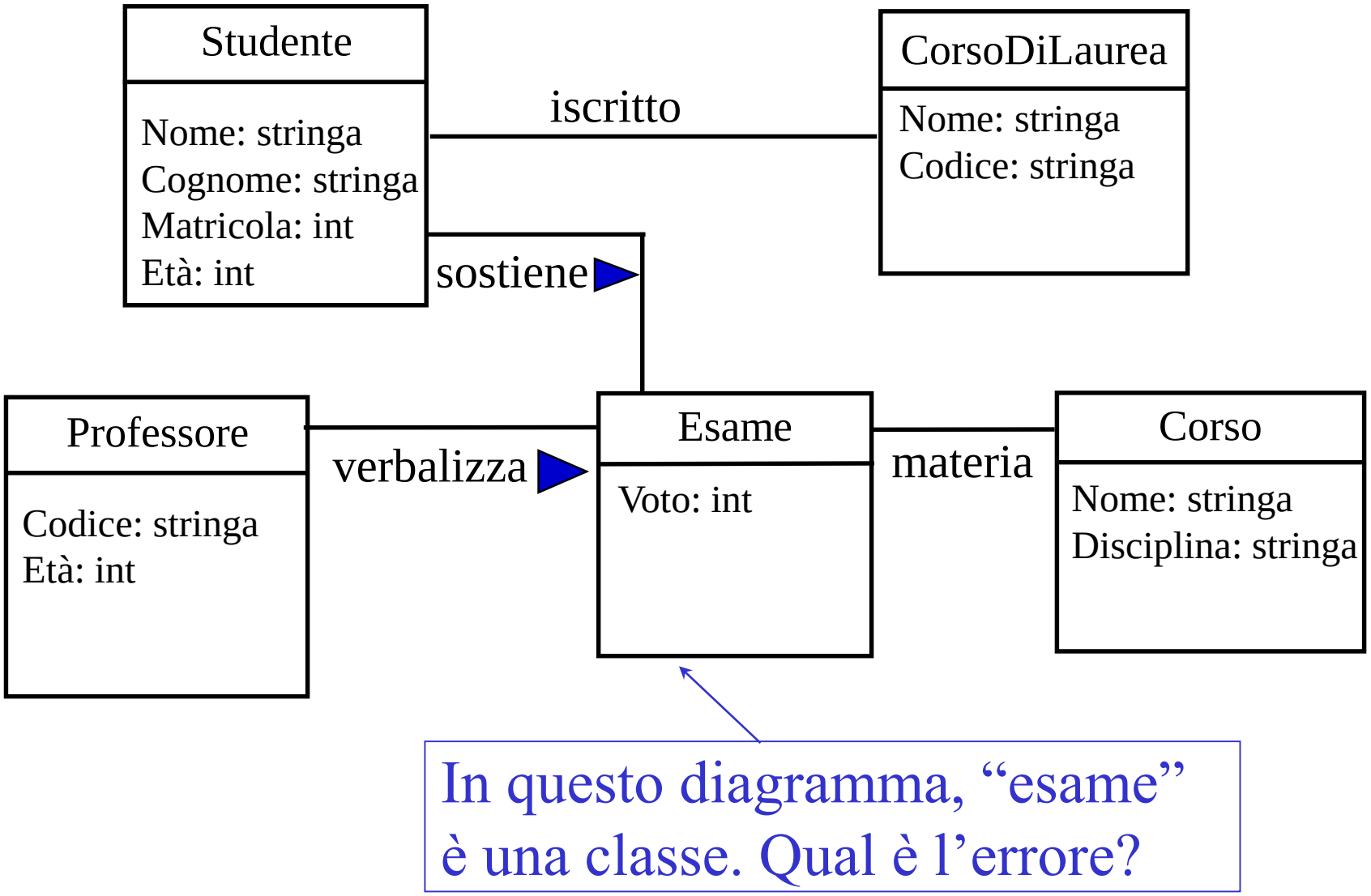
Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare gli studenti (con nome, cognome, numero di matricola, età), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame, con il professore che ha verbalizzato l'esame, ed il voto conseguito. Di ogni corso di laurea interessa il codice e il nome. Di ogni corso interessa il nome e la disciplina a cui appartiene (ad esempio: matematica, fisica, informatica, ecc.). Di ogni professore interessa codice ed età.

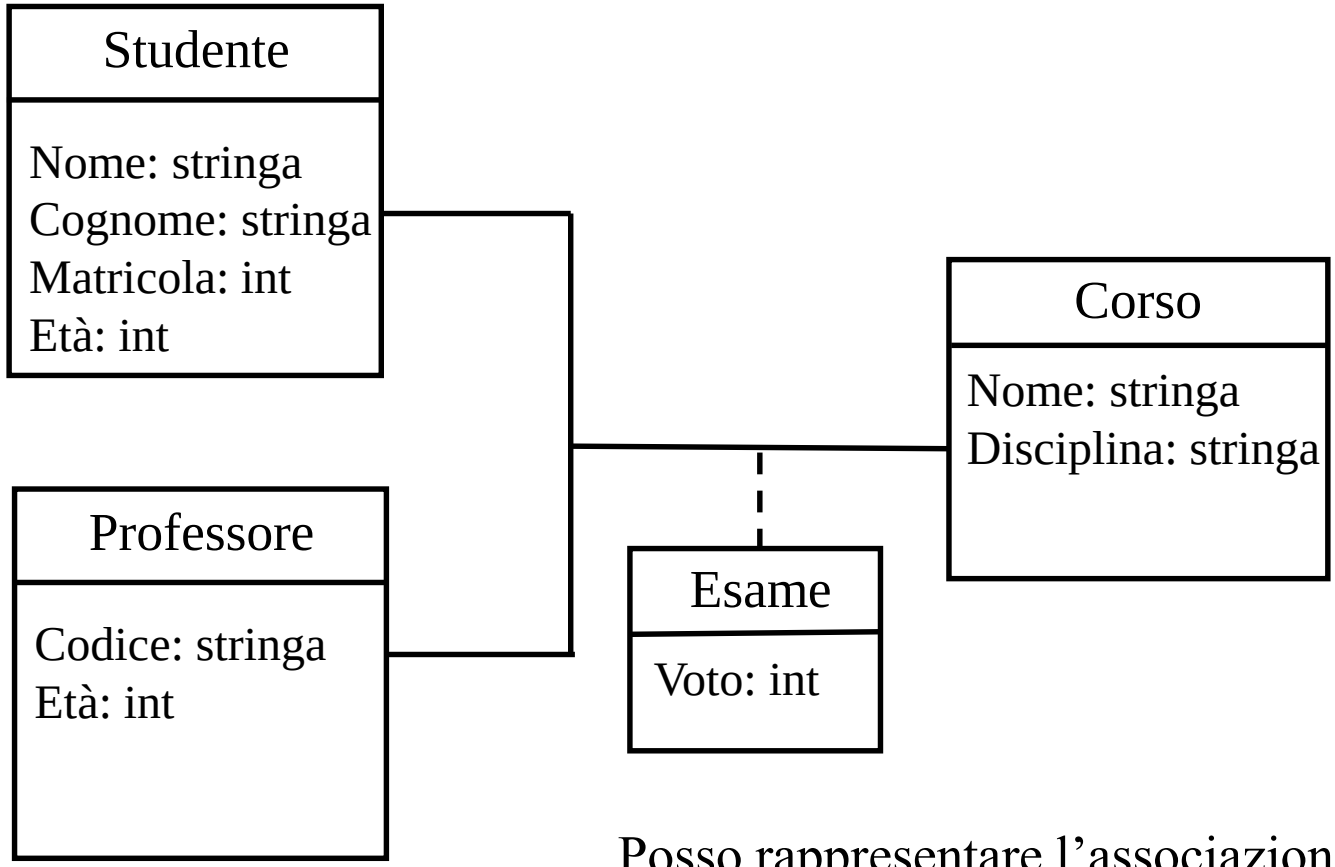
Diagramma delle classi per l'esempio



Soluzione scorretta per l'esempio



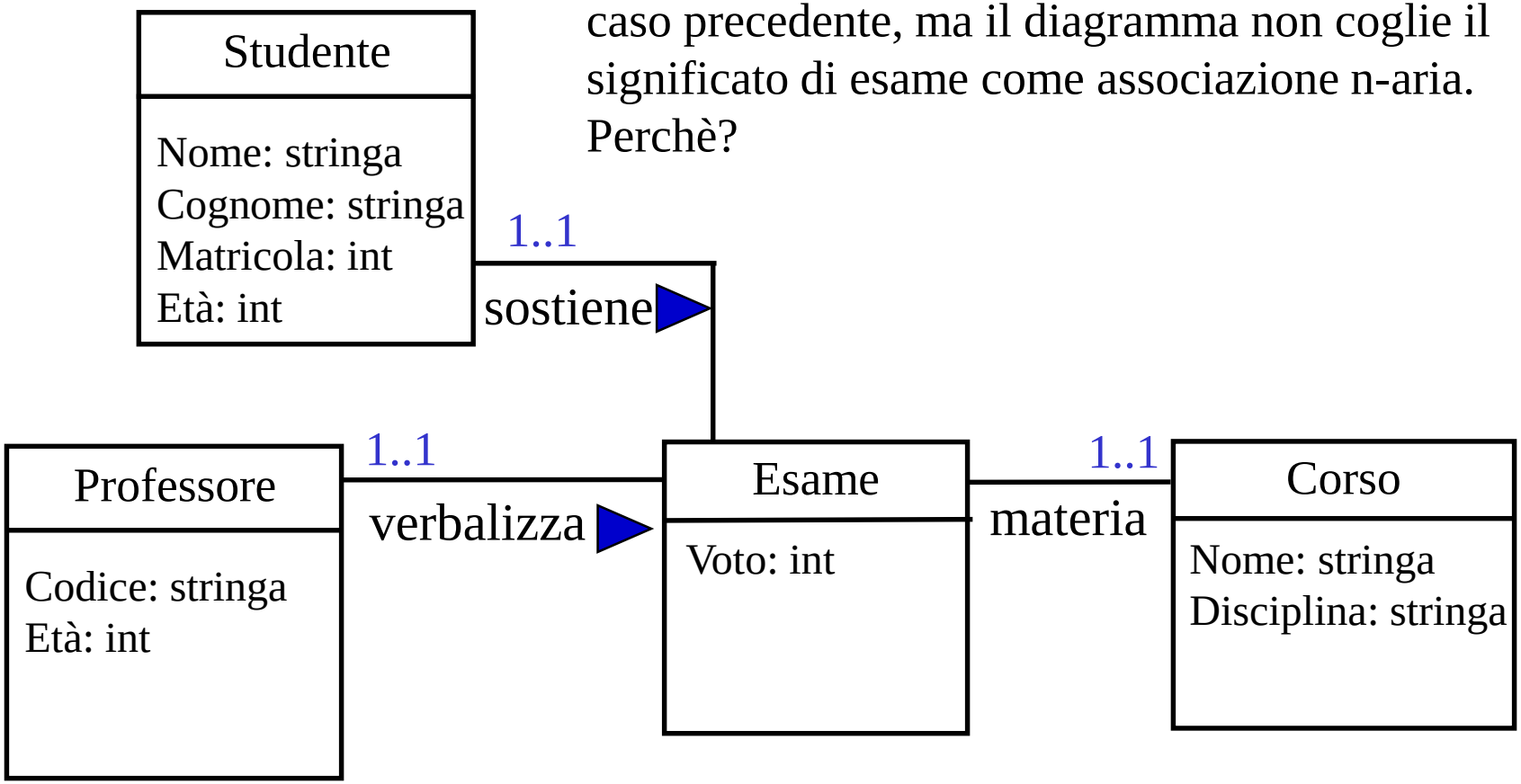
Torniamo alla associazione n-aria



Posso rappresentare l'associazione esame con una classe facendo uso di opportuni vincoli di molteplicità?

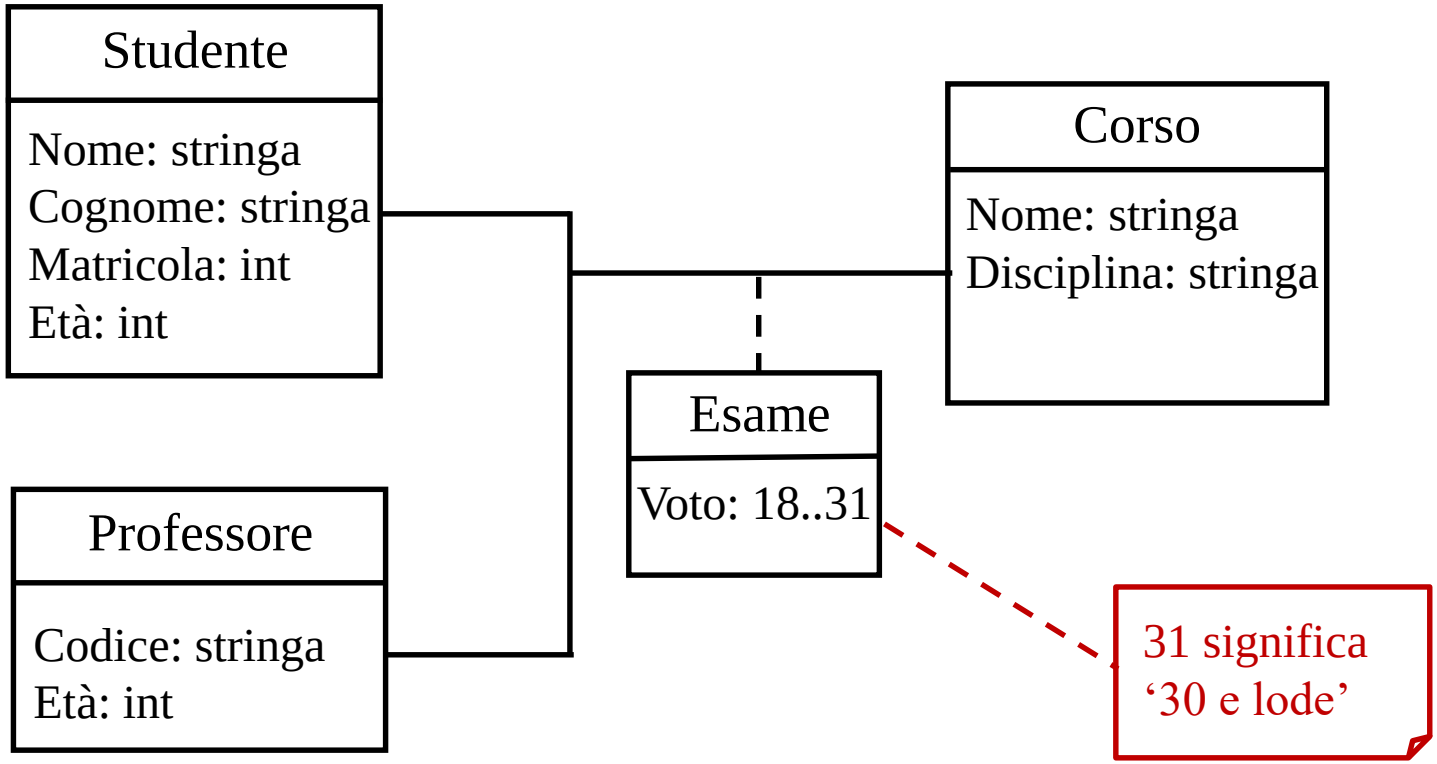
Soluzione scorretta

In questo diagramma, “esame” è una classe.
Le molteplicità migliorano la situazione rispetto al caso precedente, ma il diagramma non coglie il significato di esame come associazione n-aria.
Perchè?



