Progettazione del Software

G. De Giacomo & M. Scannapieco
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"

Analisi: UML

Analisi: UML

- 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

  OBIETTIVO:
  - costruire un modello dell'applicazione che sia completo, preciso, leggibile, e traducibile in un programma
  - 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 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
  - Il diagramma degli stati e delle transizioni
    - Descrive, per le classi significative, il tipico ciclo di vita delle sue istanze
  - 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 (metodologie utilizzate in passato)
  - diagrammi funzionali
  - diagrammi di flusso di controllo
  - diagrammi di flusso di dati
- Orientati agli oggetti (metodologie utilizzate attualmente)
  - Booch
  - OOSE (Jacobson)
  - OMT (Rambaugh)
  - Coad-Yourdon
  - Basato sul linguaggio UML

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, ...)
  - 1999/2001: Versione 1.3/1.4, descritta nei testi
  - 2003: Versione 1.5 + draft versione 2.0
- Riferimento:
  - *G. Booch, J. Rumbaugh, I. Jacobson, "The unified modeling language user guide", Addison Wesley, 1999.*
  - *http://www.omg.org* → *UML*
  - *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 (class and object diagram)
  - Diagrammi comportamentali:
    - Diagramma degli use case (use case diagram),
    - Diagramma degli stati e delle transizioni (state/transition diagram)
- Useremo UML con alcune limitazioni e regole precise

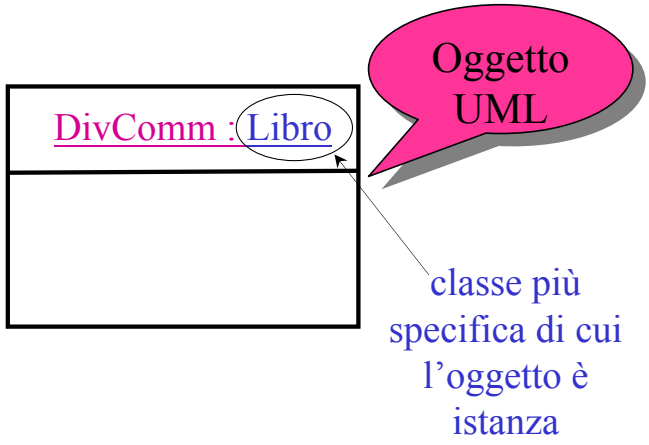
Diagramma delle classi e degli oggetti per l'analisi

- Nella fase di analisi ci si concentra sulle classi più che sugli oggetti
- Gli oggetti servono essenzialmente per per descrivere elementi singoli particolarmente significativi
- 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

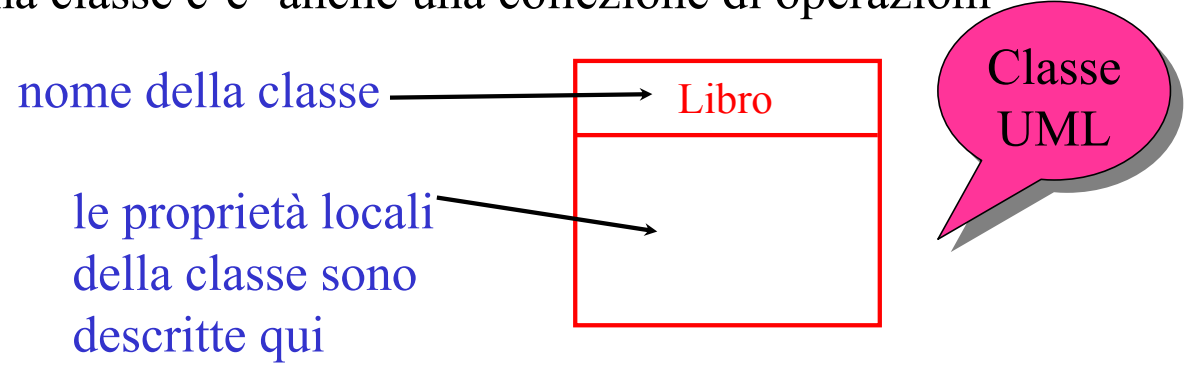
- Un oggetto in UML modella un elemento del dominio di analisi che
  - ha vita propria
  - è identificato univocamente mediante l'identificatore di oggetto
  - è istanza di una classe (la cosiddetta classe più specifica – vedremo che, in determinate circostanze, un oggetto è istanza di più classi, ma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



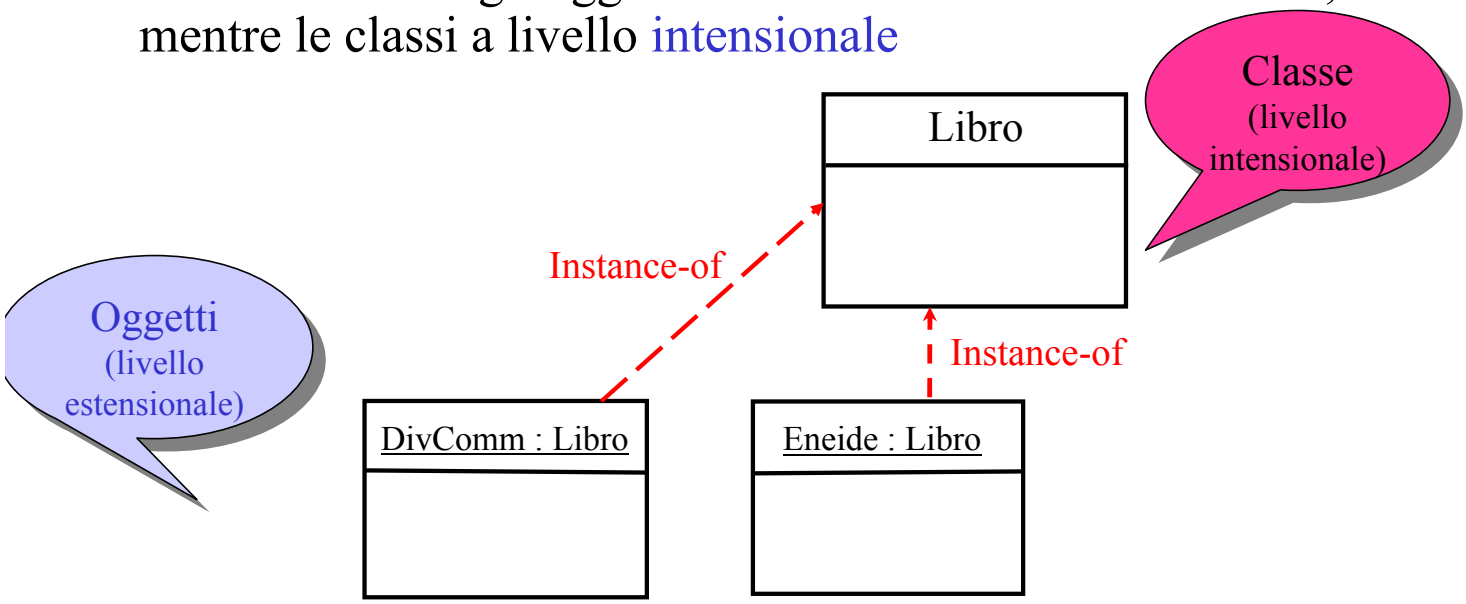
Classi in UML

- Una classe modella un insieme di oggetti omogenei (le istanze della classe) ai quali sono associate proprietà statiche e dinamiche (operazioni). Ogni classe e' descritta da:
  - un nome
  - un insieme di proprietà (astrazioni delle proprietà comuni degli oggetti che sono istanze delle classi)
- In realtà la visione di una classe come un insieme è un pò semplicistica; vedremo infatti che associata agli oggetti di una classe c'e' anche una collezione di operazioni



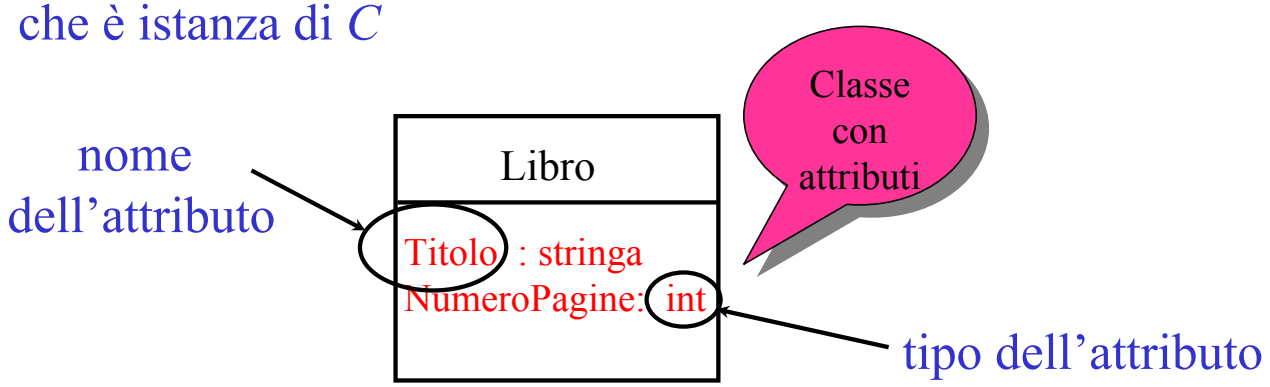
Rapporto tra classi e istanze

- 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, perchè 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



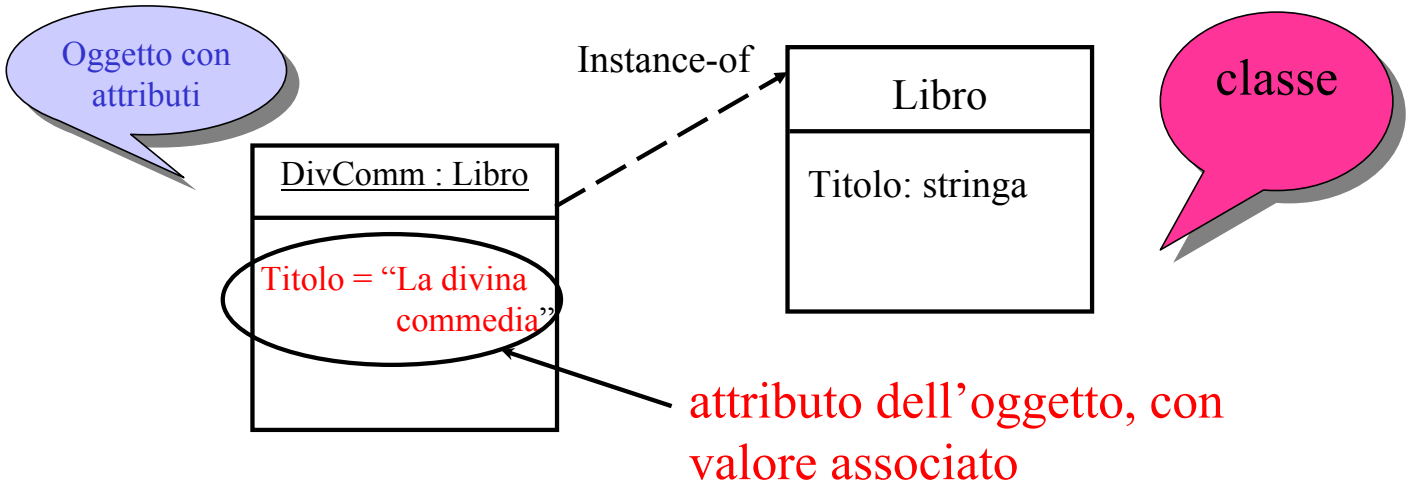
Proprietà di classi: attributi in UML

- Un attributo modella una proprietà locale della classe ed è caratterizzato da un nome e dal tipo dei valori associati
- 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*



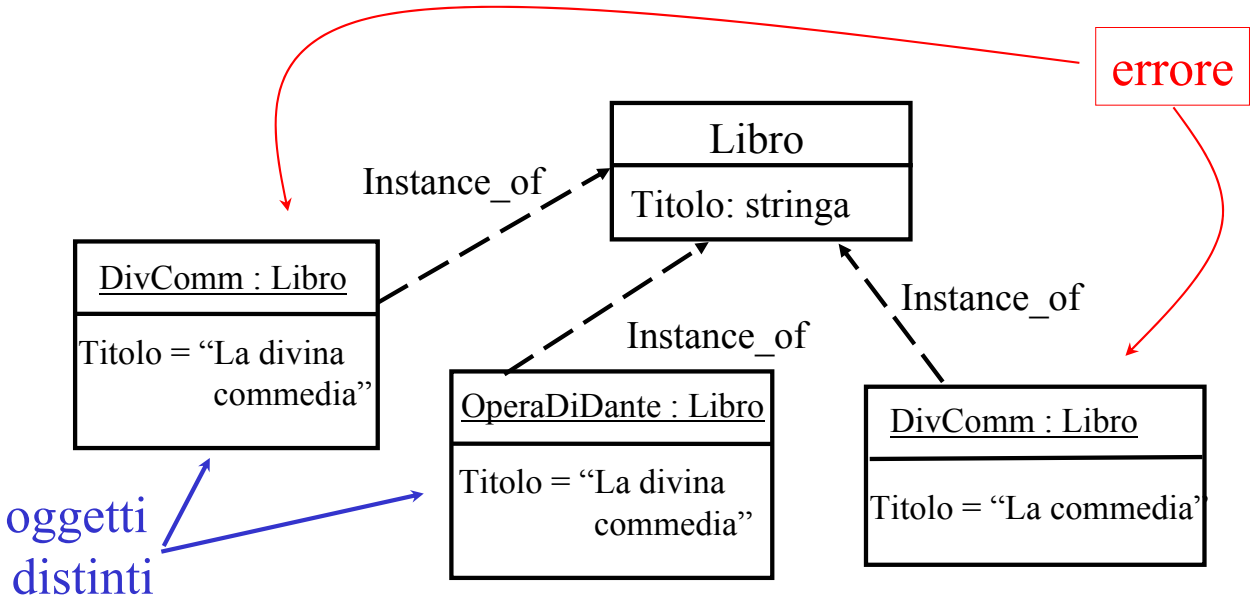
Attributi di oggetti

- 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 che non è definito nella classe di cui X è istanza

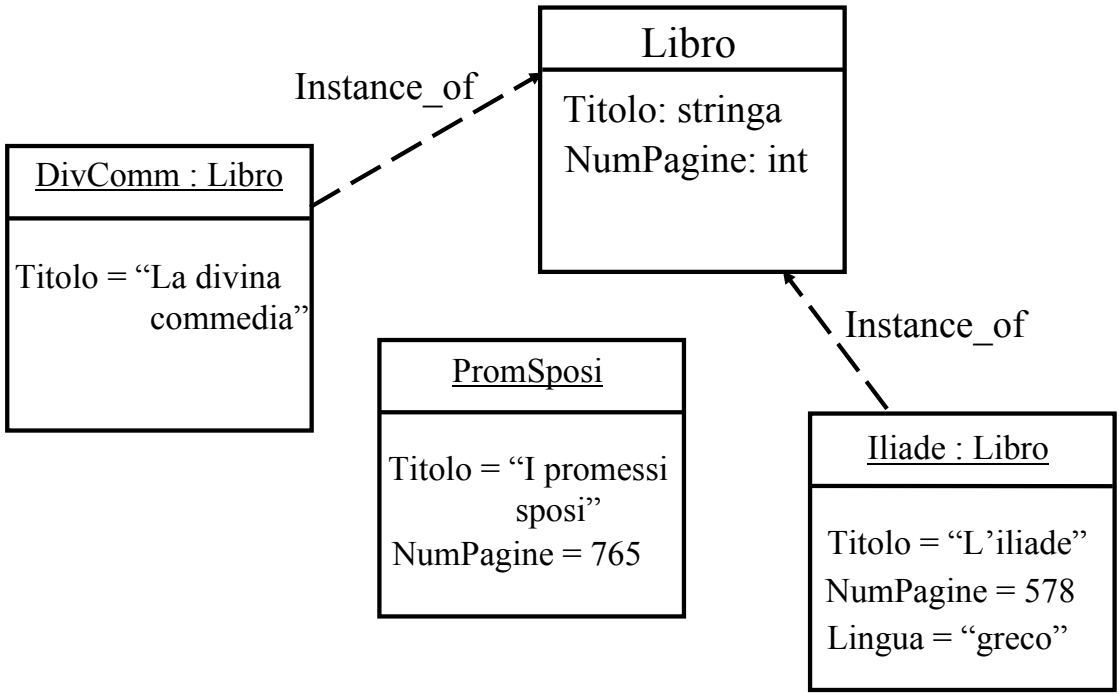


Importanza dell'identificatore di oggetto

- Due oggetti con identificatori distinti 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 attibuti



Esercizio 1



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

Errori dell'esercizio 1

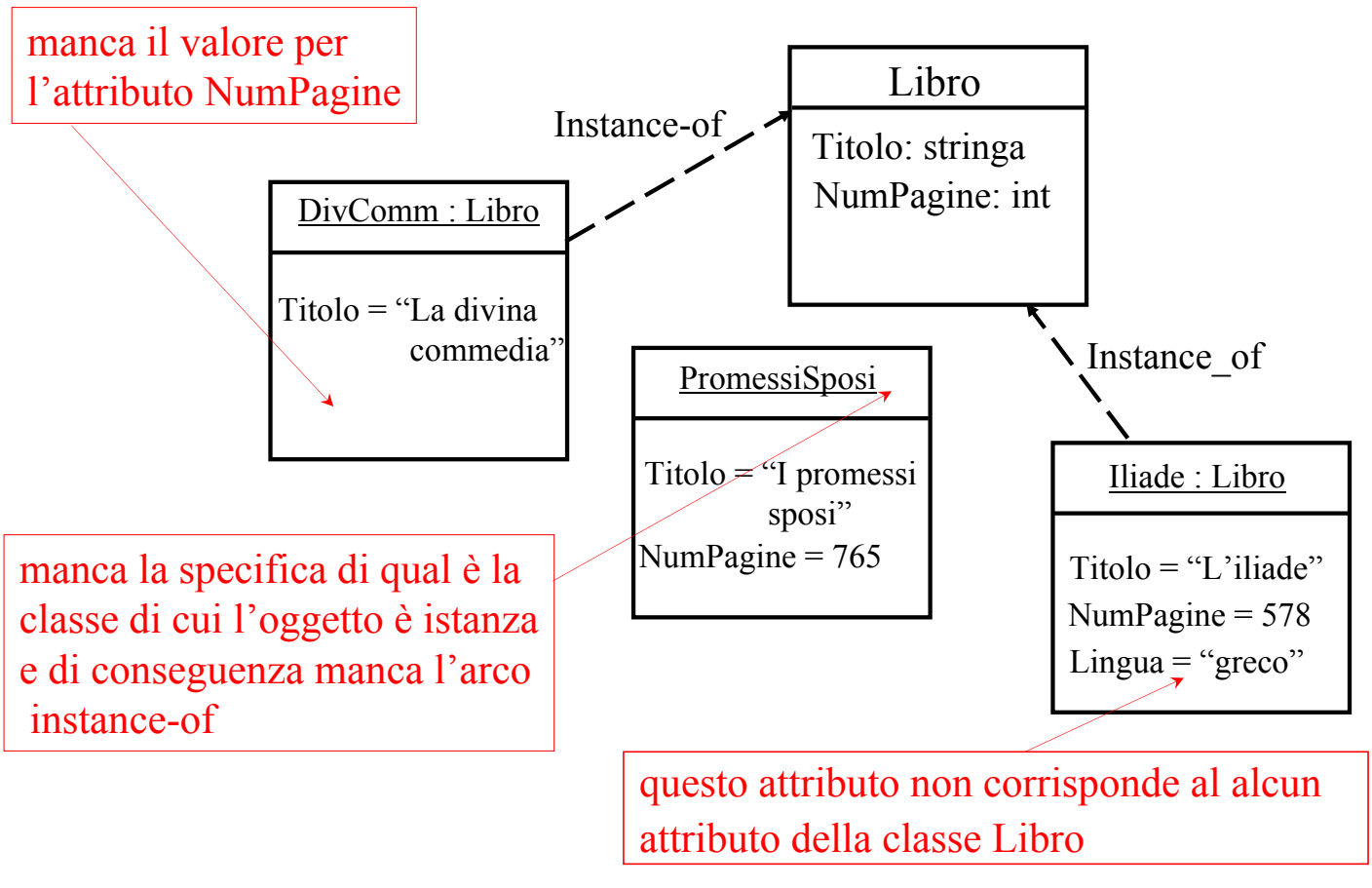
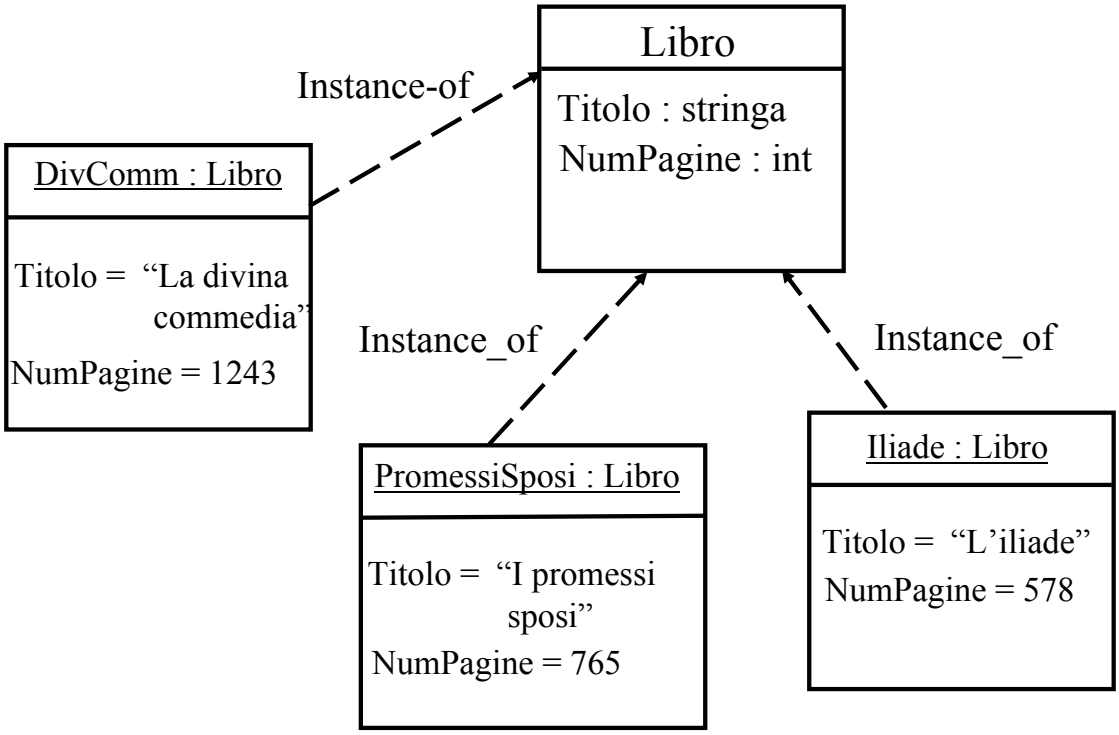