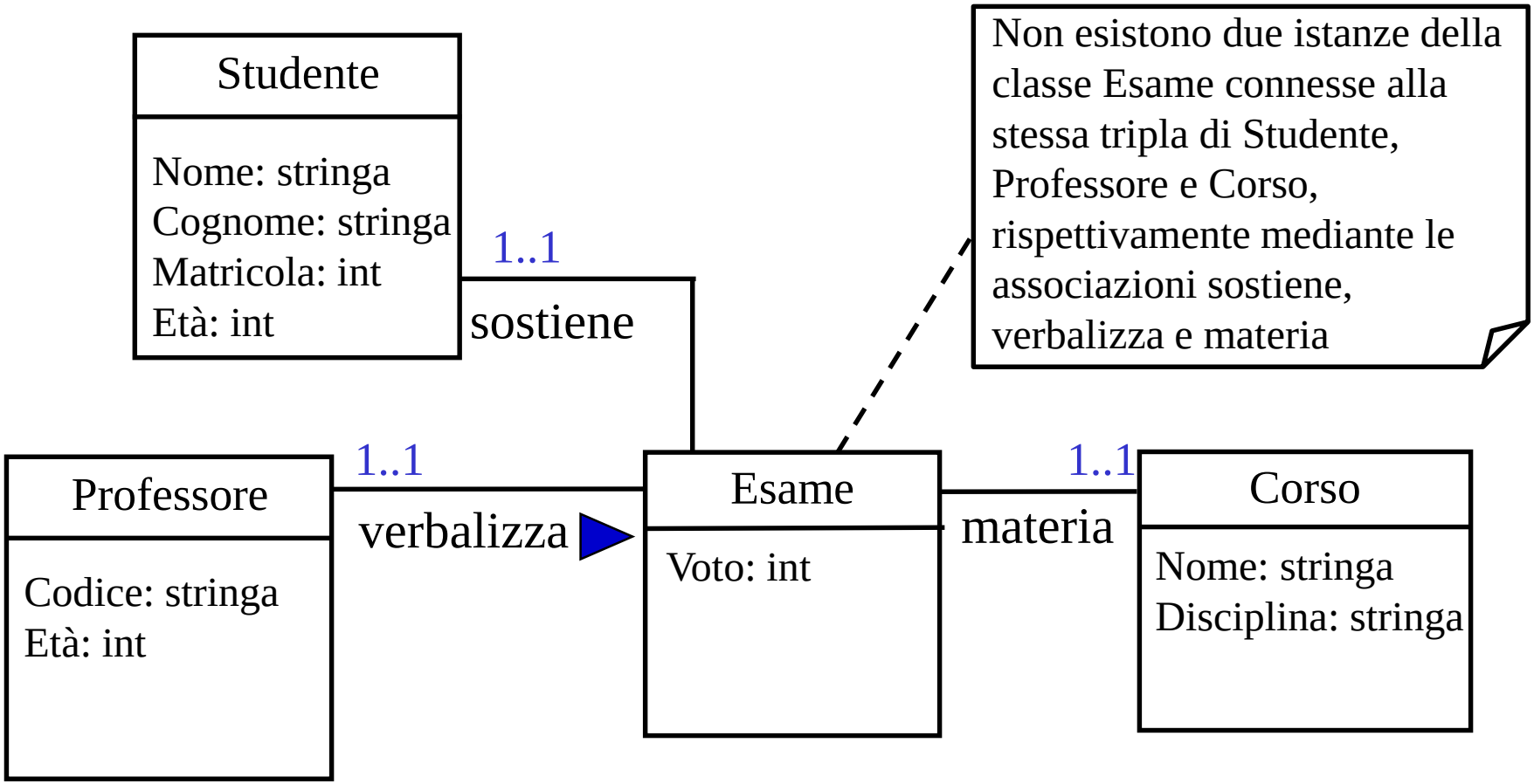
Commenti in UML

In UML, quando si vuole specificare una caratteristica che non è possibile rappresentare esplicitamente nel diagramma con i meccanismi visti finora, si può usare la nozione di **commento**



Esempio di uso dei commenti

In questo modo, il diagramma modella il concetto di Esame in modo equivalente ad una associazione n-aria tra Studente, Professore e Corso

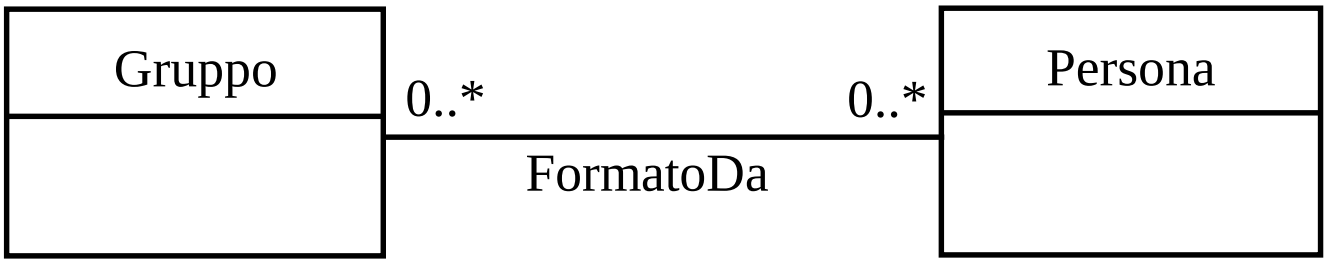


Associazioni ordinate

Supponiamo di voler descrivere gruppi di persone …

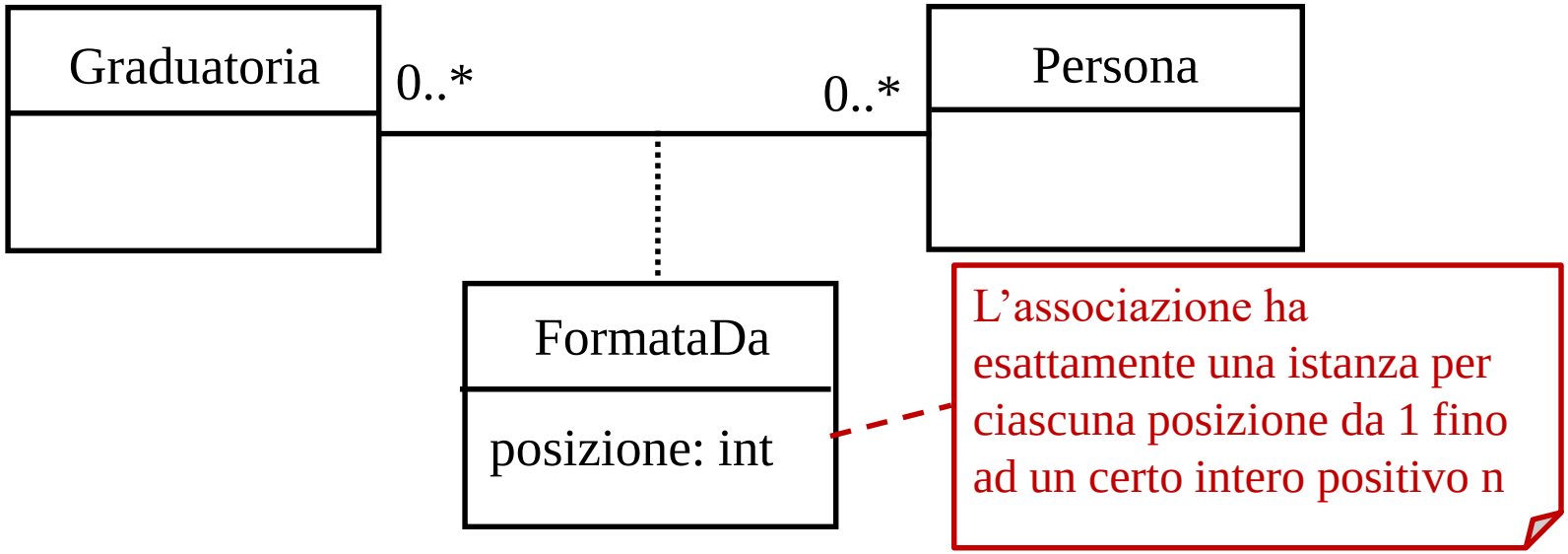
Un gruppo è formato da persone. Ogni persona può apparire in un gruppo al più una volta (ovviamente ciascuna persona può fare parte di 0, 1, molti gruppi)

In UML possiamo rappresentare questo scenario come segue:



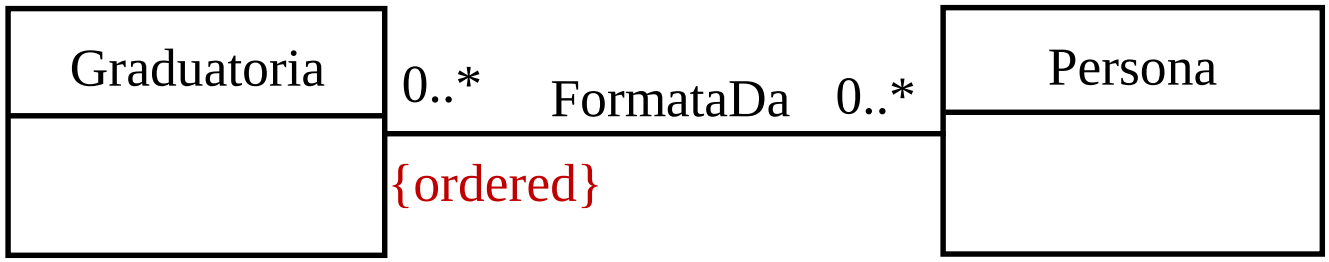
Associazioni ordinate (cont.)

- Consideriamo ora invece una graduatoria di persone…
- Una graduatoria ha un certo numero di posizioni ciascuno occupato da una sola persona *(non consideriamo pari merito in questo esempio)* e una persona può apparire in una graduatoria al più una volta.
- Una possibile rappresentazione in UML è:



Associazioni ordinate (cont.)

- La situazione descritta nell'esempio Graduatoria è molto comune, si pensi alla scaletta di un concerto, ad una presentazione formata da una sequenza di slide, ecc.
- L'attributo posizione serve **solo** a mantenere questo ordine (e per svolgere il suo lavoro deve sempre rispettare il vincolo del commento)
- In UML si può semplificare la descrizione utilizzando l'asserzione **{ordered}**



{ordered} posto vicino a Graduatoria dice che data una istanza ***g*** di Graduatoria le istanze della associazione FormataDa che coinvolgono ***g*** sono ordinate (senza menzionare quale attributo utilizziamo per mantenere l'ordine).

Associazioni ordinate (cont.)

La soluzione con **{ordered}** è da preferire alla soluzione con un attributo esplicito "posizione" perché:

- è più semplice (non fa uso di vincoli esterni - espressi nei commenti) ed è quindi **più leggibile**
- **astrae** da come verrà mantenuta l'informazione sull'**ordine** evitando di introdurre uno specifico attributo ("posizione") necessario a questo scopo

Esercizio 12

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare delle playlist costituite da un nome (una stringa) e da un elenco di brani ***eventualmente ripetuti****. Ciascun brano è caratterizzato dal nome (una stringa), la durata (un reale) e il nome del file (una stringa)*

Esercizio 12 (soluzione)

Generalizzazione in UML

Fino ad ora abbiamo assunto che due classi siano sempre disgiunte. In realtà sappiamo che può accadere che tra due classi sussista la relazione **is-a**, e cioè che **ogni istanza di una sia anche istanza dell'altra**.

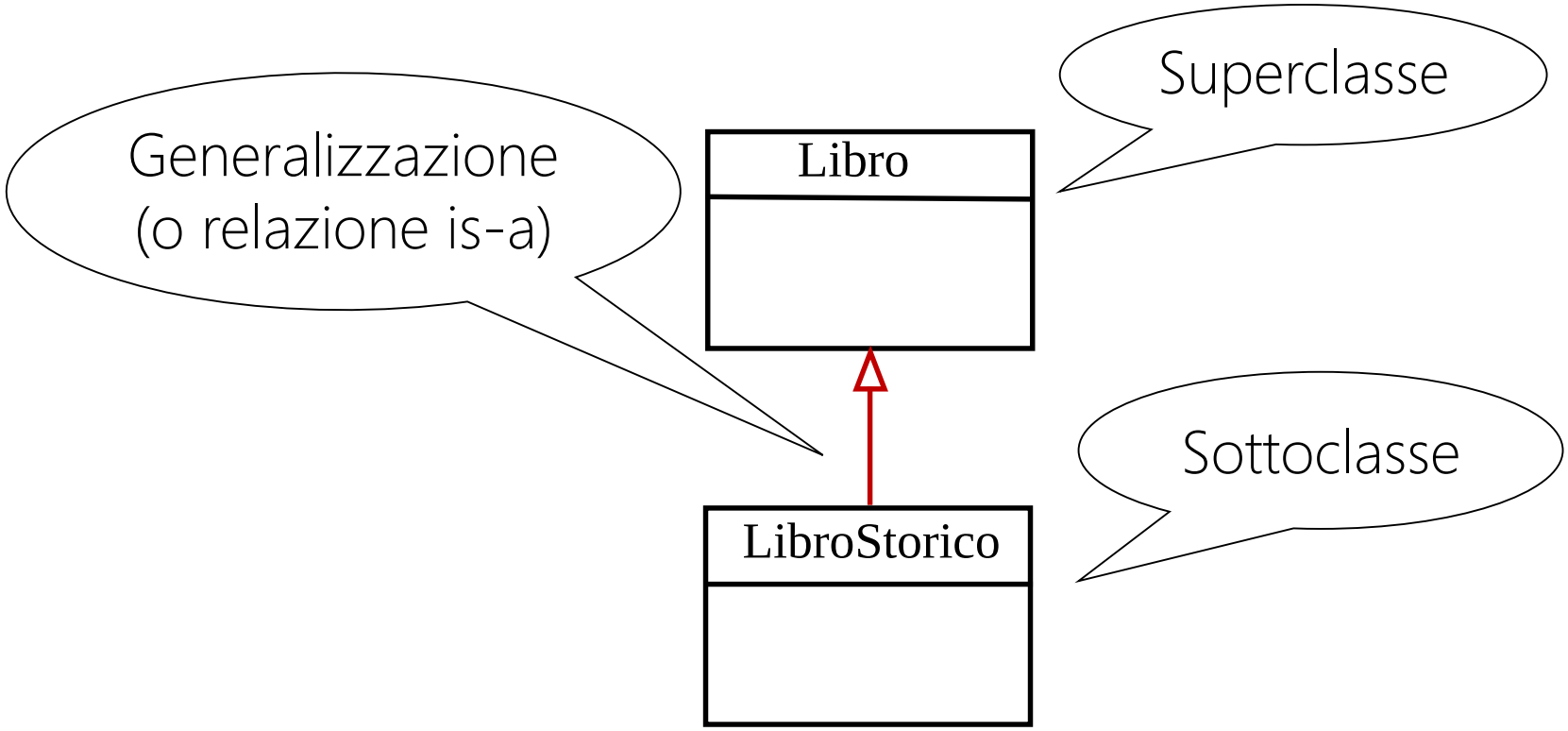
In UML la relazione **is-a** si modella mediante la nozione di **generalizzazione**

La **generalizzazione** coinvolge una superclasse ed una o più sottoclassi (dette anche **classi derivate**). Il significato della generalizzazione è il seguente: **ogni istanza di ciascuna sottoclasse è anche istanza della superclasse**

Quando la sottoclasse è una, la generalizzazione modella appunto la **relazione is-a** tra la sottoclasse e la superclasse

Generalizzazione in UML

Esempio di generalizzazione (siccome la generalizzazione coinvolge due classi, essa modella la relazione is-a)

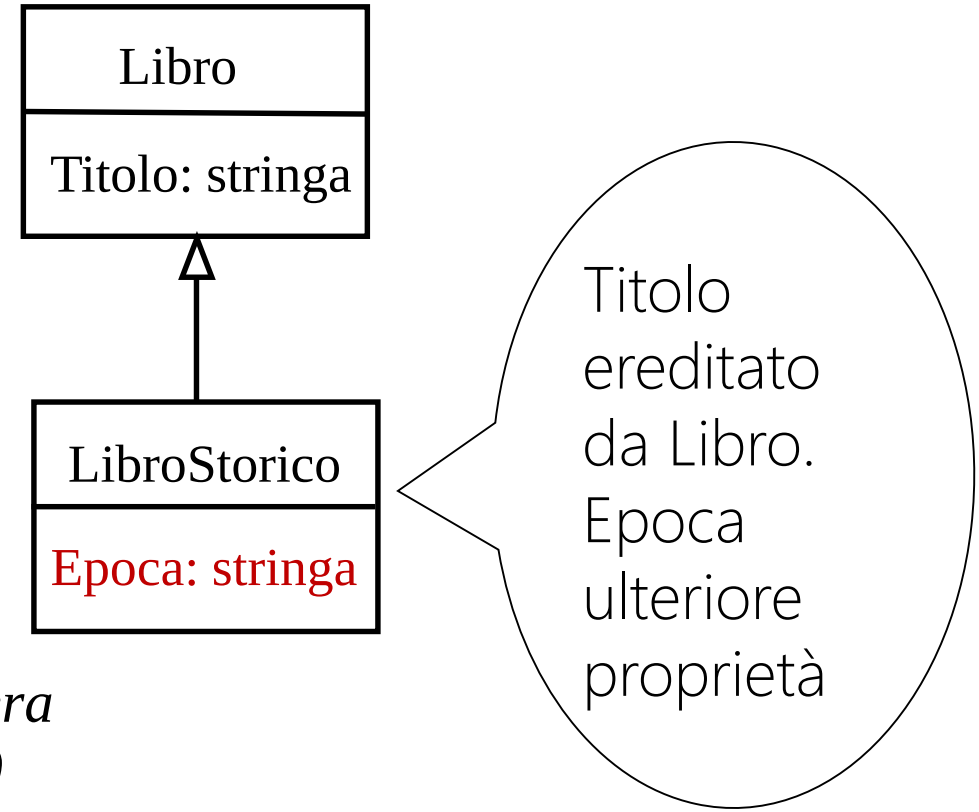


Ereditarietà in UML

Principio di ereditarietà: ogni proprietà della superclasse è anche una proprietà della sottoclasse, e **non** si riporta esplicitamente nel diagramma