Diagramma corretto per l'esercizio 1



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

- 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 $<x,y>$ tali che $x \in S_1$ e $y \in S_2$

- Ovviamente, la nozione si estende banalmente a relazioni tra n insiemi: $R \subseteq S_1 \times S_2 \times \cdots \times S_n$

Intermezzo: relazione matematica

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

$S_1 \times S_2 = \{ <1,a>, <1,b>, <2,a>, <2,b>, <3,a>, <3,b> \}$

- Esempio di relazione tra S_1 e S_2 :

$R \subseteq S_1 \times S_2 = \{ <1,a>, <1,b>, <2,a> \}$

- 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

- Consideriamo gli insiemi
  studente = {Paolo, Anna, Lucia}
  esame = {Analisi, Geometria}

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

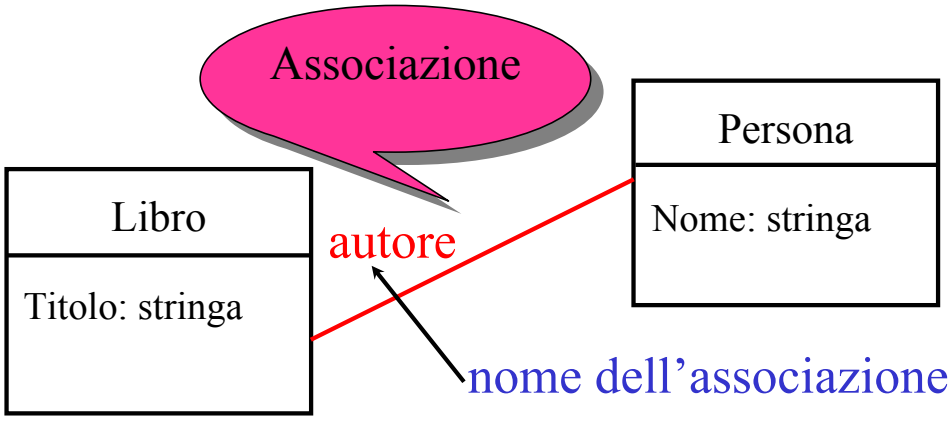
  sostenuto = {<Paolo,Analisi>, <Anna,Analisi>, <Anna,Geometria>}

- 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

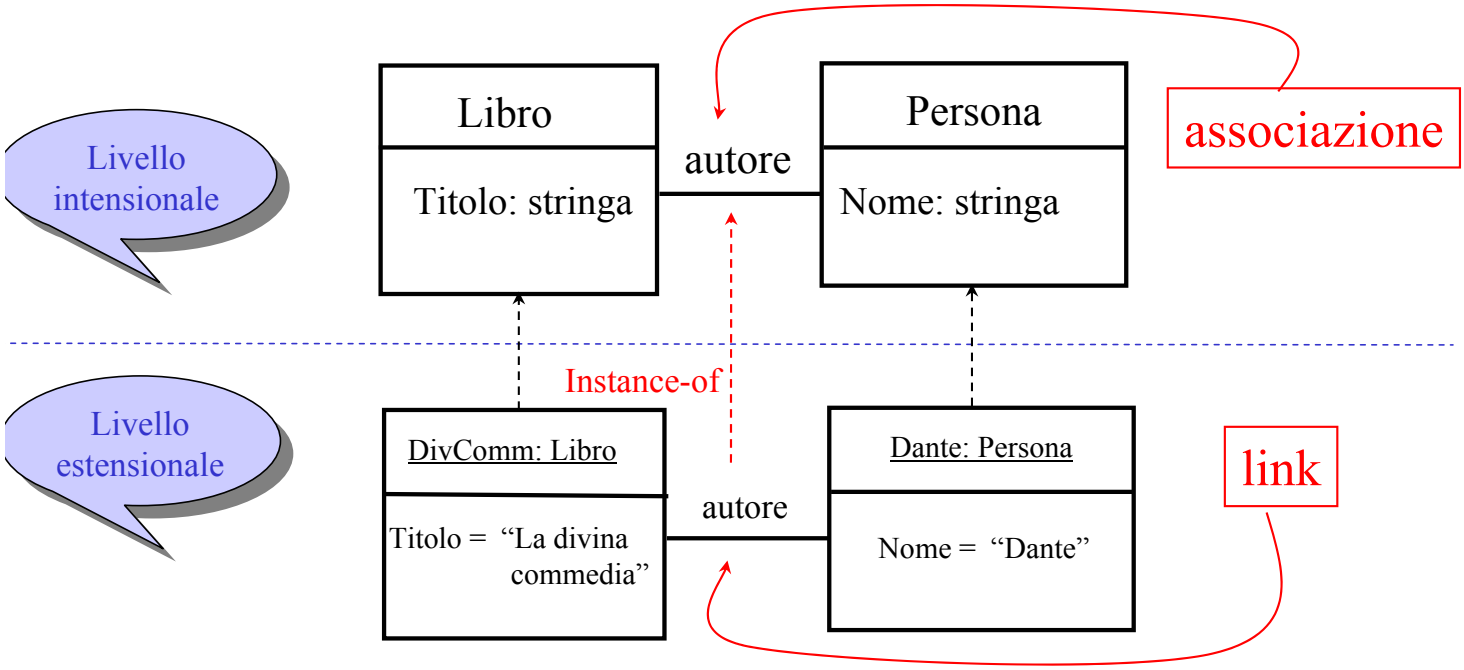
- Per il momento, ci limitiamo a discutere associazioni tra due classi (ma le associazioni possono coinvolgere N classi)
- Una associazione (o relazione) tra una classe C_1 ed una classe C_2 modella una relazione matematica tra l'insieme delle istanze di C_1 e l'insieme delle istanze di C_2
- Gli attributi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



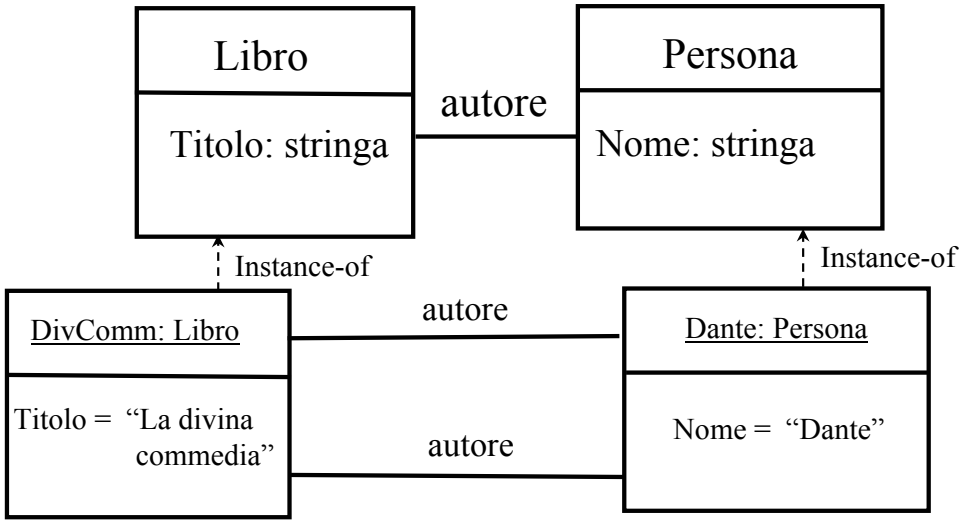
Istanze di associazioni: link

- 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



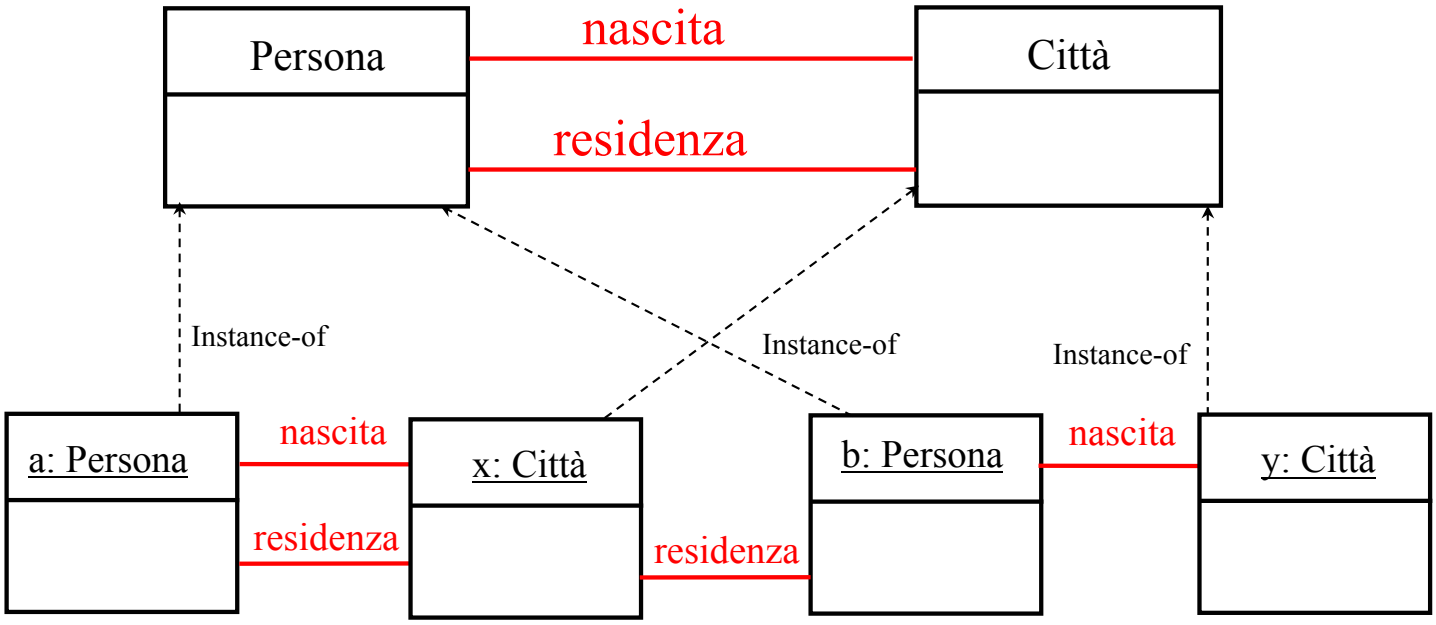
Istanze di associazioni: link

- Come gli oggetti sono istanze delle classi, così i link sono istanze delle associazioni (gli archi instance-of non sono necesssari)
- 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:



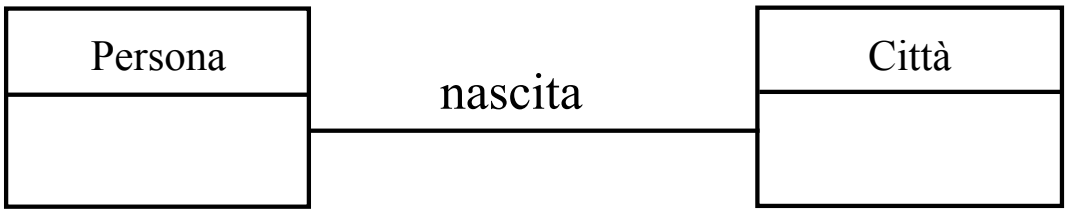
Più associazioni tra due classi

- 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:



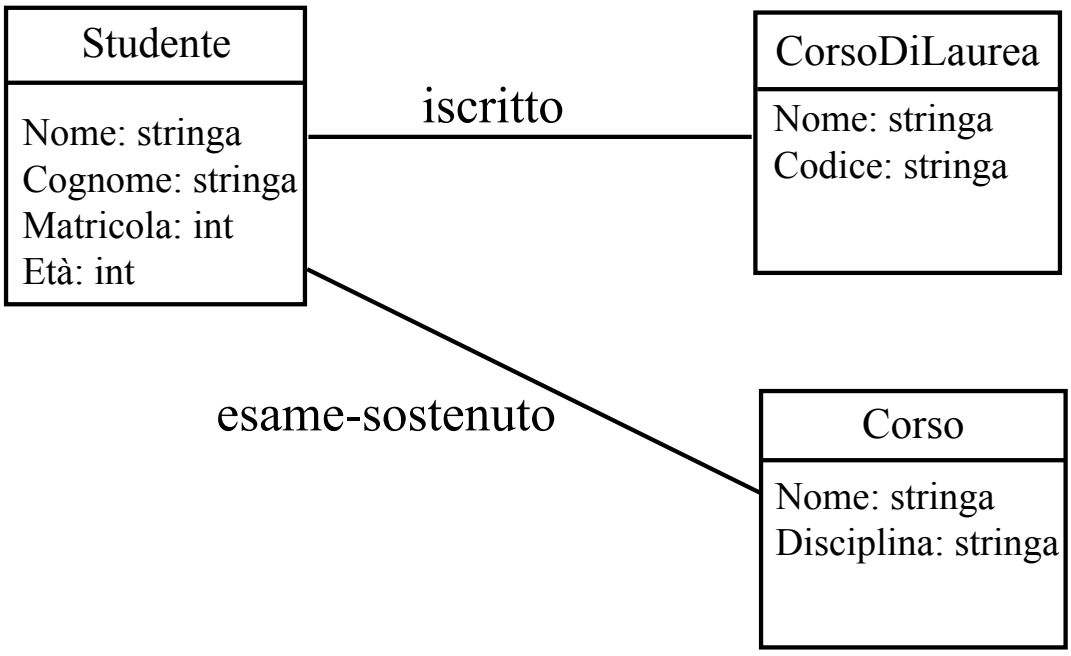
  - Una istanza di Persona può essere legata a zero, una, o più istanze di Città da link di tipo “nascita”
  - Vedremo successivamente come si possono specificare condizioni sul numero di link che coinvolgono un oggetto in UML (ad esempio per imporre che ogni istanza deve avere esattamente un link di tipo “nascita” con una istanza di Città)

Esempio

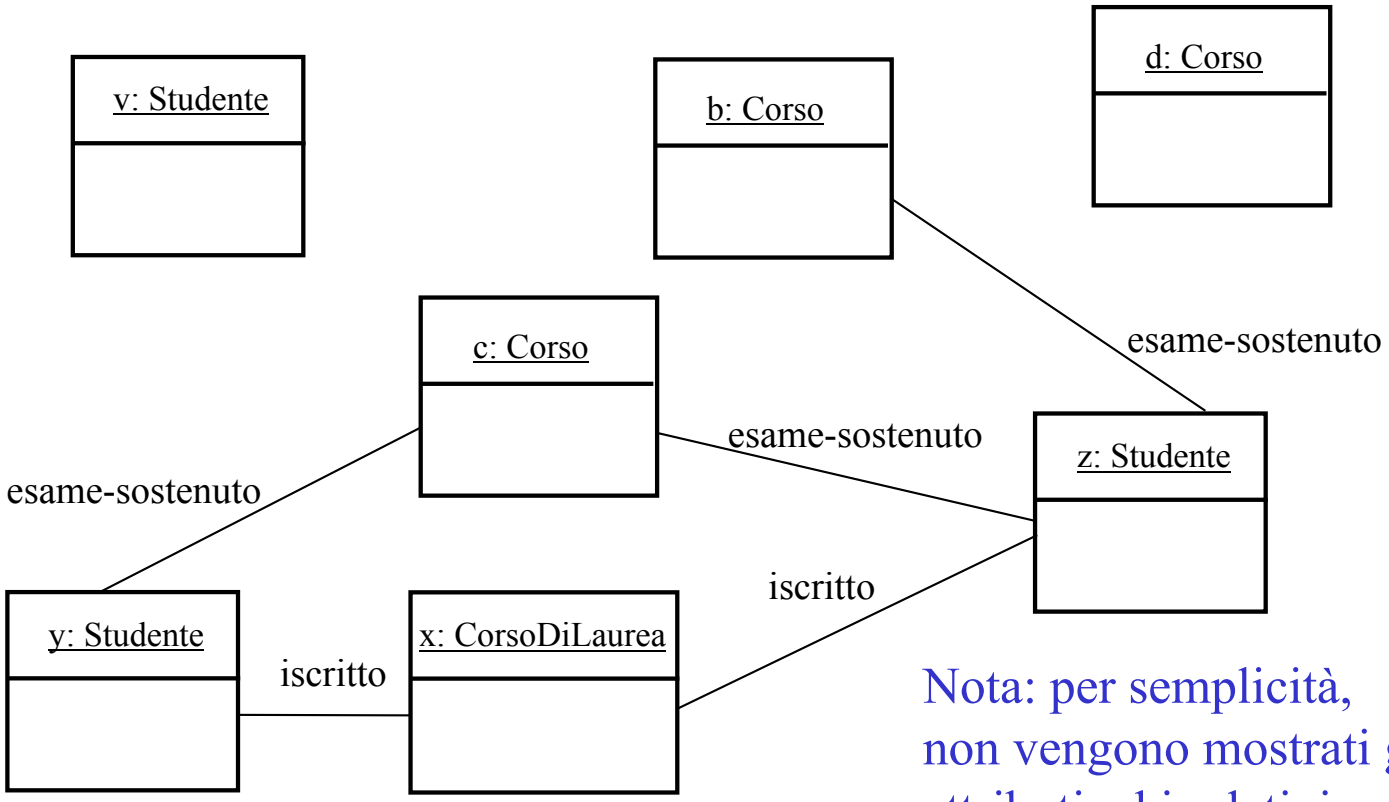
Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

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. 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.).