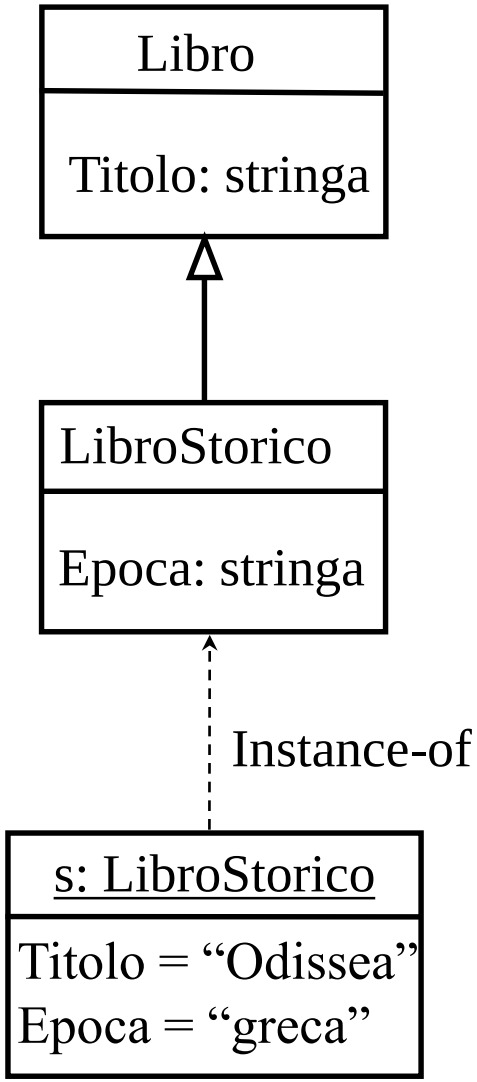
Dal fatto che
1. Ogni istanza di Libro ha un Titolo
2. Ogni istanza di LibroStorico è una istanza di Libro

segue logicamente che
3. Ogni istanza di LibroStorico ha un Titolo



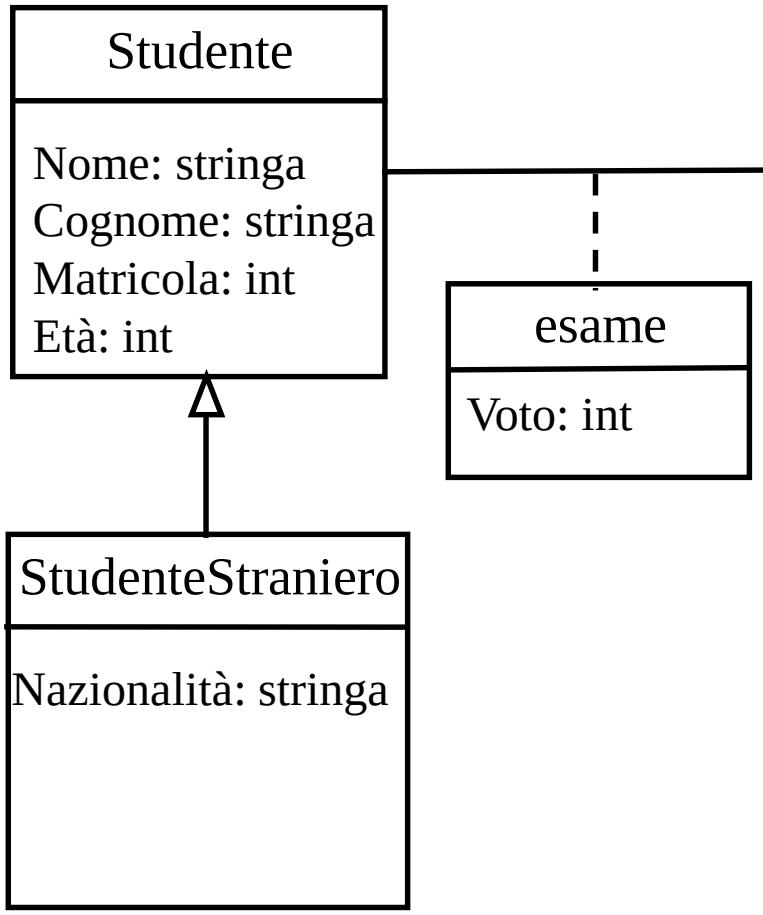
Ragionamento sillogistico (cfr. opera di Aristotele più di due millenni fa)

Ereditarietà in UML: istanze



- **s** è istanza sia di LibroStorico (classe più specifica) sia di Libro
- non è più vero che due classi diverse sono disgiunte: Libro e LibroStorico non sono ovviamente disgiunte
- resta comunque vero che ogni istanza ha una ed una sola classe più specifica di cui è istanza; in questo caso la classe più specifica di **s** è LibroStorico
- **s** ha un valore per tutti gli attributi di LibroStorico, sia quelli propri, sia quelli ereditati dalla classe Libro

Ereditarietà sulle associazioni

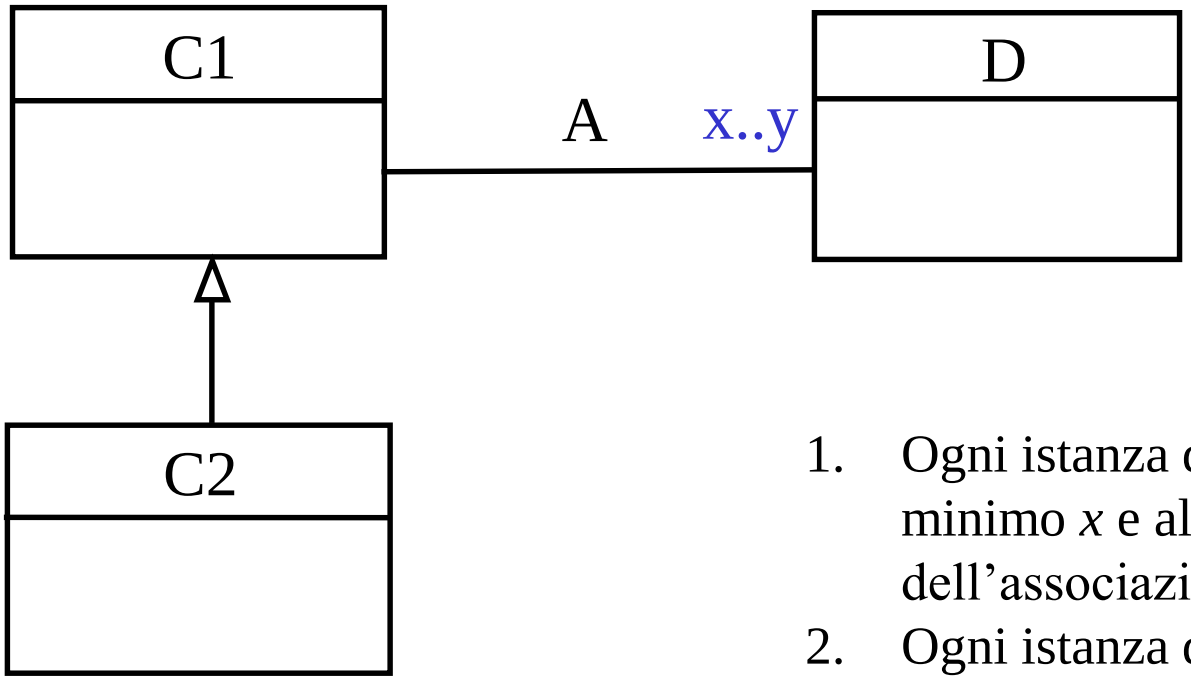


1. Ogni istanza di Studente può essere coinvolta in un numero qualunque di istanze della associazione “esame”
2. Ogni istanza di StudenteStraniero è una istanza di Studente

QUINDI

3. Ogni istanza di StudenteStraniero può essere coinvolta in un numero qualunque di istanze della associazione “esame”

Ereditarietà sulle molteplicità

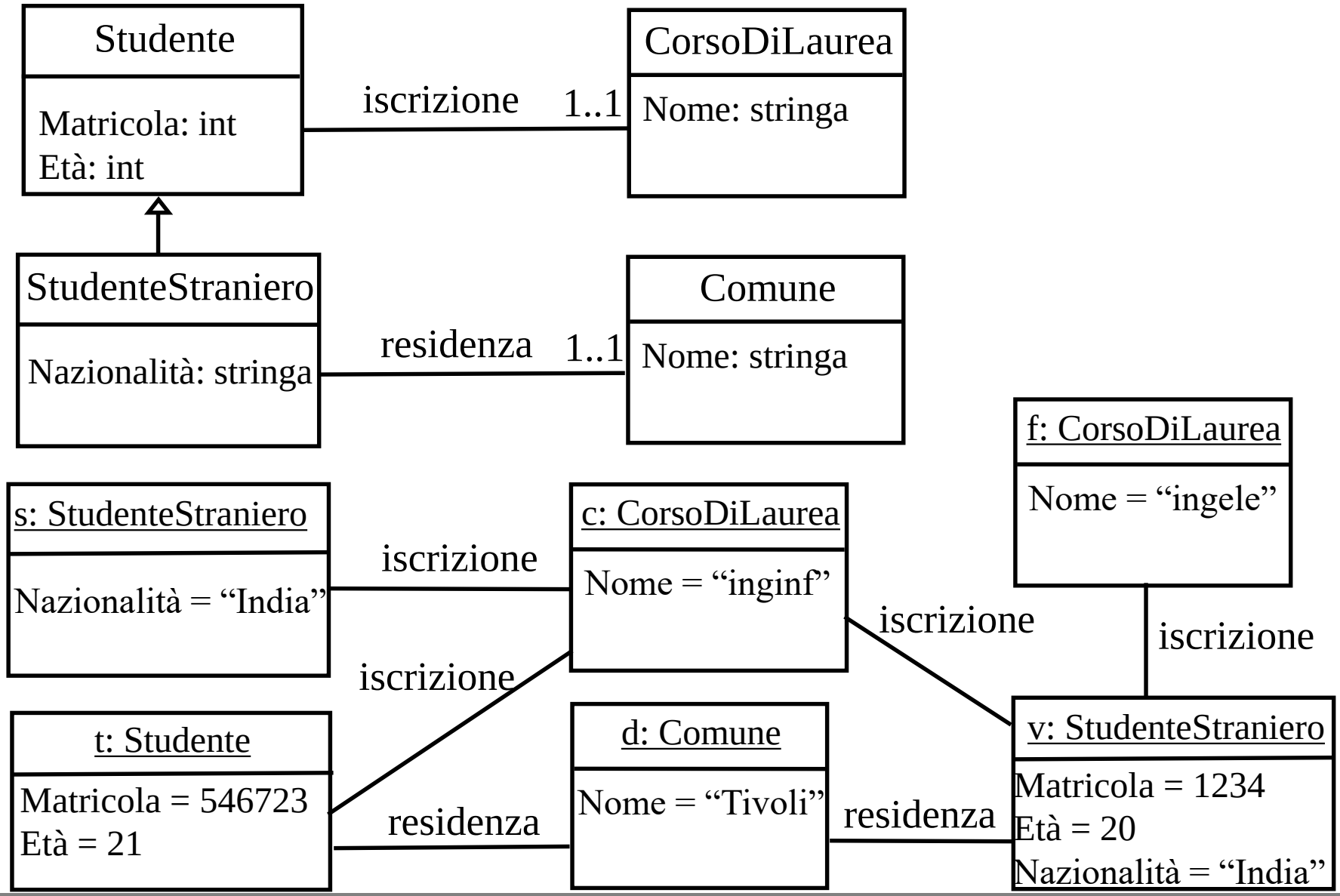


1. Ogni istanza di C_1 è coinvolta in al minimo x e al massimo y istanze dell'associazione A
2. Ogni istanza di C_2 è una istanza di C_1

QUINDI

3. Ogni istanza di C_2 è coinvolta in al minimo x e al massimo y istanze dell'associazione A

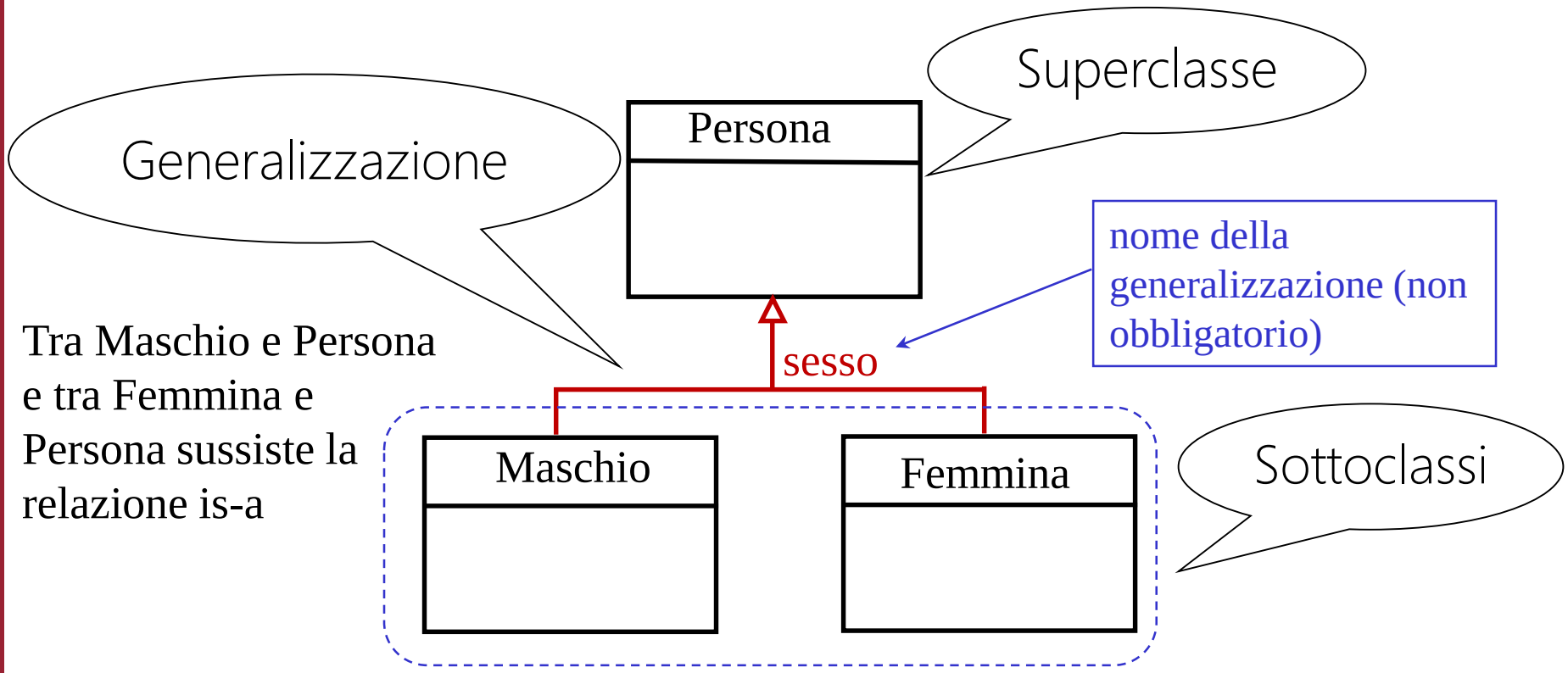
Esercizio 13: individuare gli errori



Esercizio 13 (soluzione)

Generalizzazione in UML

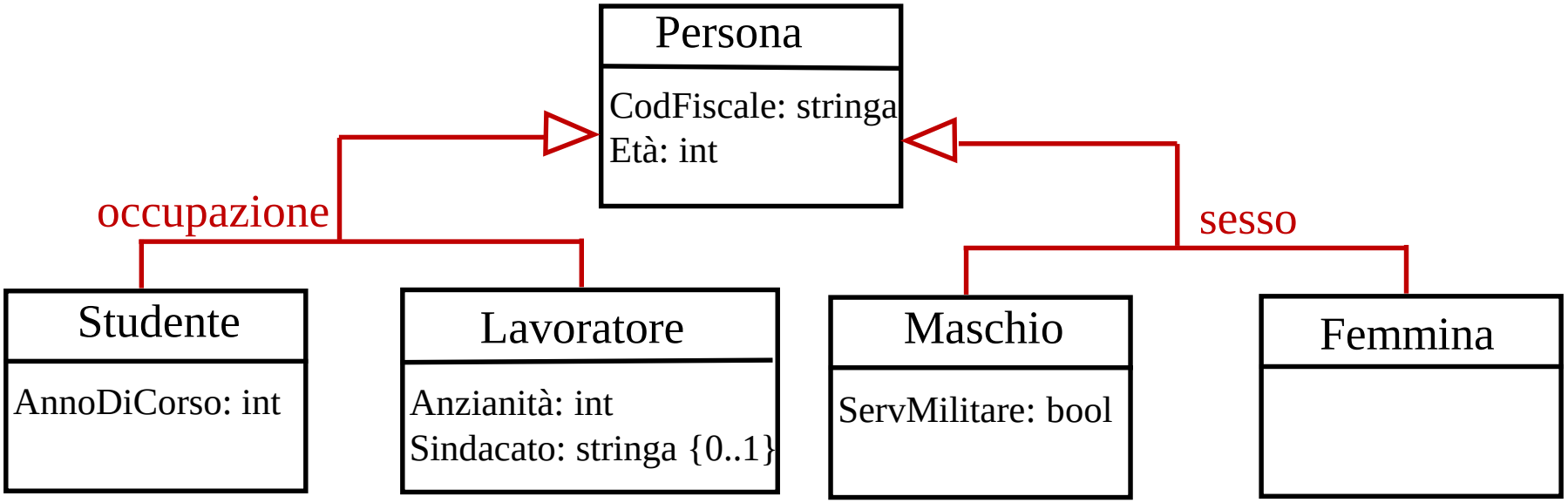
Finora, abbiamo considerato la generalizzazione come mezzo per modellare la relazione is-a tra due classi. La superclasse però può anche generalizzare diverse sottoclassi rispetto ad un unico criterio (che si può indicare con un nome, che è il nome della generalizzazione)



Tra Maschio e Persona e tra Femmina e Persona sussiste la relazione is-a

Diverse generalizzazioni della stessa classe

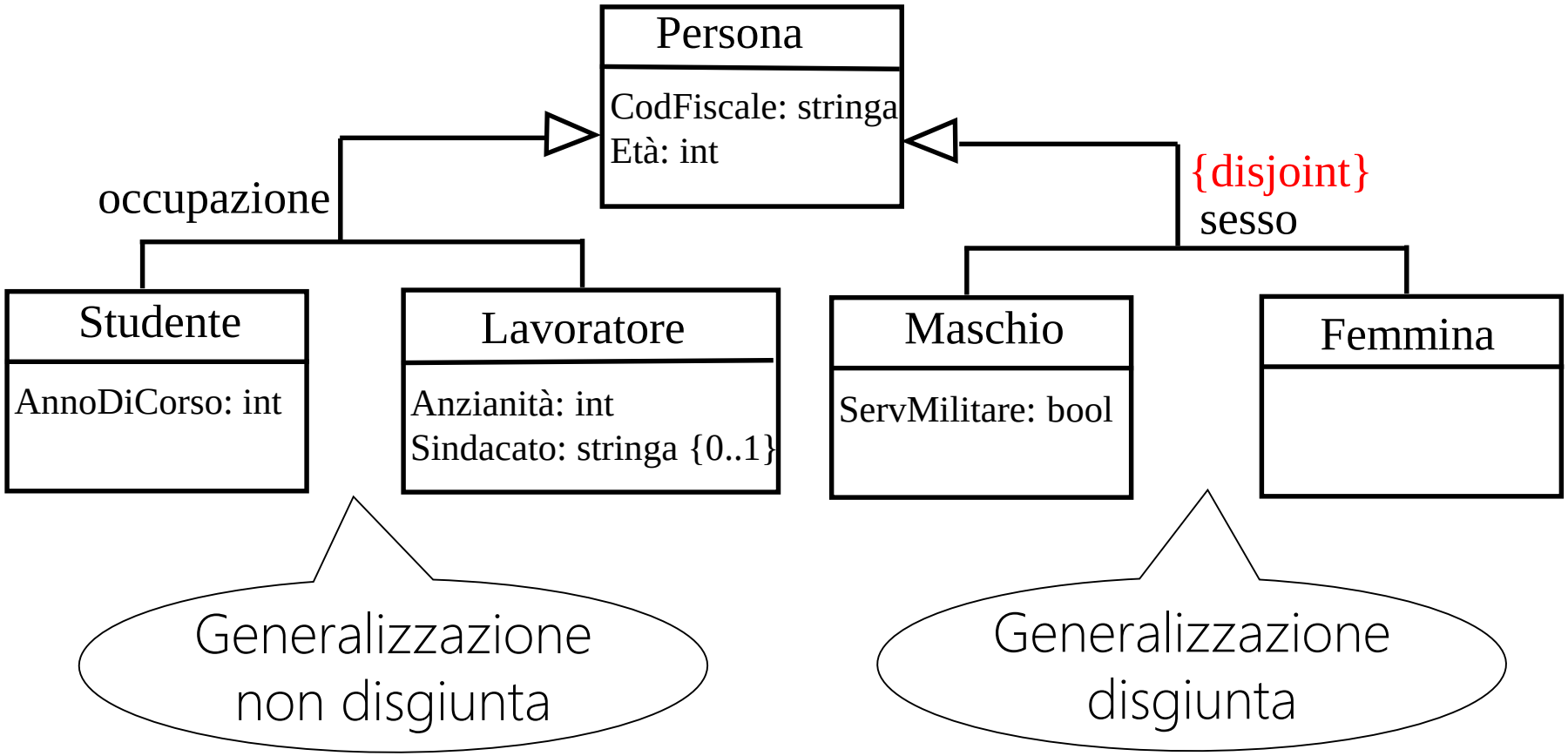
La stessa superclasse può partecipare a diverse generalizzazioni



Concettualmente, non c'è alcuna correlazione tra due generalizzazioni diverse, perchè rispondono a due criteri diversi di classificare le istanze della superclasse

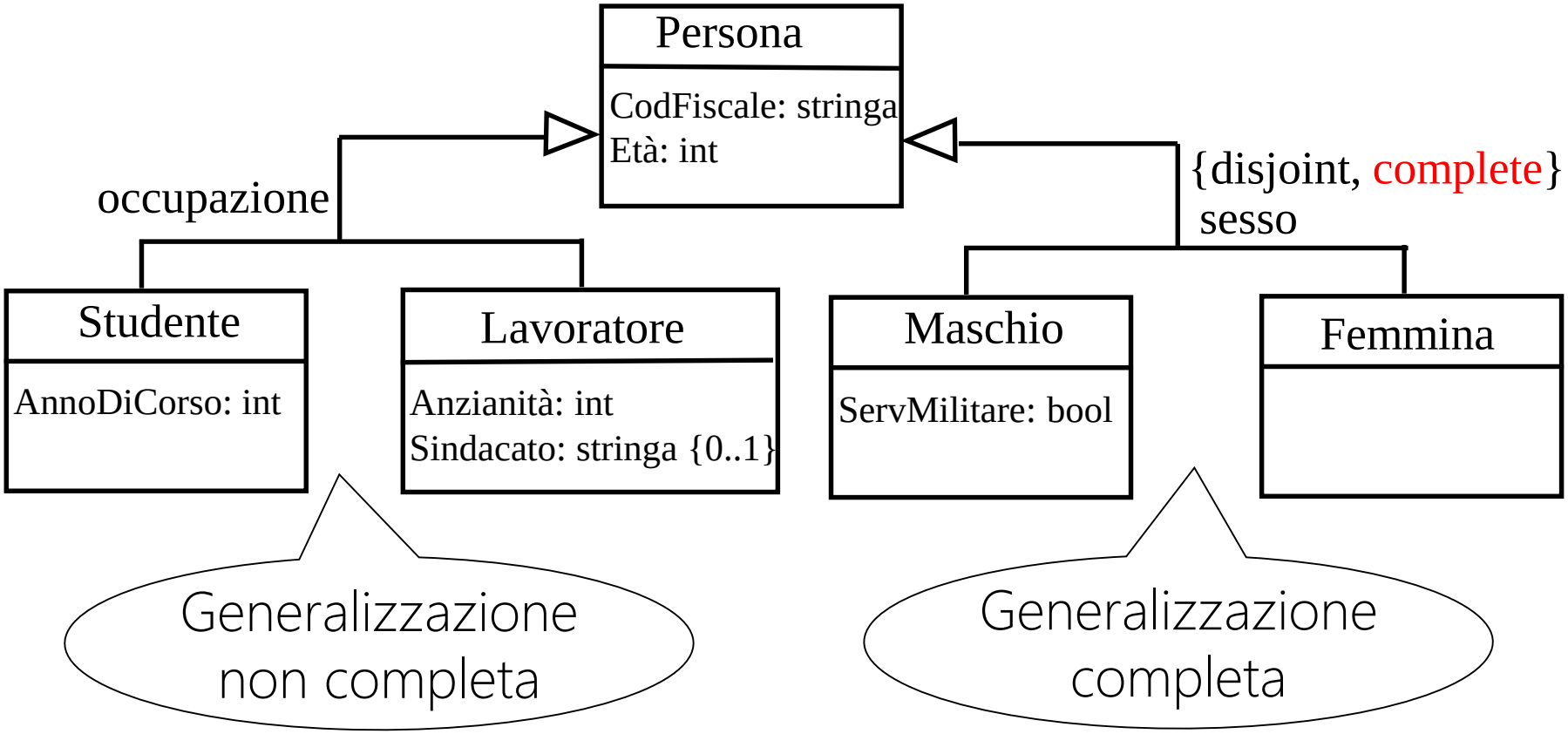
Generalizzazioni disgiunte

Una generalizzazione può essere disgiunta (le sottoclassi sono **disgiunte a coppie**) o no

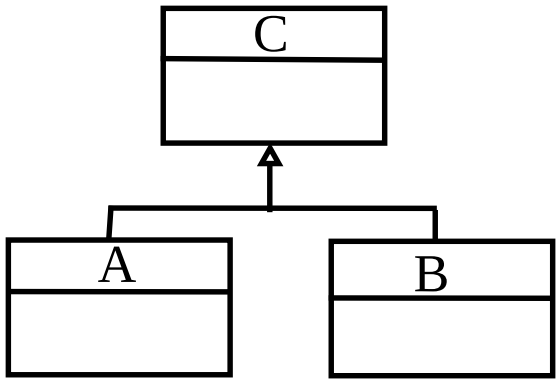
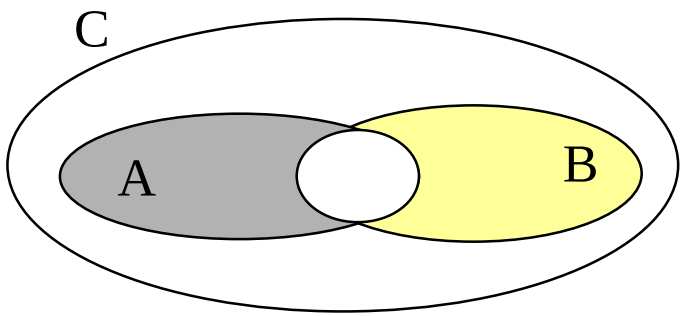
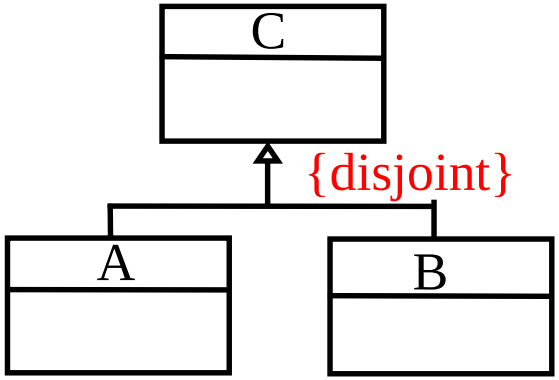
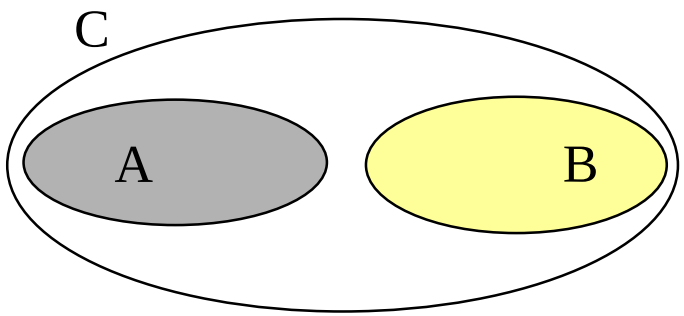


Generalizzazioni complete

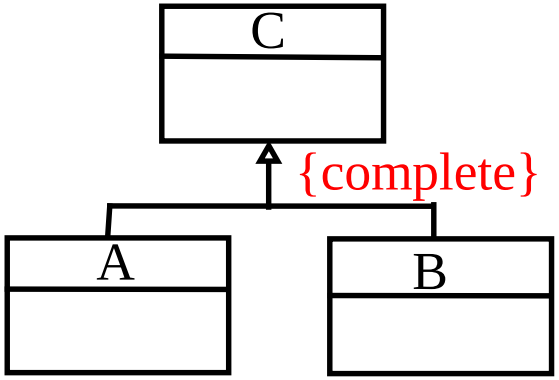
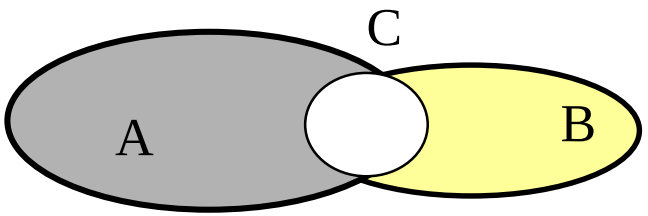
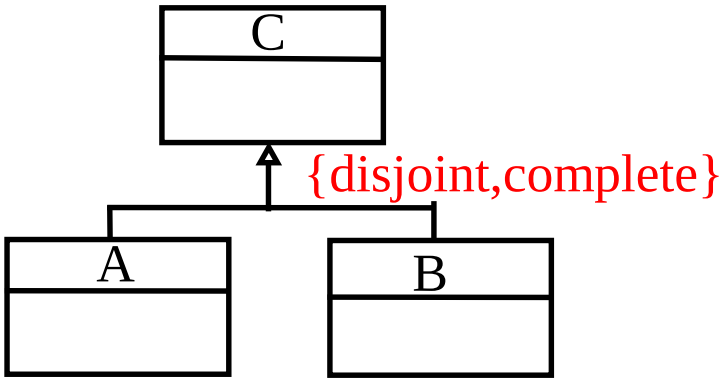
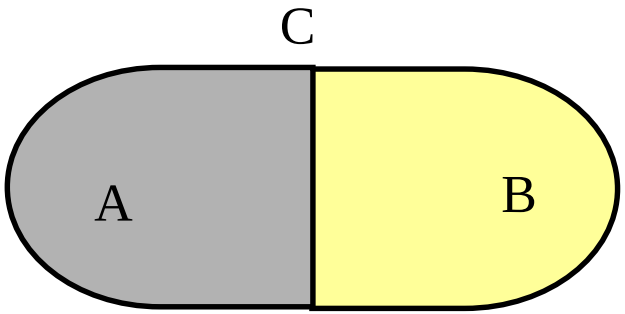
Una generalizzazione può essere completa (l'unione delle istanze delle sottoclassi è uguale all'insieme delle istanze della superclasse) o no