Diagramma delle classi per l'esempio



Esempi di oggetti



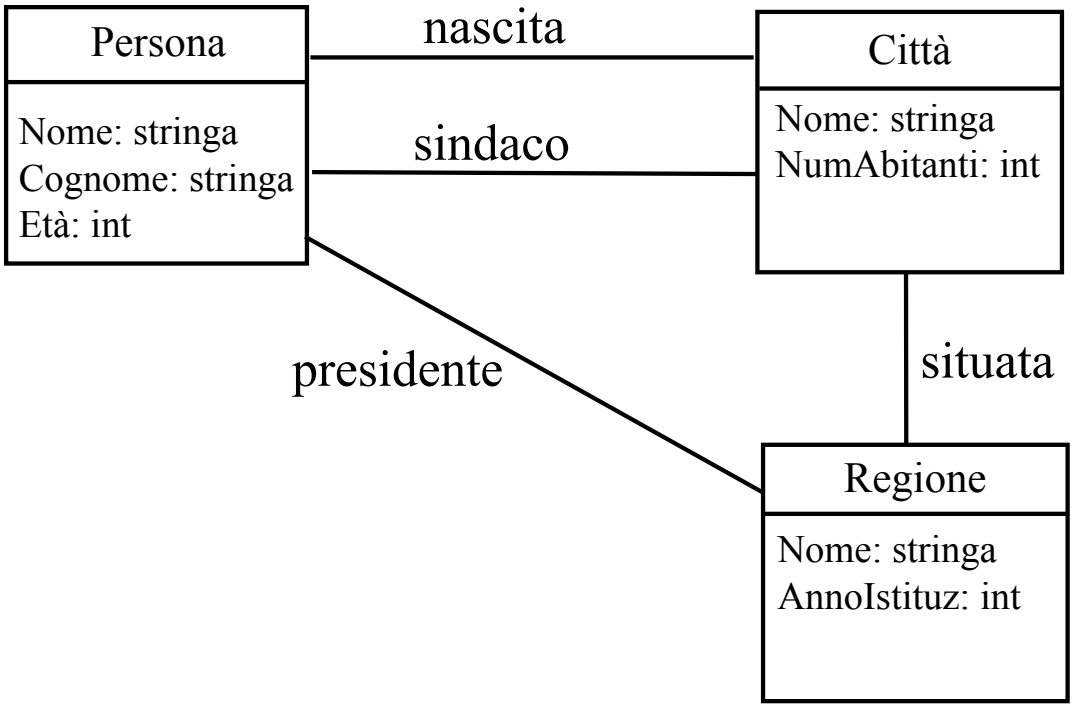
Nota: per semplicità, non vengono mostrati gli attributi ed i relativi valor

Esercizio 2

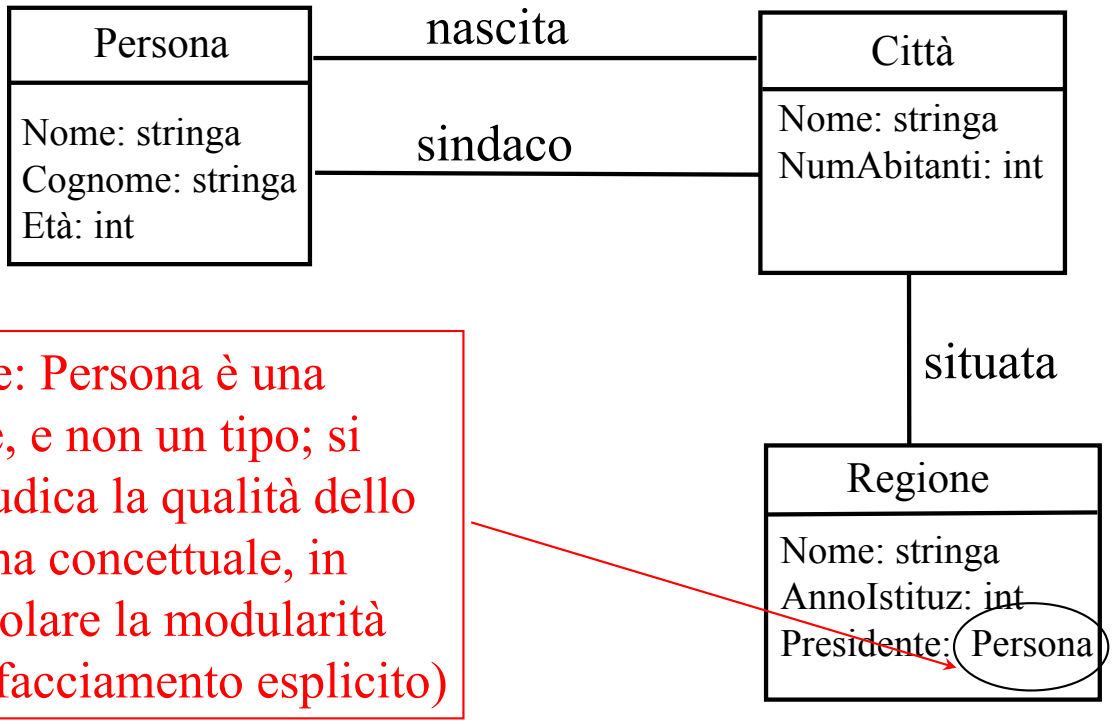
Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

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).

Soluzione dell'esercizio 2

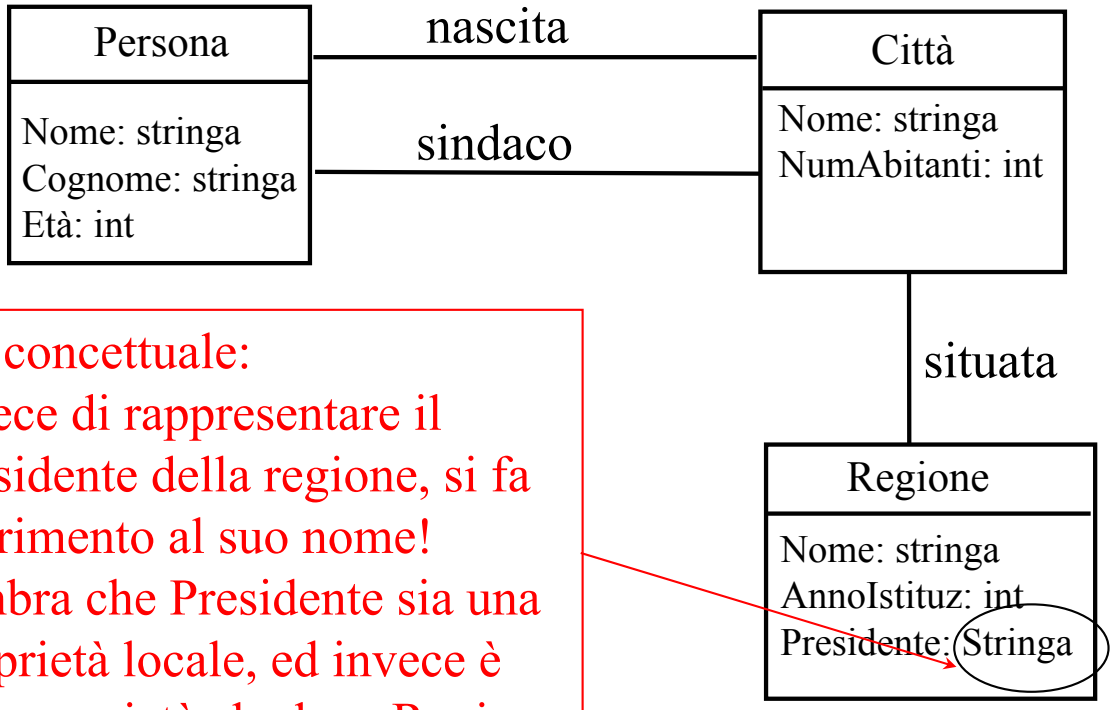


Possibile errore nell'esercizio 2



Errore: Persona è una classe, e non un tipo; si pregiudica la qualità dello schema concettuale, in particolare la modularità (interfacciamento esplicito)

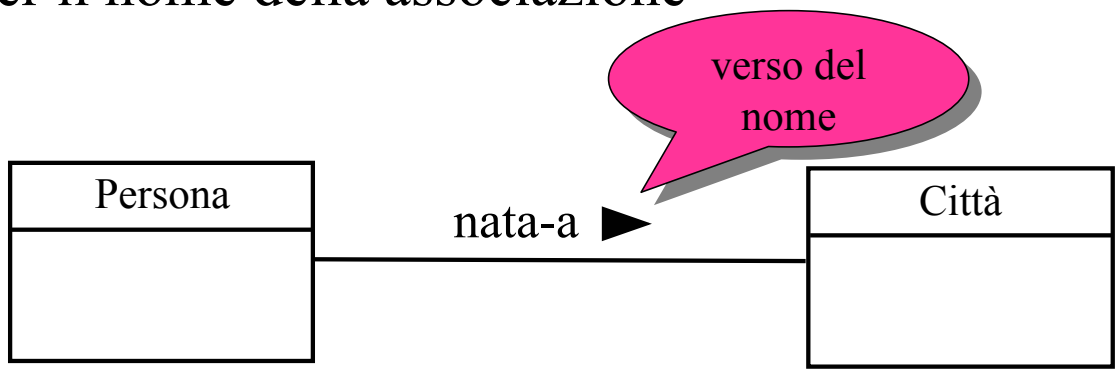
Possibile errore nell'esercizio 2



Errore concettuale:
- invece di rappresentare il Presidente della regione, si fa riferimento al suo nome!
- sembra che Presidente sia una proprietà locale, ed invece è una proprietà che lega Regione a Persona.