Generalizzazioni

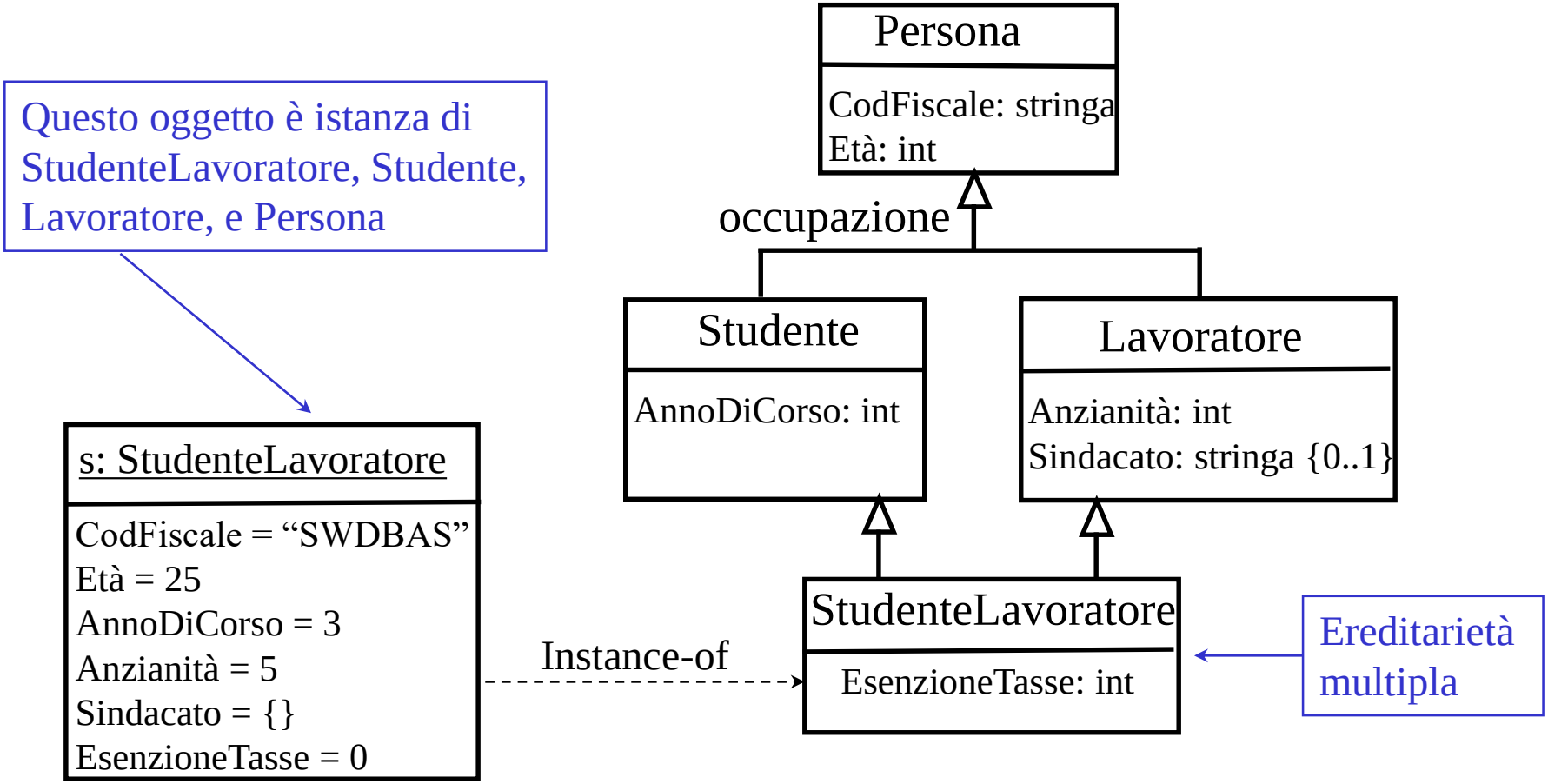
Livello intensionale	Livello estensionale
	
	

Generalizzazioni

Livello intensionale	Livello estensionale
	
	

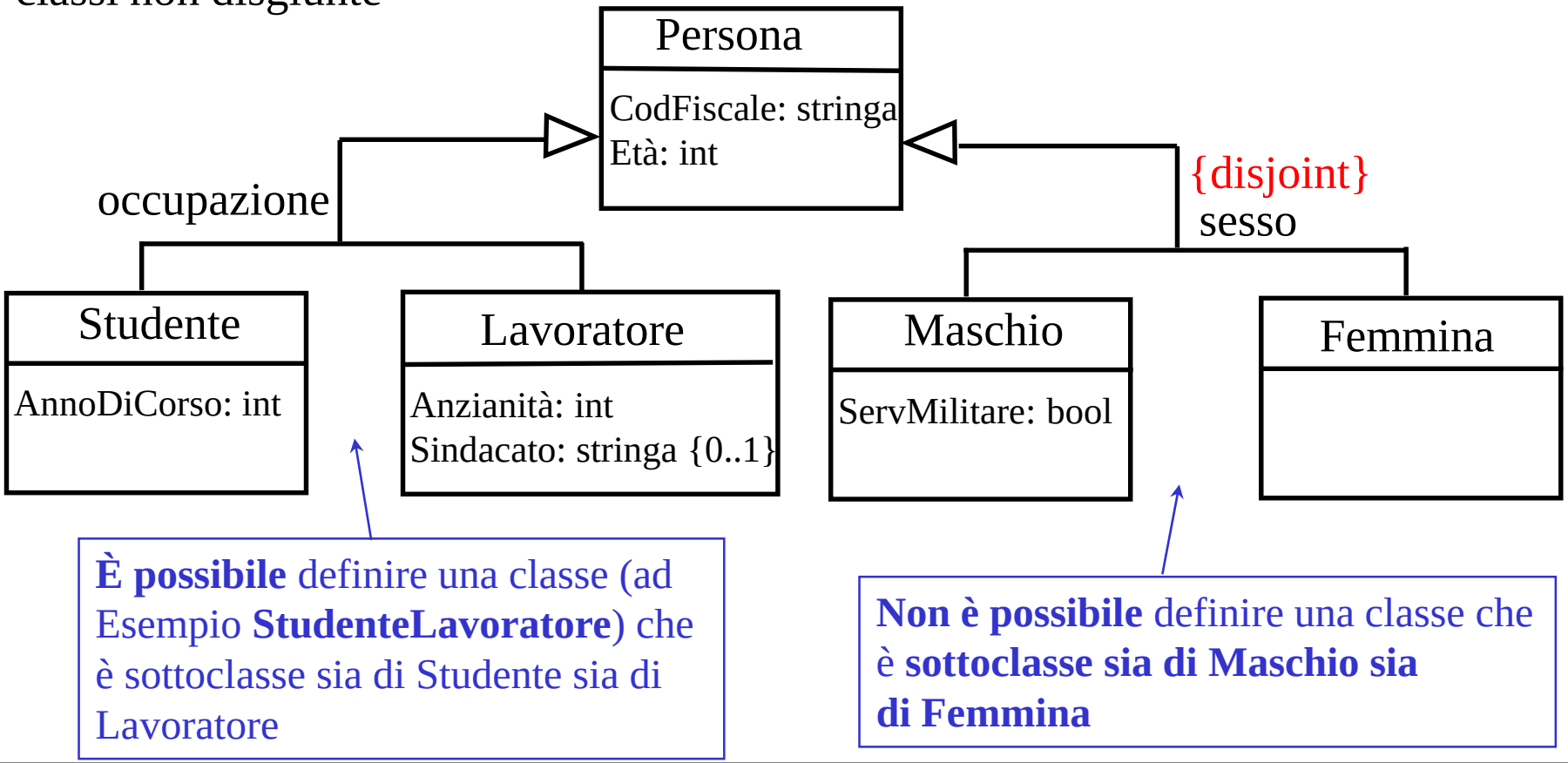
Ereditarietà multipla

Attenzione: poichè un oggetto è istanza di **una sola classe più specifica**, due sottoclassi non disgiunte possono avere istanze comuni solo se hanno una sottoclasse comune (ereditarietà multipla)



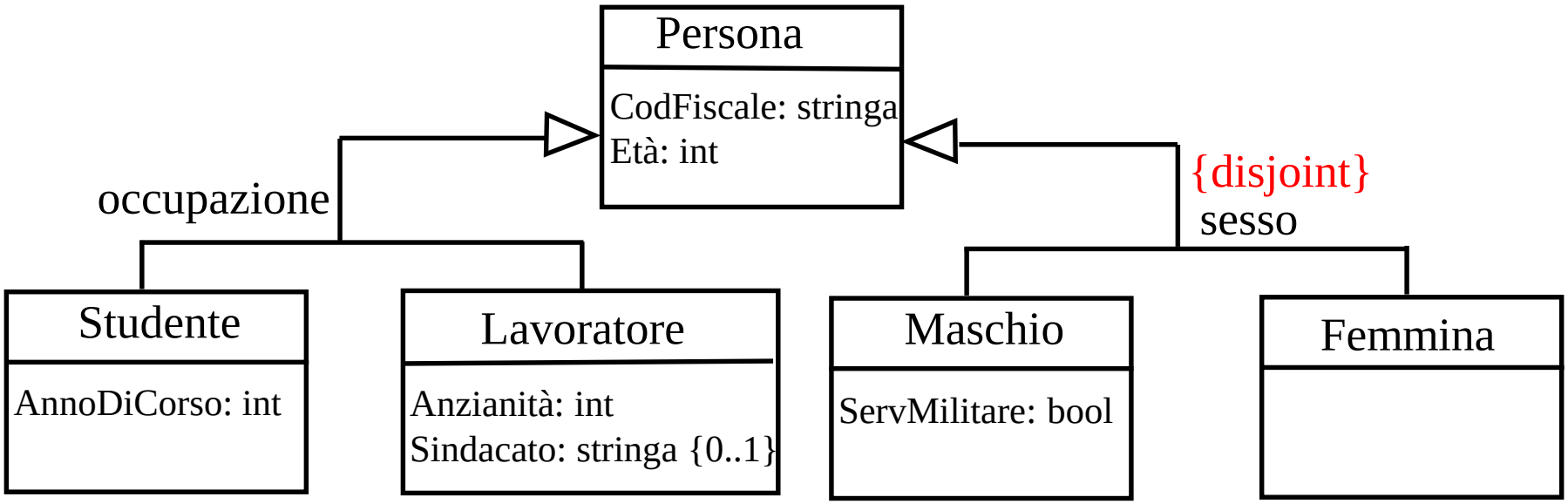
Differenza tra classi disgiunte e non disgiunte

Da quanto detto, la differenza tra due classi mutuamente disgiunte e due classi non mutuamente disgiunte sta solo nel fatto che due classi disgiunte non possono avere sottoclassi comuni, mentre è possibile definire una classe come sottoclasse di due classi non disgiunte



Il problema delle classi disgiunte (1)

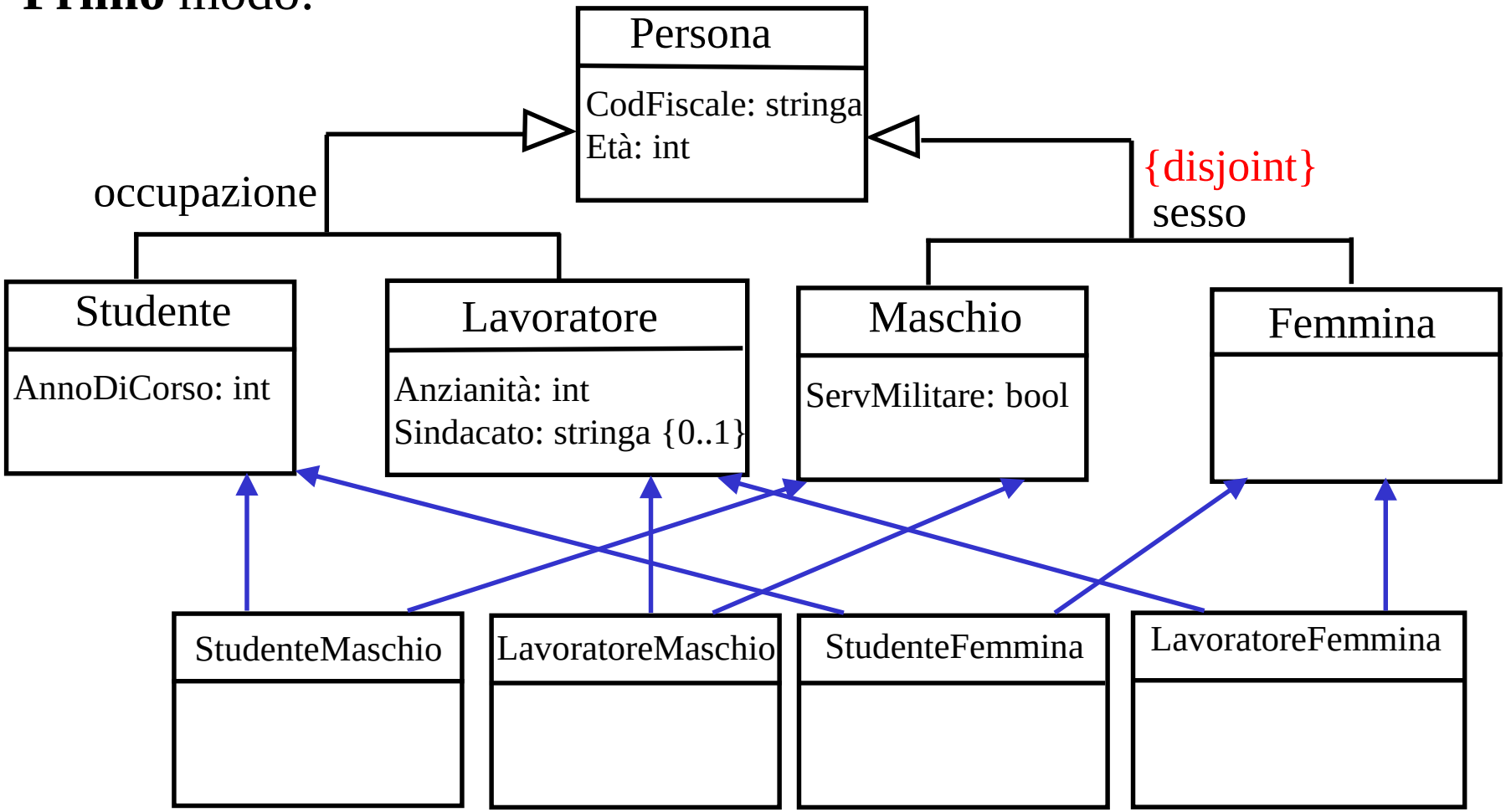
Consideriamo le seguenti generalizzazioni:



Questo diagramma descrive una situazione in cui non possono essere definite istanze di Studenti che sono anche esplicite istanze di Maschio (o di Femmina), e istanze di Lavoratore che sono anche istanze esplicite di Maschio (o di Femmina)

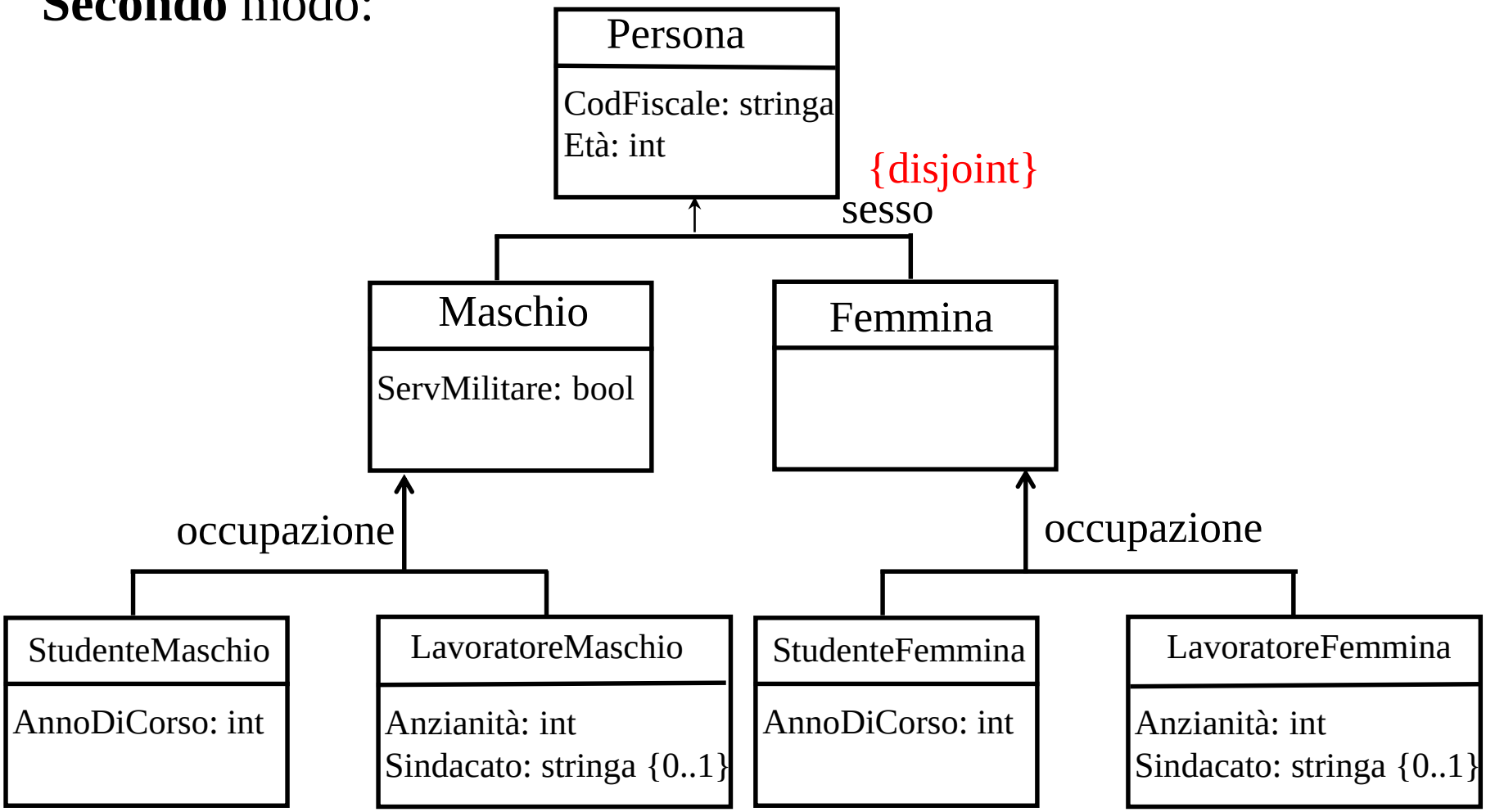
Il problema delle classi disgiunte (2)

Se vogliamo definirle si può ristrutturare lo schema in due modi.
Primo modo:

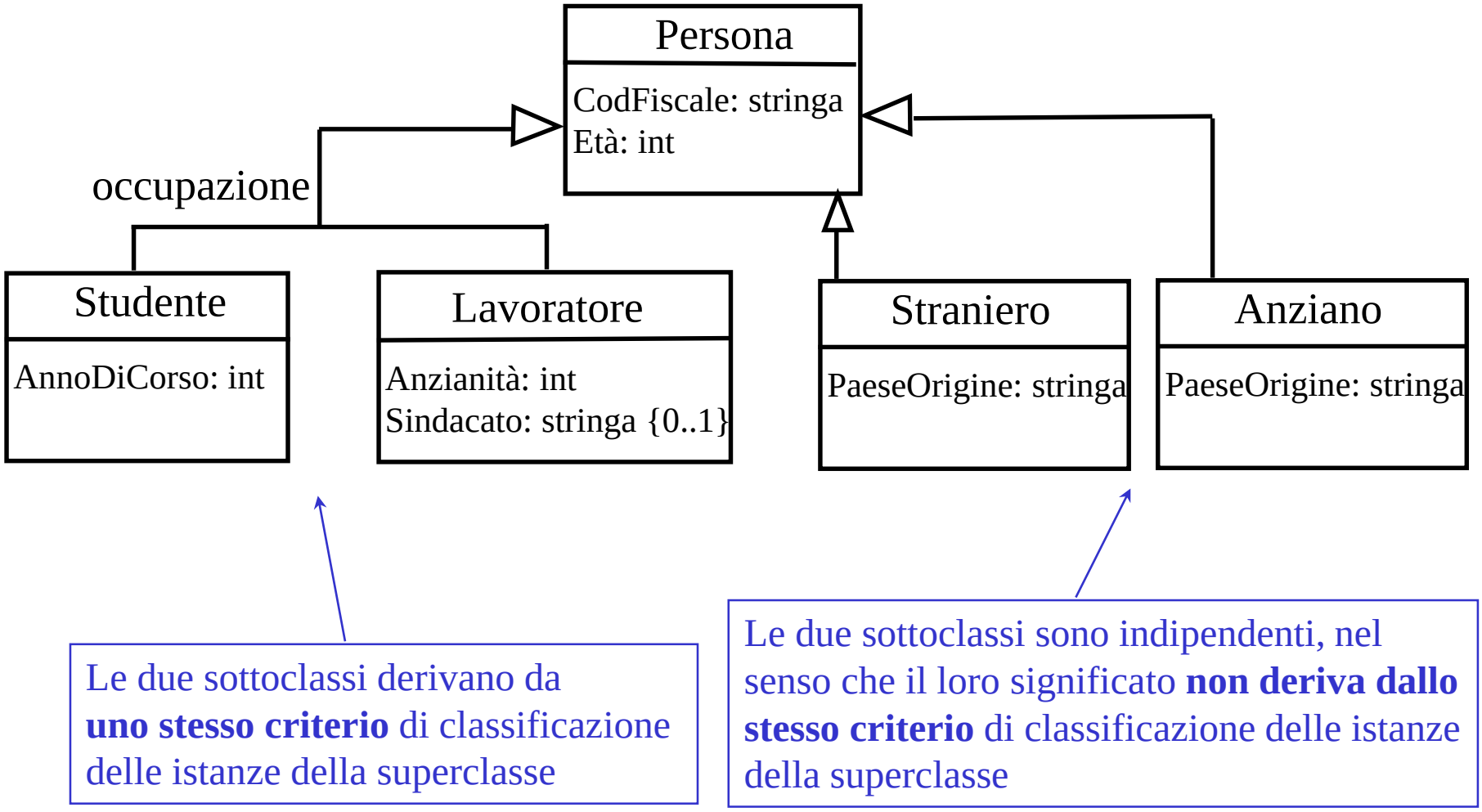


Il problema delle classi disgiunte (3)

Secondo modo:



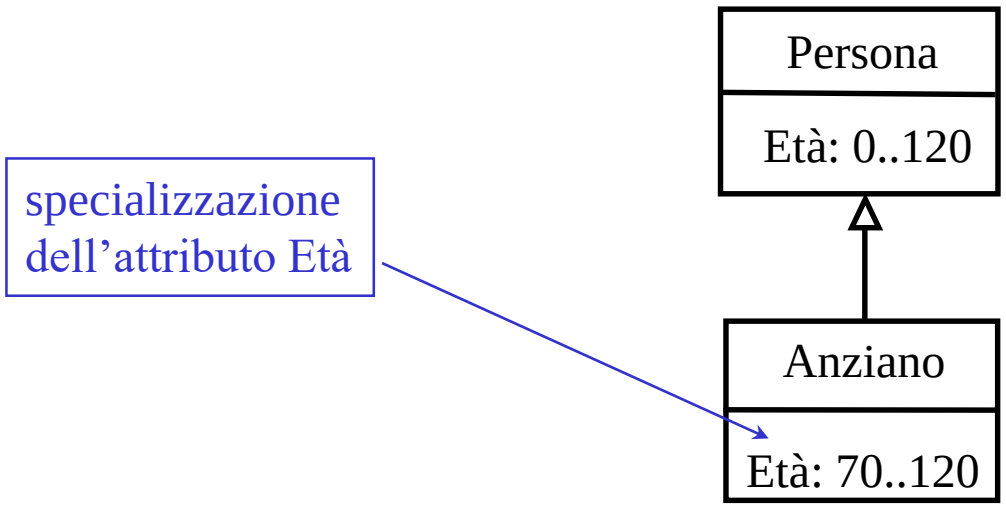
Differenza tra due isa e una generalizzazione



Specializzazione (1)

In una generalizzazione la sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche specializzare le proprietà ereditate dalla superclasse.

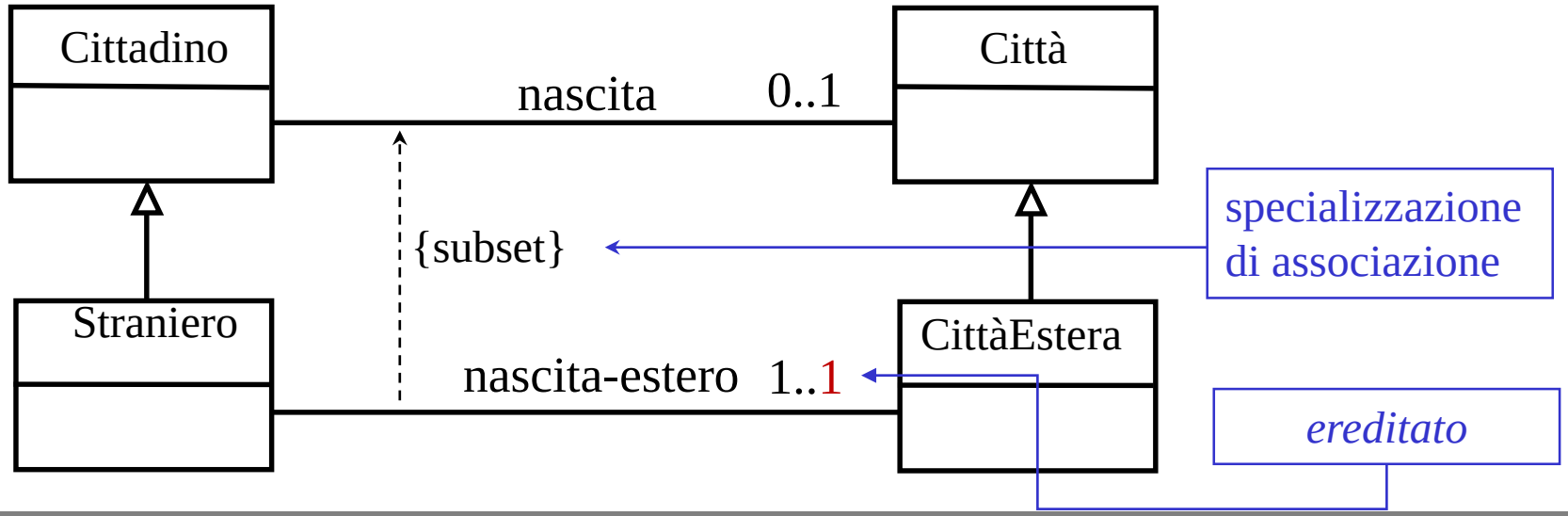
Specializzazione di un attributo: Se una classe C_1 ha un attributo A di tipo T_1, e se C_2 è una sottoclasse di C_1, specializzare A in C_2 significa definire A anche in C_2 ed assegnargli un tipo T_2 i cui valori sono un sottoinsieme dei valori di T_1.



Specializzazione (2)

Specializzazione di una associazione: Se una classe C_1 partecipa ad una associazione R con un'altra classe C_3, e se C_2 è una sottoclasse di C_1, specializzare R in C_2 significa:

- Definire una nuova associazione R_1 tra la classe C_2 e una classe C_4 che è sottoclasse di C_3 (al limite C_4 può essere la classe C_3 stessa)
- Stabilire una dipendenza di tipo {subset} da R_1 a R
- Definire eventualmente molteplicità più specifiche su R_1 rispetto alle corrispondenti molteplicità definite su R (si noti che la molteplicità massima su R_1 deve essere minore o uguale a quella su R)



Esercizio 13

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Dei voli aerei interessa il codice, la durata, l'aeroporto di partenza, l'aeroporto di arrivo e gli eventuali aeroporti delle tappe intermedie con l'ordine delle tappe.

Degli aeroporti interessa il nome, il numero delle piste, la città e la regione in cui si trovano.

I voli si dividono in giornalieri, settimanali, mensili.

Dei voli giornalieri interessa l'orario di partenza. Dei voli settimanali interessa il giorno della settimana e l'orario di partenza. Dei voli mensili interessa il giorno del mese, l'orario di partenza, e le regioni che sorvola nel tragitto.

Esercizio 13 (soluzione)

Esercizio 14

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

L'applicazione da progettare riguarda le informazioni sulle contravvenzioni elevate in un comune. Di ogni contravvenzione interessa il veicolo a cui è stata effettuata, il vigile che l'ha elevata, il numero di verbale e la data.
Di ogni vigile interessa il nome, il cognome ed il numero di matricola. Di ogni veicolo interessa la targa. Esistono solamente due categorie di veicoli, che sono fra loro disgiunte: automobili e motocicli. Delle automobili interessa la potenza in kilowatt, dei motocicli il numero di telaio.
Sappiamo che il comune vuole effettuare, come cliente della nostra applicazione, dei controlli sul lavoro del proprio personale. In particolare, si deve progettare il diagramma delle classi in modo che, dato un vigile, il comune possa controllare se un vigile ha elevato una contravvenzione per più di una volta ad uno stesso veicolo.

Esercizio 14 (soluzione)

Esercizio 15

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Le officine riparano i veicoli. Di ogni officina interessano il nome, l'indirizzo, il numero di dipendenti, i dipendenti (con l'informazione su quanti anni di servizio ciascun dipendente ha svolto presso l'officina), ed il direttore (unico). Si noti che un dipendente lavora presso una ed una sola officina, e che un direttore dirige una ed una sola officina, e non è anche un dipendente (e vice-versa). Dei dipendenti e dei direttori interessano: il codice fiscale, l'indirizzo ed il numero telefono. Dei direttori interessa anche l'età. Di ogni riparazione interessa: il codice, l'officina presso cui è eseguita, il veicolo oggetto della riparazione, l'ora e la data di accettazione, e, per le riparazioni portate a termine, l'ora e la data di riconsegna. Di ogni veicolo interessano: il modello, il tipo, la targa, l'anno di immatricolazione, ed il proprietario. Dei proprietari interessa codice fiscale, l'indirizzo, ed il numero telefono. Si noti che il proprietario di un veicolo riparato può essere anche un dipendente o il direttore di un'officina.

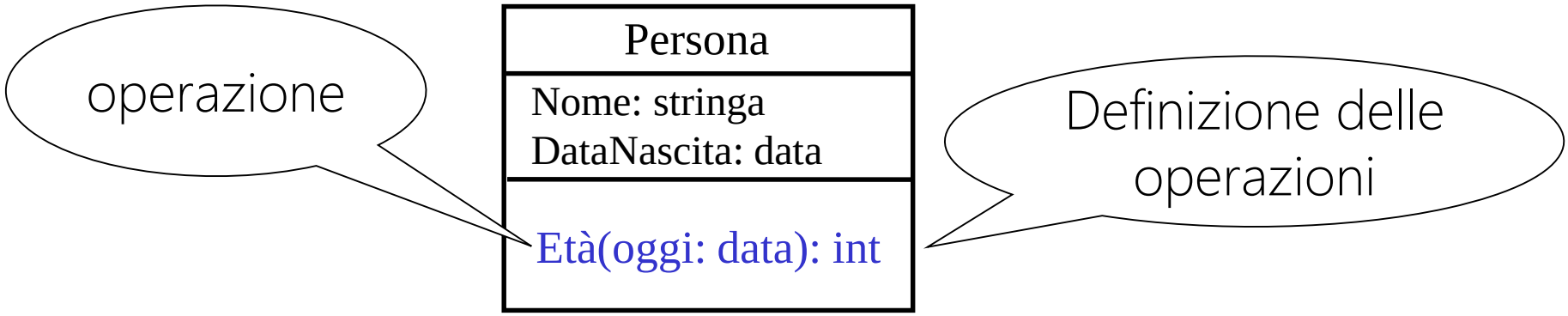
Esercizio 15 (soluzione)

Operazioni

Finora abbiamo fatto riferimento solamente a proprietà statiche (attributi e associazioni) di classi. In realtà, le classi (e quindi le loro istanze) sono caratterizzate anche da proprietà **dinamiche**, che in UML si definiscono mediante le **operazioni.**

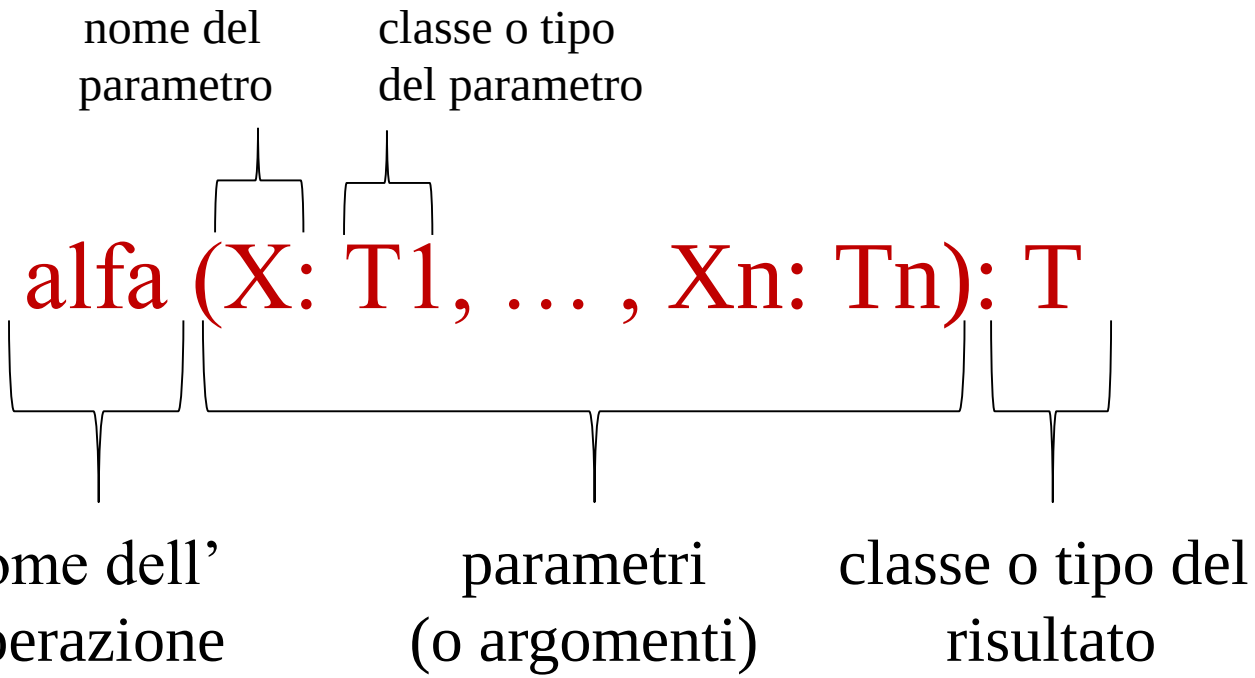
Una operazione associata ad una classe *C* indica che sugli oggetti della classe *C* si può eseguire una computazione, cioè una elaborazione (detta anche **metodo**),