Nomi di associazioni

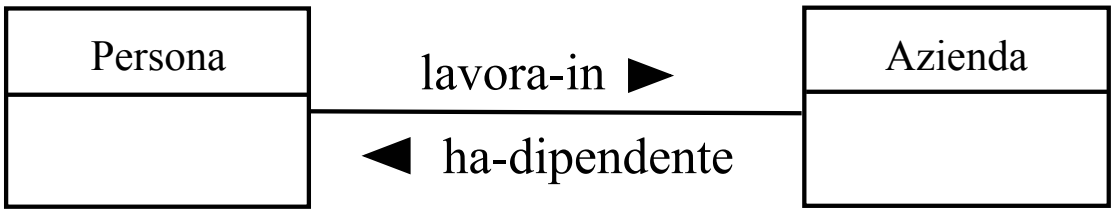
- Alcune volte è interessante specificare un verso per il nome della associazione



- Attenzione: la notazione riportata sopra non significa che l'associazione ha 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

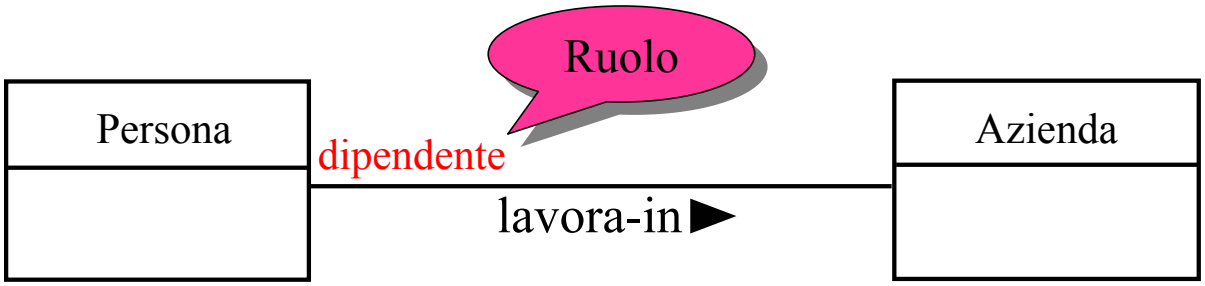
- È anche possibile assegnare due nomi con i relativi versi alla stessa associazione



- 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



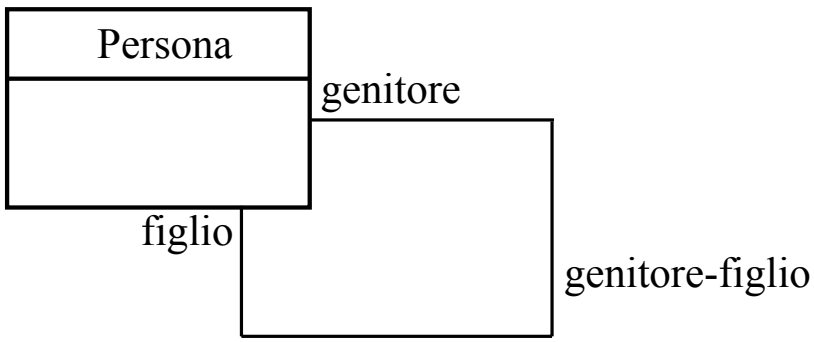
- Il ruolo si indica con un nome posizionato lungo la linea che rappresenta l'associazione, vicino alla classe alla quale si riferisce
- Nell'esempio, dipendente è il ruolo che la persona gioca nell'associazione "lavora-in" con Azienda

Ruoli nei link

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

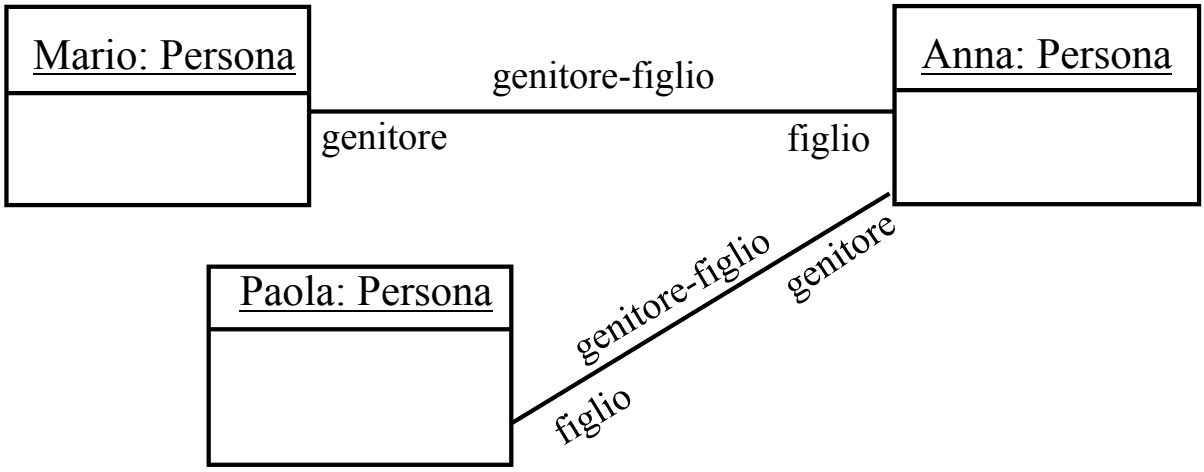
Ancora sui ruoli

- 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:



Osservazioni sui ruoli

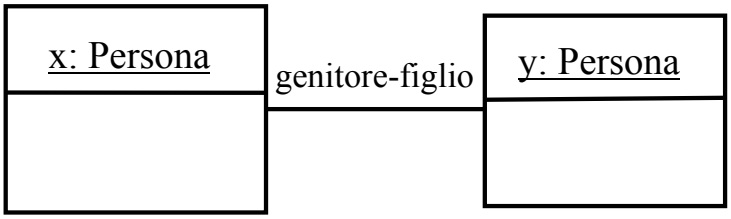
Esempio della necessità dei ruoli nei link:



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

Importanza dei ruoli

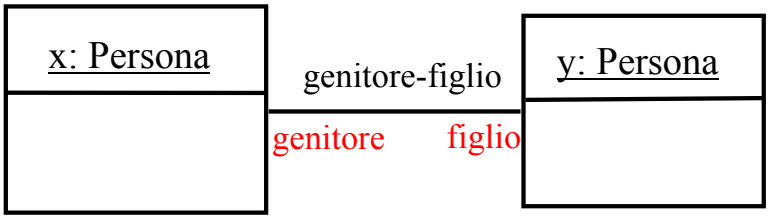
Senza ruoli non è chiaro chi è il genitore e chi il figlio. In altre parole, dalla coppia <*x,y*> in genitore-figlio non si evincono i ruoli di *x* e *y*



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

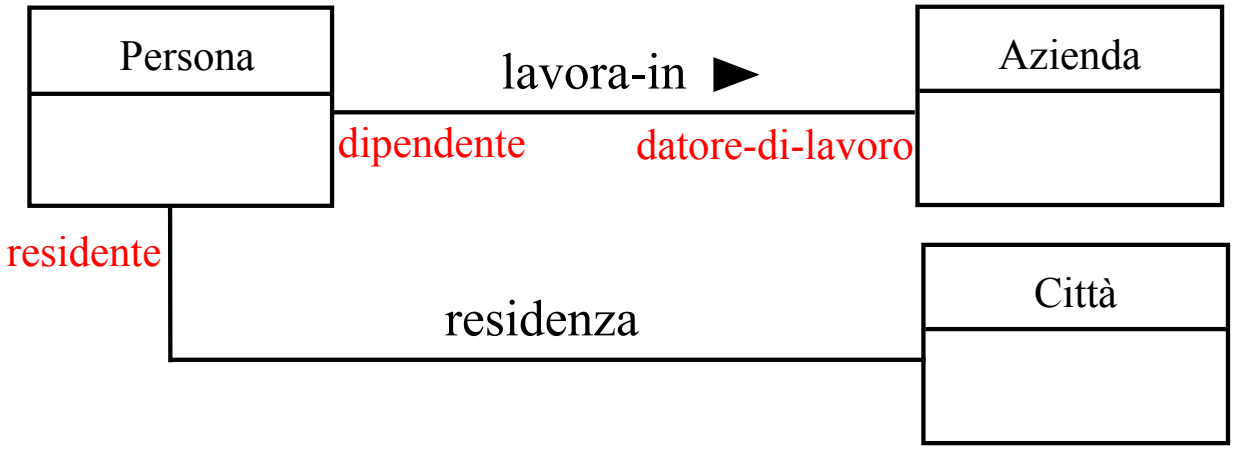
<genitore:x, figlio:y>

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



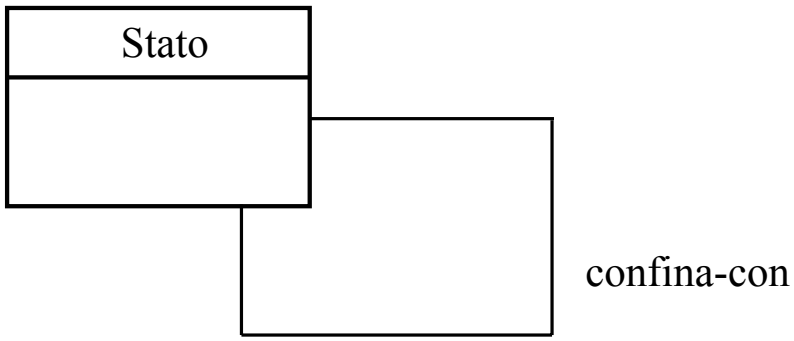
Importanza dei ruoli

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

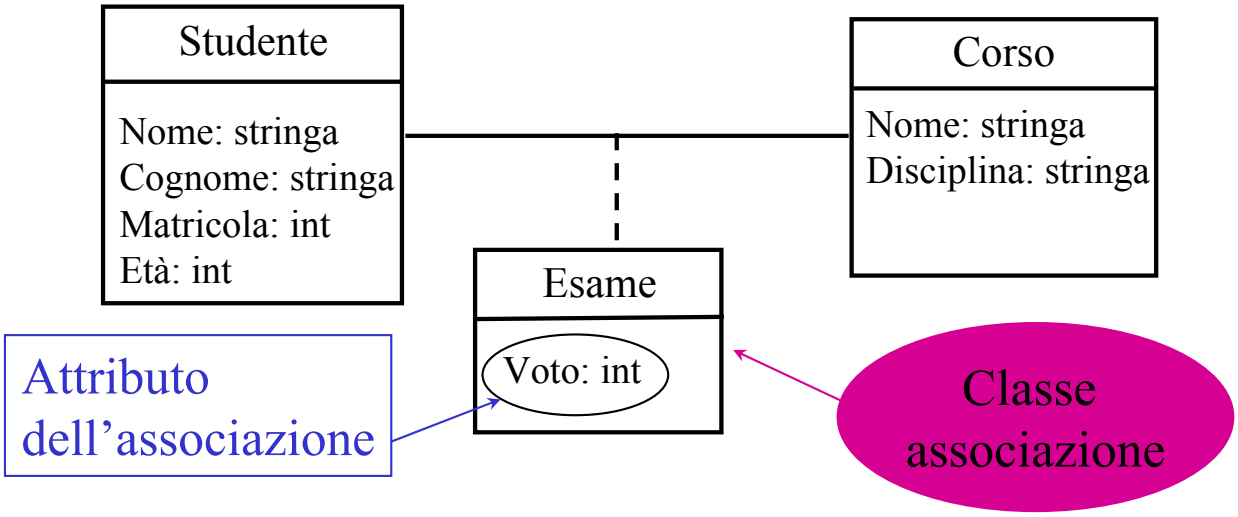
- Ci sono casi in cui l'associazione insiste più volte sulla stessa classe, ma il ruolo non è significativo, in particolare quando l'associazione rappresenta una relazione simmetrica
- Esempio:



Attributi di associazione

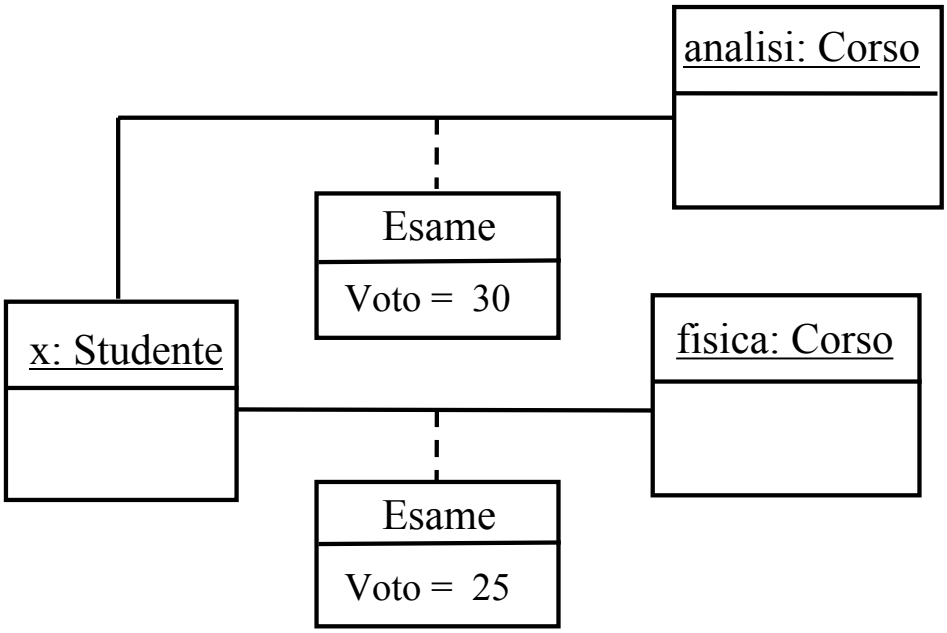
Analogamente alle classi, anche le associazioni possono avere attributi. Formalmente, un attributo di una associazione è una funzione che associa ad ogni link che è istanza dell'associazione un valore di un determinato tipo

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



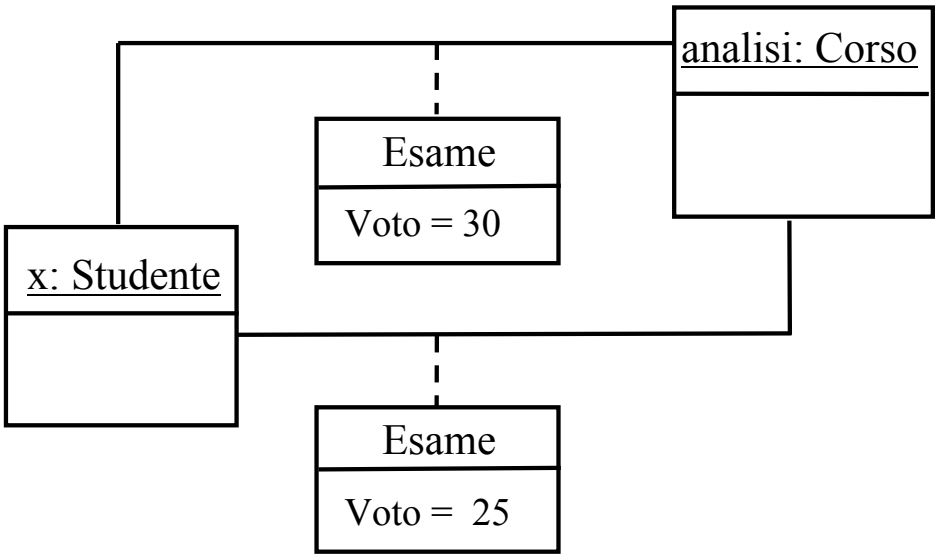
Attributi nei link

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



Attributi nei link

Con la presenza degli attributi, il significato dell'associazione (relazione matematica) non cambia. Quindi questo è un errore:

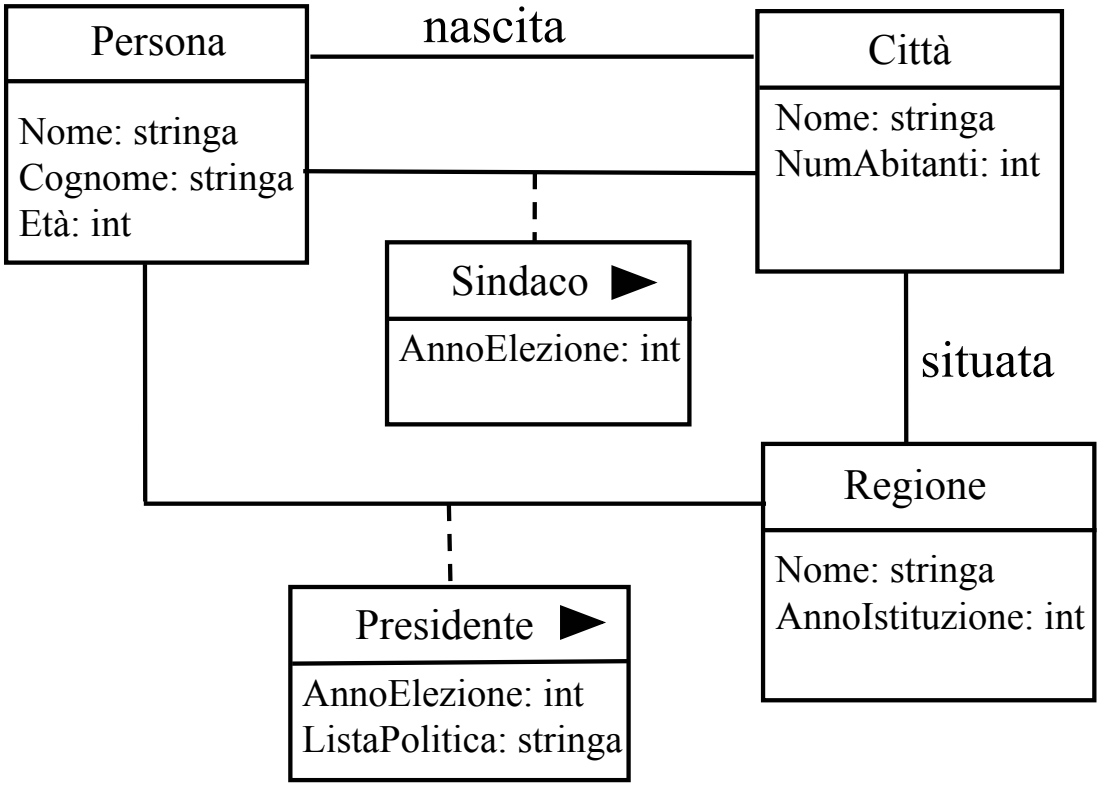


Esercizio 3

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

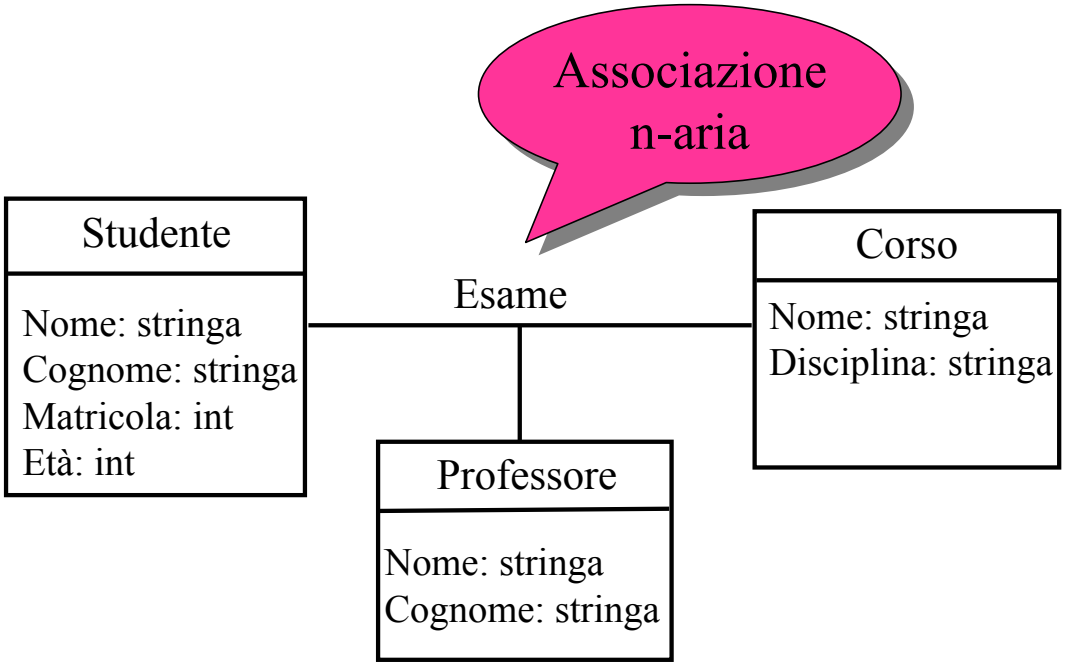
Si vogliono modellare le persone (con nome, cognome, età), le città di nascita (con nome, numero di abitanti, e sindaco, compresa l'indicazione dell'anno di elezione), e le regioni in cui si trovano le città (con nome, anno di istituzione, e presidente, con l'anno di elezione e lista politica in cui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