- o per calcolare le proprietà
- o per effettuare cambiamenti di stato (cioè per modificare le proprietà)



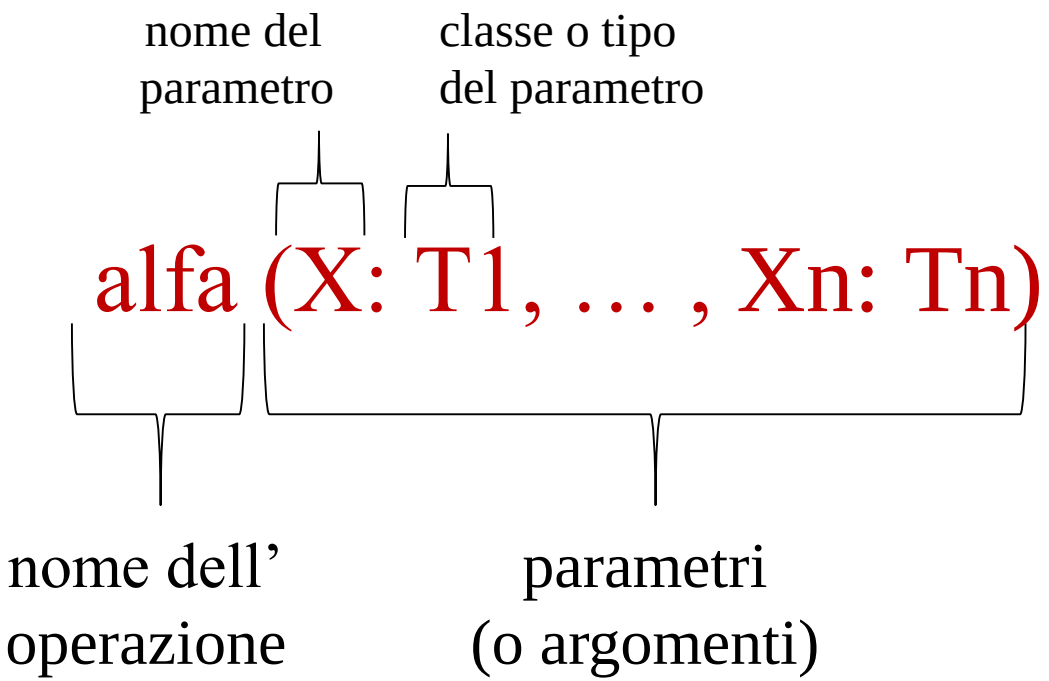
Definizione di una operazione (1)

In una classe, una operazione si definisce specificando la **segnatura** (nome, parametri e il tipo del eventuale risultato) e **non il metodo** (cioè non la specifica di cosa fa l'operazione)



Definizione di una operazione (2)

Non è necessario che una operazione **restituisca** un valore o un oggetto. Una operazione può anche solo effettuare azioni senza calcolare un risultato. In questo caso l'operazione si definisce così:



Osservazioni sulle operazioni (1)

- Una operazione di una classe *C* è pensata per essere invocata facendo riferimento ad una istanza della classe *C*, chiamata **oggetto di invocazione**.

- Esempio di invocazione:

  p.Età(oggi)

  (dove p è un oggetto della classe Persona).

- In altre parole, nell'attivazione di ogni operazione, oltre ai parametri c'è **sempre implicitamente in gioco un oggetto** (l'oggetto di invocazione) della classe in cui l'operazione è definite.

Osservazioni sulle operazioni (2)

Attenzione: le **operazioni** che si definiscono sul modello di analisi sono le operazioni che **caratterizzano concettualmente** la classe.

Altre operazioni, più orientate alla **realizzazione** del software (come ad esempio le operazioni che consentono di gestire gli attributi, ossia conoscerne o cambiarne il valore), **non devono** essere definite in questa fase.

Esercizio 16

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

Si vogliono modellare gli studenti (con nome, cognome, numero di matricola, età), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame, con il professore che ha verbalizzato l'esame, ed il voto conseguito. Di ogni corso di laurea interessa il codice e il nome. Di ogni corso interessa il nome e la disciplina a cui appartiene (ad esempio: matematica, fisica, informatica, ecc.). Di ogni professore interessa codice ed età.

Al momento dell'iscrizione, lo studente specifica il corso di laurea a cui si iscrive.

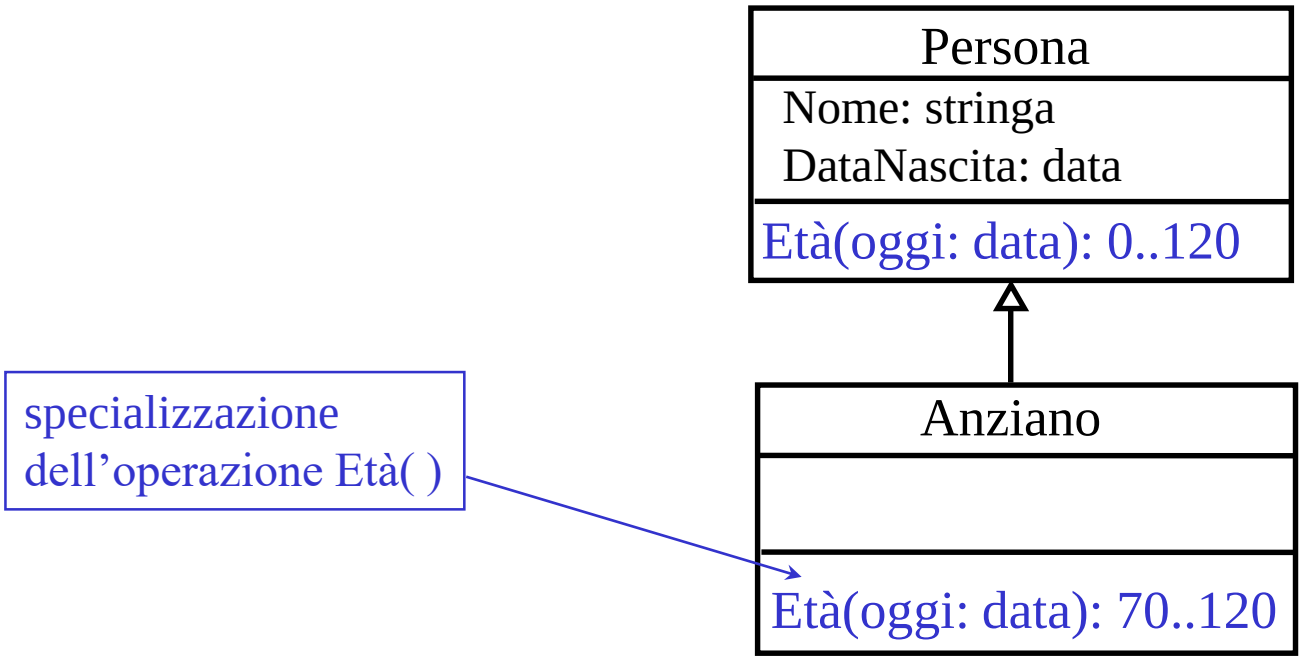
Dopo l'effettuazione di un esame, il professore comunica l'avvenuta verbalizzazione dell'esame con i dati relativi (studente, corso, voto).

La segreteria vuole periodicamente calcolare la media dei voti di uno studente, e il numero di studenti di un corso di laurea.

Esercizio 16 (soluzione)

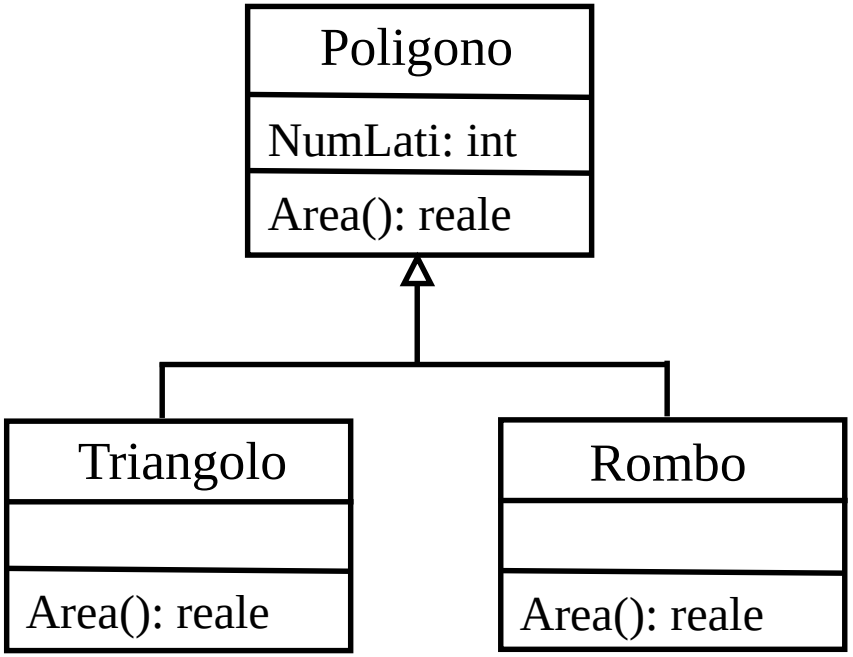
Specializzazione di operazioni (1)

Oltre agli attributi e alle associazioni, anche le operazioni si possono specializzare nelle sottoclassi. Una operazione si specializza specializzando i parametri e/o il tipo di ritorno.



Specializzazione di operazioni (2)

In genere il **metodo** associato ad una operazione specializzata in una sottoclasse è **diverso** dal metodo associato alla stessa operazione nella superclasse



La segnatura dell'operazione Area() non è cambiata rispetto alla classe padre. Lo schema specifica che ogni poligono ha l'operazione Area(), ma il metodo associato ad esso dipende dal tipo di poligono.

Osservazione sui tipi (1)

Finora abbiamo semplicemente assunto che si possano usare nel diagramma delle classi i tipi di dato **semplici** (come ad esempio **int**, **stringa**, ecc.)

In realtà, si possono usare anche tipi di dato **più complessi** (non altre classi).

Ad esempio si possono usare tipi complessi come:

- **Record,**
- **Insieme,**
- **Lista,**
- **Array,**
- **Ecc.**

Osservazione sui tipi (1)

Ad esempio, si può pensare di utilizzare il tipo **indirizzo** come record con campi

- *“strada” (di tipo stringa) e*
- *“numero civico” (di tipo intero)*

O ancora, possiamo usare il tipo **data** come record con campi

- *giorno (di tipo 1..31),*
- *mese (di tipo 1..12) e*
- *anno (di tipo intero).*

Semantica dei diagrammi delle classi: riassunto

Concetto	Significato	Note
Oggetto	Elemento	Ogni oggetto ha vita propria ed ha un unico identificatore
Classe	Insieme di oggetti	Insieme con operazioni
Tipo	Insieme di valori	Un valore non ha vita propria
Attributo	Funzione (o relazione, se multivalore)	Da classi (e associazioni) a tipi
Associazione	Relazione	Sottoinsieme del prodotto cartesiano
Relazione is-a	Sottoinsieme	Implica ereditarietà
Generalizzazione disgiunta e completa	Partizione	Le sottoclassi formano una partizione della superclasse
Operazione	Computazione	Le operazioni vengono definite nelle classi

Aspetti metodologici nella costruzione del diagramma delle classi

Un metodo comunemente usato per costruire il diagramma delle classi prevede i seguenti passi

1. *Individua le classi e gli oggetti di interesse*
2. *Individua gli attributi delle classi*
3. *Individua le associazioni tra classi*
4. *Individua gli attributi delle associazioni*
5. *Determina le molteplicità di associazioni e attributi*
6. *Individua le generalizzazioni, partendo o dalla classe più generale e scendendo nella gerarchia, oppure dalle classi più specifiche e risalendo nella gerarchia*
7. *Determina le specializzazioni*
8. *Individua le operazioni ed associale alle classi*
9. *Controllo di qualità*

Correggi, modifica, estendi

Controllo di qualità sul diagramma delle classi

- *È stata fatta una scelta oculata su come modellare i vari concetti?*
  - *Se con attributi o con classi*
  - *Se con classi o con associazioni*
- *Sono stati colti tutti gli aspetti importanti delle specifiche?*
- *Si è verificato che le generalizzazioni non formano cicli?*
- *Le specializzazioni sono corrette?*
- *Si possono applicare ulteriori generalizzazioni?*
- *Ci sono classi che sono sottoinsiemi di classi disgiunte?*

Scelta tra attributi e classi (1)

La scelta deve avvenire tenendo presente le seguenti differenze tra classi e tipi

	Classe	Tipo
Istanze	oggetti	valore
Istanze identificate da	identificatore di oggetto	valore
Uguaglianza	basata su identificatore	basata su valore
Realizzazione	da progettare	tipicamente predefinita, oppure basata su strutture di dati predefinite

Scelta tra attributi e classi (2)

Un concetto verrà modellato come

una classe

- se le sue istanze hanno **vita propria**
- se le sue istanze possono essere identificate **indipendentemente** da altri oggetti
- se ha o si prevede che avrà delle **proprietà indipendenti** dagli altri concetti
- se su di esso si "**predica**" nello schema concettuale

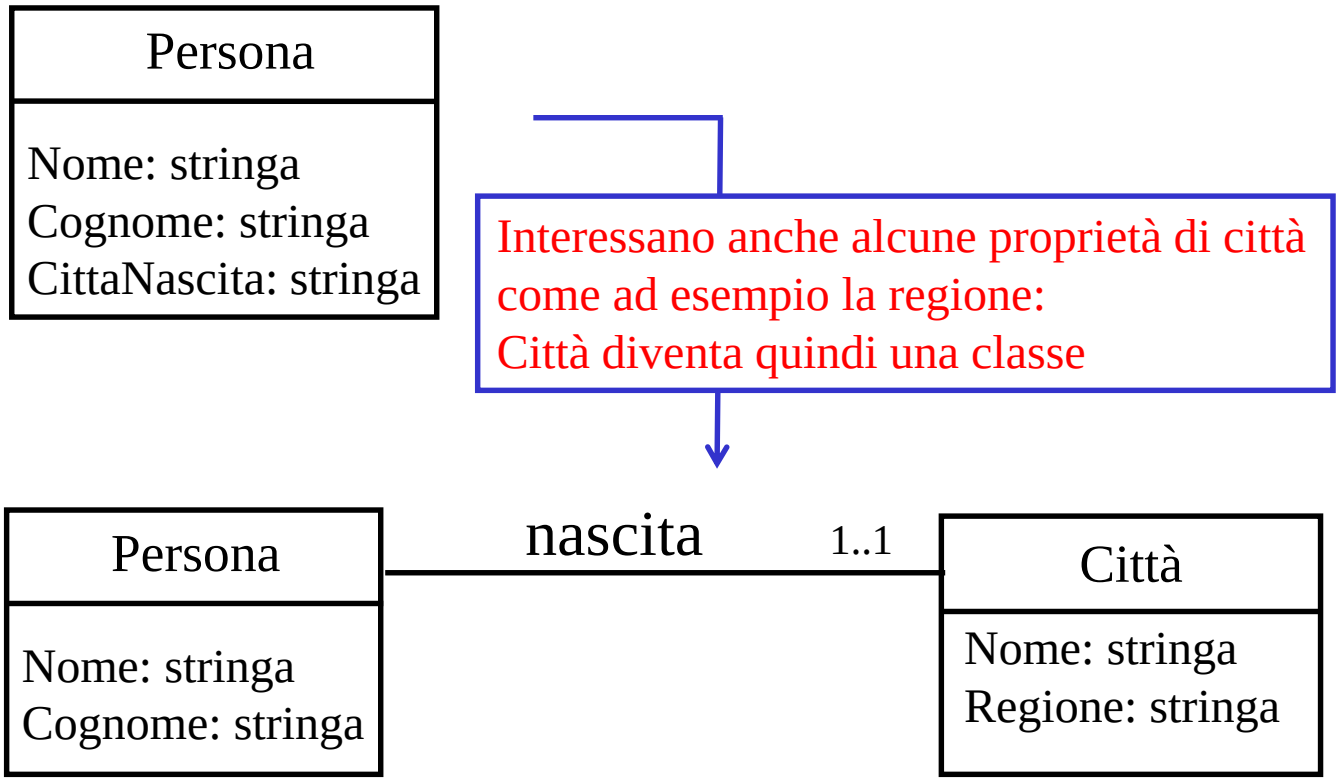
un attributo

- se le sue istanze non hanno vita propria
- se ha senso solo per rappresentare proprietà di altri concetti
- se non si "predica" su di esso nello schema concettuale

Scelta tra attributi e classi (2)

Le scelte possono cambiare durante l'analisi.

Esempio:



Scelta tra classi e associazione

Un concetto verrà modellato come

una classe

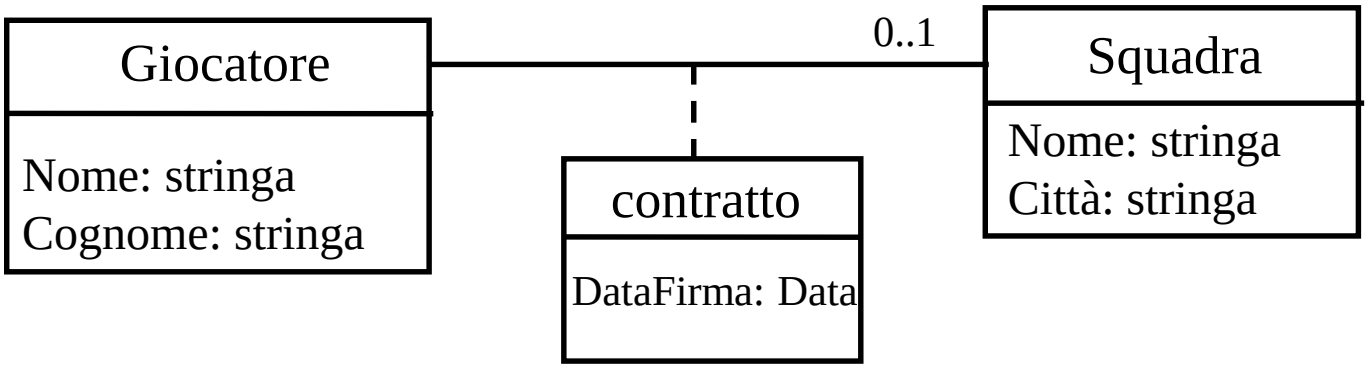
- se le sue istanze hanno **vita propria**
- se le sue istanze possono essere **identificate** indipendentemente da altri oggetti
- se ha o si prevede che avrà delle **associazioni con altri concetti**

una associazione

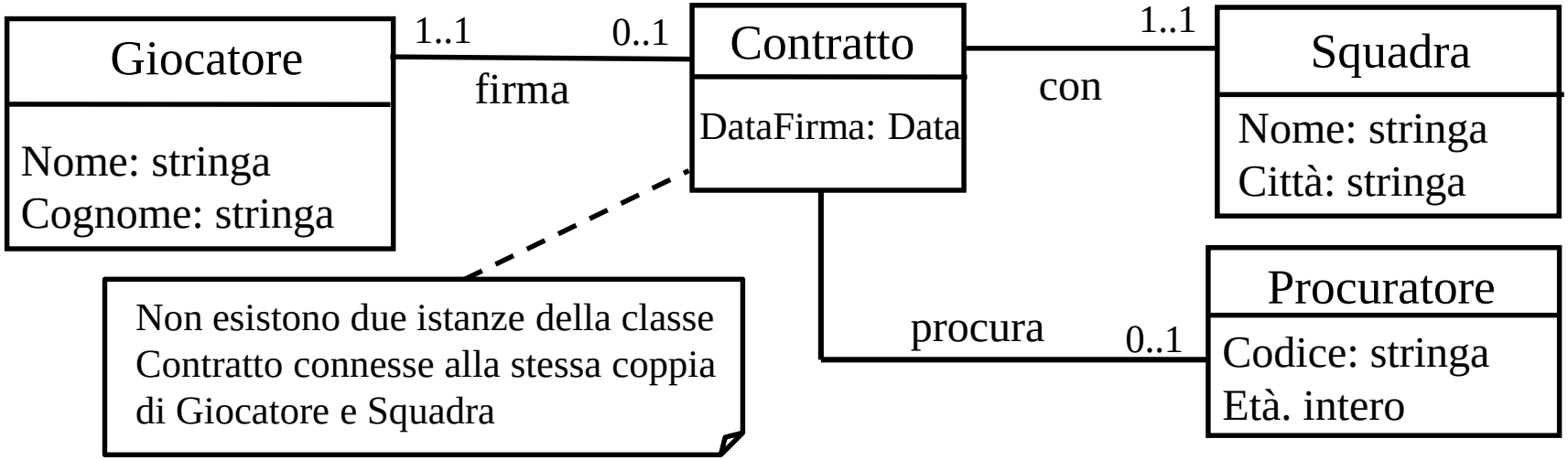
- se le sue istanze rappresentano n-ple di altre istanze
- se non ha senso pensare alla partecipazione delle sue istanze ad altre associazioni

Scelta tra classi e associazioni

Le scelte possono cambiare durante l'analisi. Esempio:

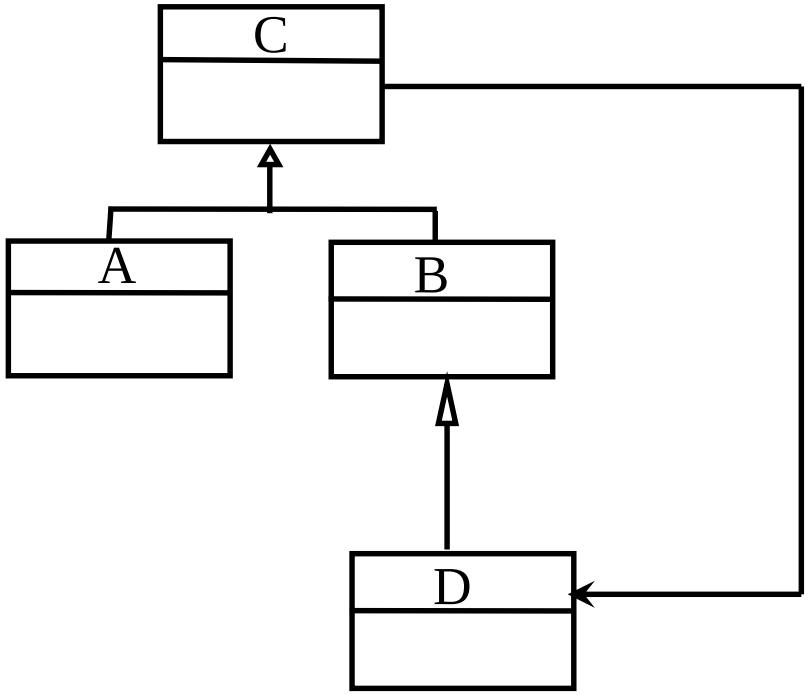


Interessa anche il procuratore (con codice ed età), se c'è: Contratto diventa una classe



Verifiche sulle generalizzazioni

Il grafo delle generalizzazioni **non** può contenere cicli!



Ciclo nel grafo delle generalizzazioni: le classi C, B e D hanno le stesse istanze!

Verifiche sulle specializzazioni

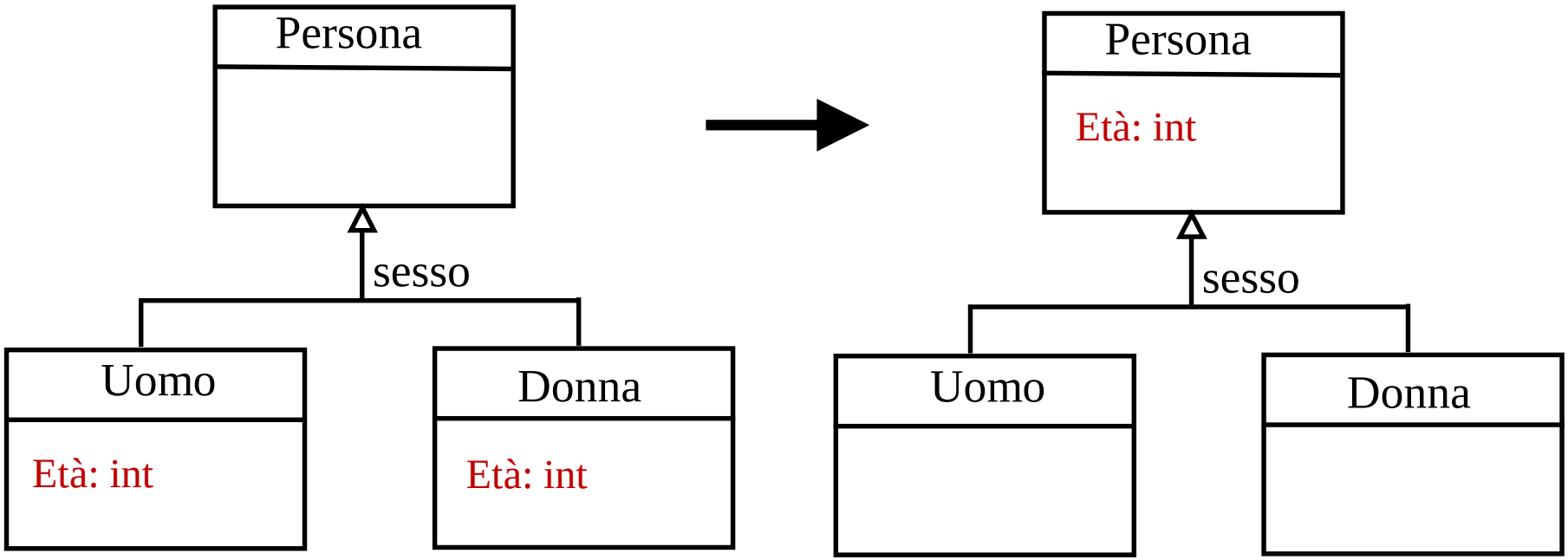
- **Specializzazione di un attributo**: se una classe C_1 ha un attributo A di tipo T_1, se C_2 è una sottoclasse di C_1, e se A è specializzato in C_2, allora il tipo assegnato ad A in C_2 deve essere un tipo T_2 i cui valori sono un sottoinsieme dei valori di T_1.

- **Specializzazione di una associazione**: se una classe C_1 partecipa ad una associazione R con un'altra classe C_3, se C_2 è una sottoclasse di C_1, ed R è specializzata in C_2 in una associazione R_1 con C_4 allora:
  - tra R_1 ed R deve esserci una dipendenza di tipo {subset}
  - per R_1 deve essere definita una molteplicità massima uguale o più ristretta che per R
  - C_4 è una sottoclasse di C_3 (al limite C_3 e C_4 sono uguali)

Si possono applicare ulteriori generalizzazioni?

È bene verificare che gli attributi siano stati associati alle classi giuste in una generalizzazione

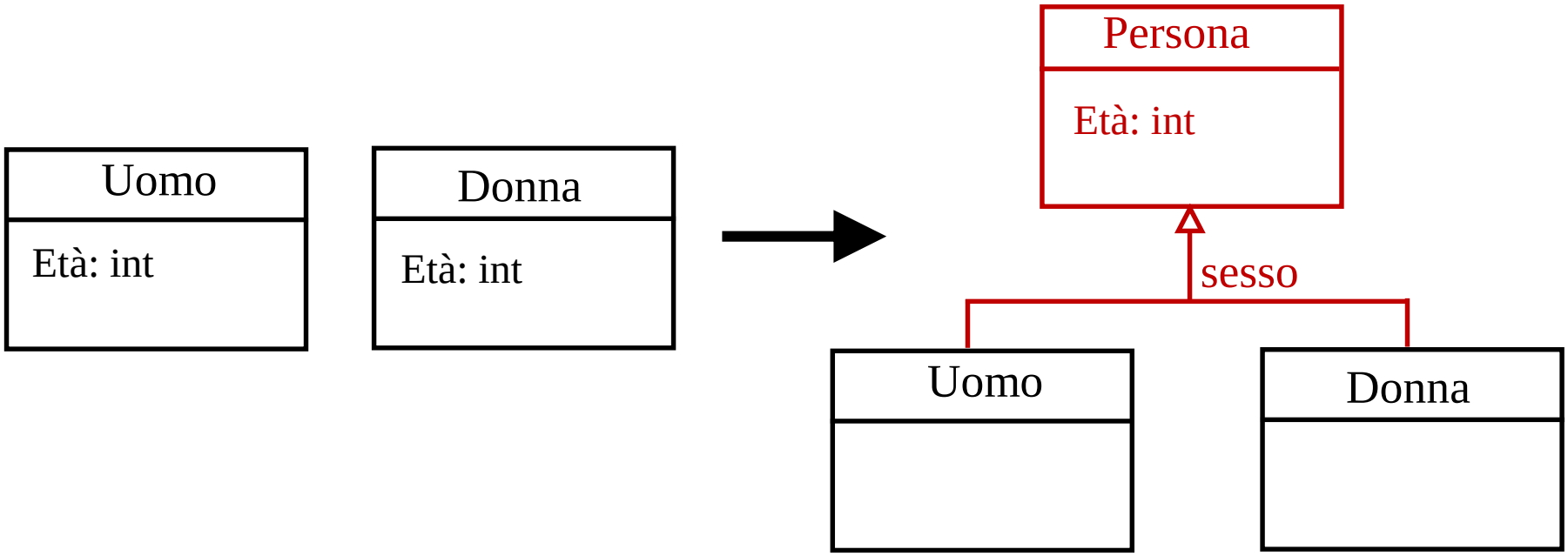
Esempio:



Si possono applicare ulteriori generalizzazioni?

È bene verificare se non sia il caso di introdurre nuove classi generalizzazioni

Esempio:



Esercizio 17

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

L'applicazione da progettare riguarda le informazioni su un insieme di pozzi petroliferi. Di ogni pozzo interessa il codice, e quali compagnie petrolifere estraggono greggio dal pozzo, con l'informazione sul numero di barili di greggio estratti all'anno da ognuna di tali compagnie. Di ogni compagnia interessa il codice fiscale ed il numero di dipendenti. Si noti che ogni pozzo consente l'estrazione di greggio ad un numero qualunque (anche zero) di compagnie, ed ogni compagnia può estrarre greggio da un numero qualunque (anche zero) di pozzi. Dato un pozzo, è di interesse conoscere il tasso di inquinamento del pozzo stesso, secondo un calcolo che dipende dalla categoria alla quale appartiene il pozzo stesso. Esistono infatti due e solo due categorie di pozzi, i pozzi terrestri ed i pozzi marini.

(1) Dei pozzi terrestri interessa la superficie di estensione del pozzo.
Il tasso di inquinamento per un pozzo terrestre si calcola sommando la quantità di greggio estratto annualmente da quel pozzo dalle varie compagnie.

(2) Dei pozzi marini interessa l'anno di installazione, e l'area geografica in cui si trova. Ogni area geografica può ospitare al più un pozzo, è identificata da un codice, ed è caratterizzata da un grado di importanza ambientale (un numero intero). Il tasso di inquinamento per un pozzo marino si calcola sommando la quantità di greggio estratto annualmente da quel pozzo dalle varie compagnie, e moltiplicando il risultato per il grado di importanza ambientale dell'area geografica in cui il pozzo si trova (se l'area non è nota, si moltiplica per il valore 20).

Esercizio 17 (soluzione)

Esercizio 18

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

In un campionato di calcio ci sono squadre composte da giocatori e coinvolte in un certo numero di partite.

Di una squadra interessano il nome, i colori sociali, la città e la regione di appartenenza, le partite che ha giocato (sia in casa che fuori casa).

Di ogni partita interessano anche la data, il risultato e l'arbitro, di cui a sua volta si vuole conoscere nome, cognome, luogo e data di nascita, età e codice fiscale. Alla fine della partita l'arbitro ufficializza il risultato.

Ogni squadra è composta da almeno 11 giocatori. Di ogni giocatore interessano il nome, il cognome, il luogo e la data di nascita, l'età, il codice fiscale, il ruolo in cui gioca e la squadra in cui gioca attualmente. Inoltre, interessa conoscere anche in quali squadre ha giocato in passato, ed in che stagione (si noti che un giocatore può avere giocato in una stessa squadra in più di una stagione).

Di ogni squadra vogliamo anche conoscere l'allenatore in prima e l'allenatore in seconda. Quest'ultimo è sempre un giocatore "anziano", la cui età supera i 30 anni. Di entrambi gli allenatori interessano nome, cognome, luogo e data di nascita e codice fiscale.

Infine, sappiamo che gli utenti del sistema vogliono conoscere quanti giocatori ha una certa squadra.

Esercizio 18 (soluzione)