Soluzione dell'esercizio 3



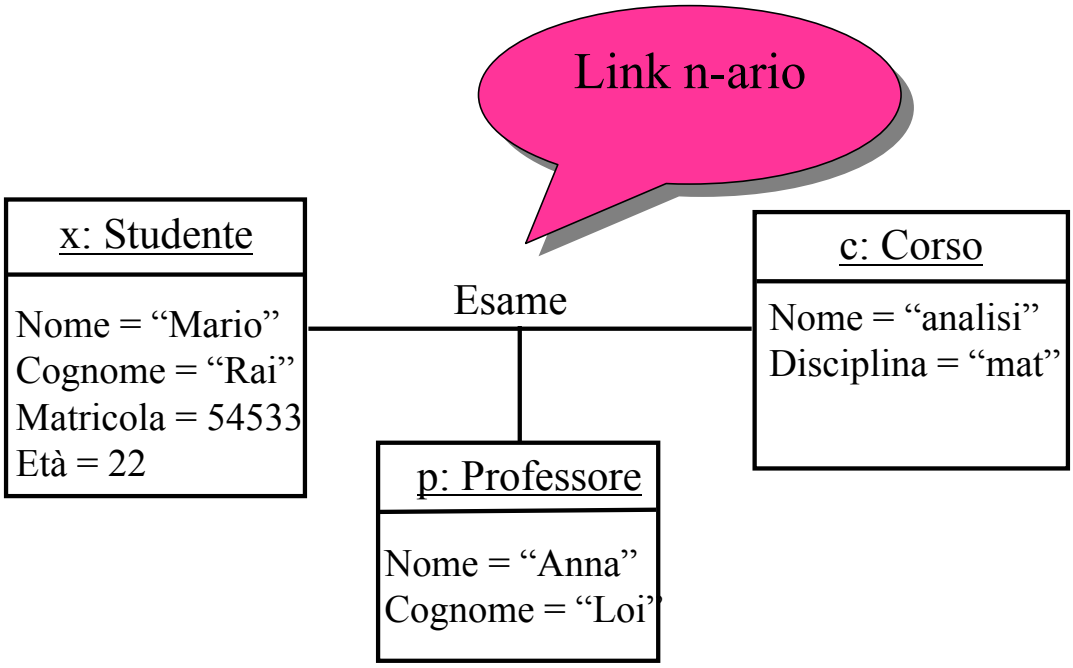
Associazioni n-arie

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



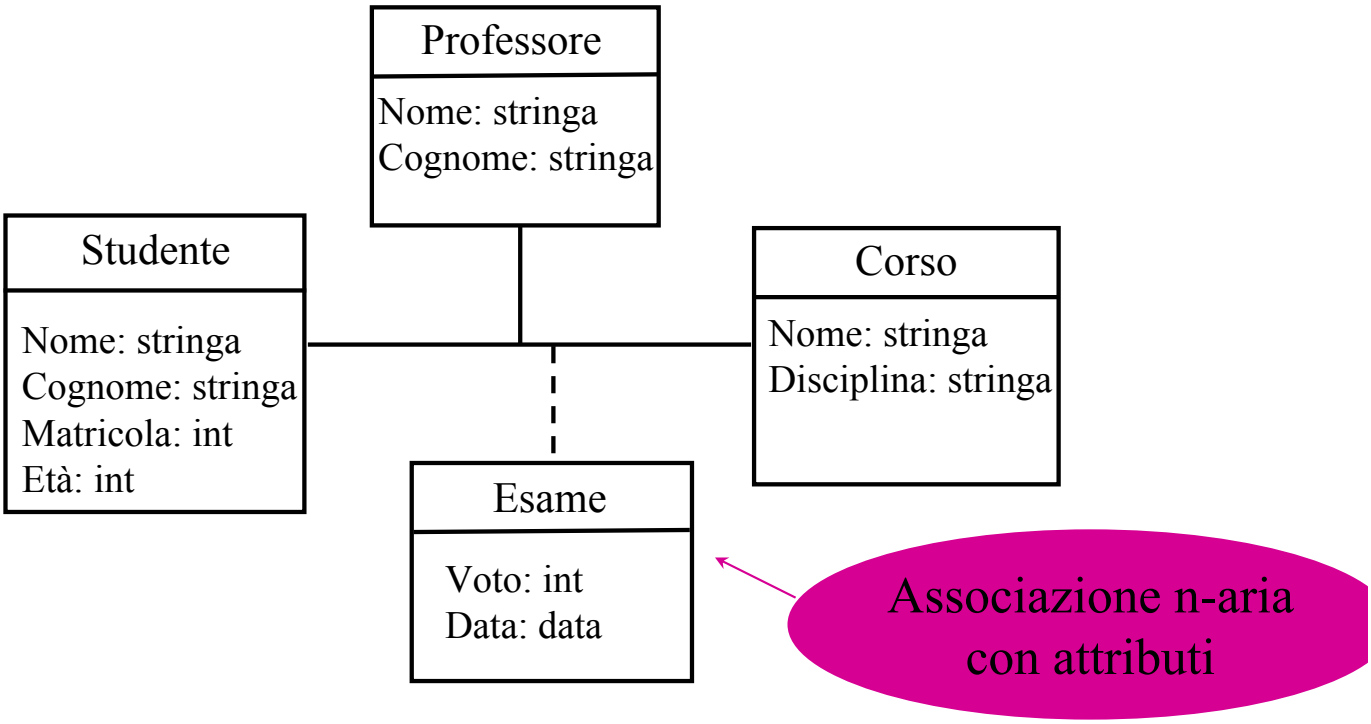
Istanze di associazioni n-arie: link n-ari

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



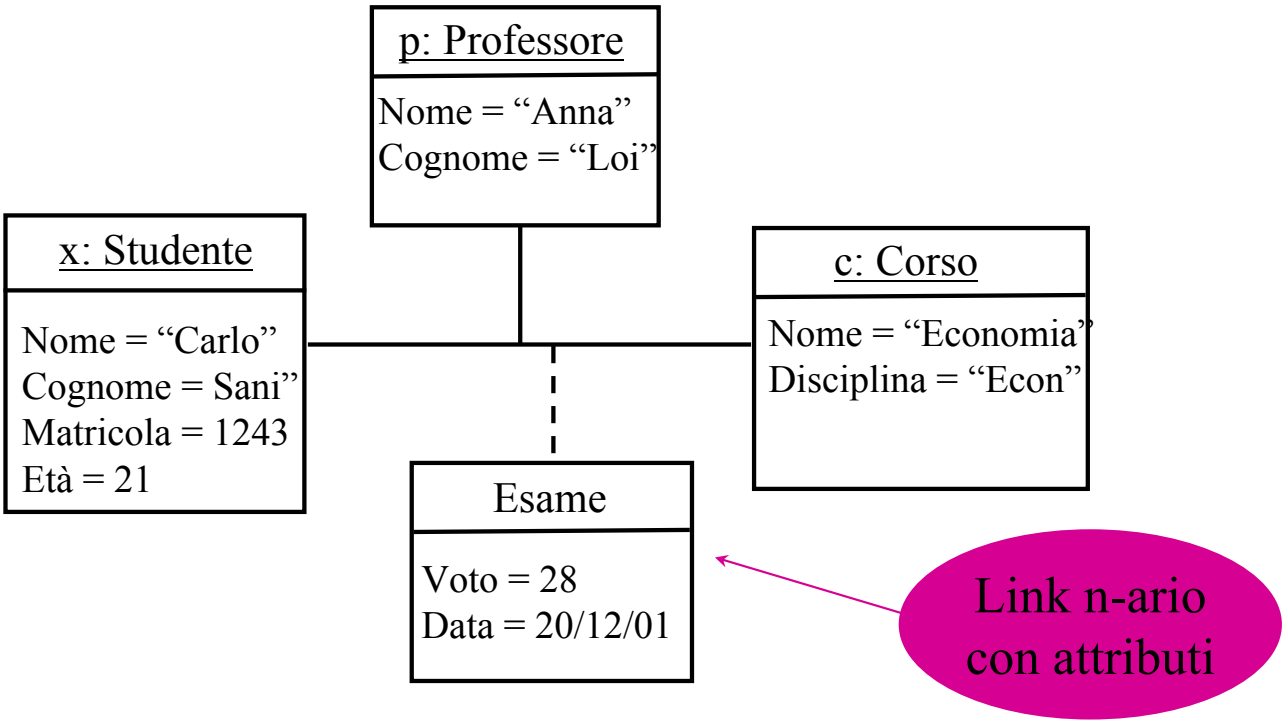
Associazioni n-arie con attributi

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

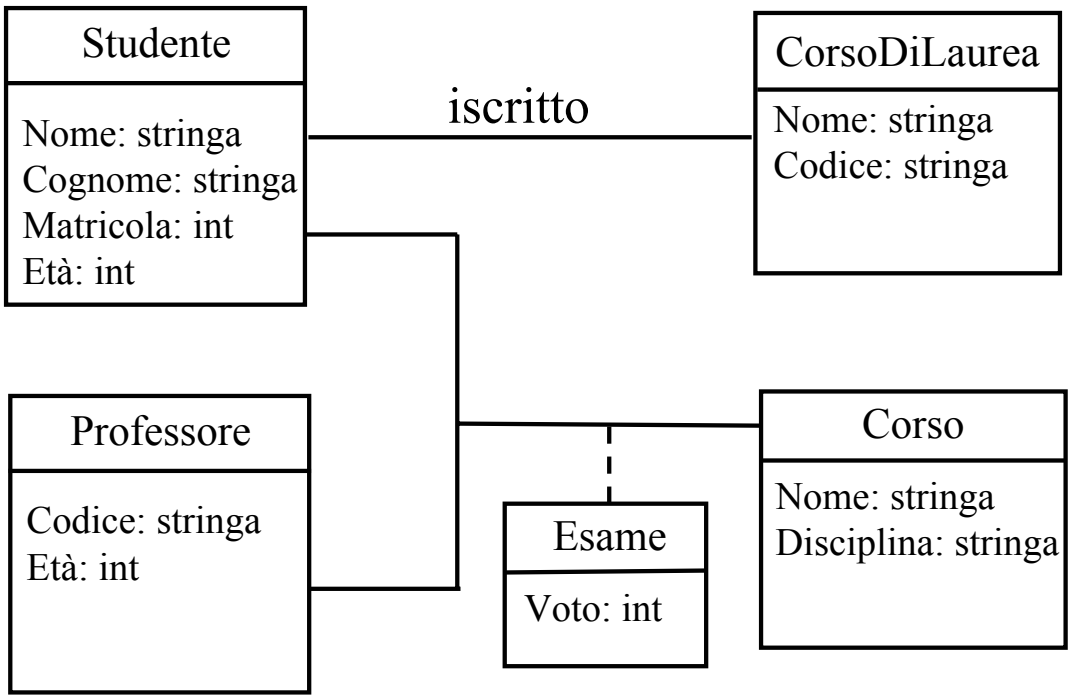


Esempio

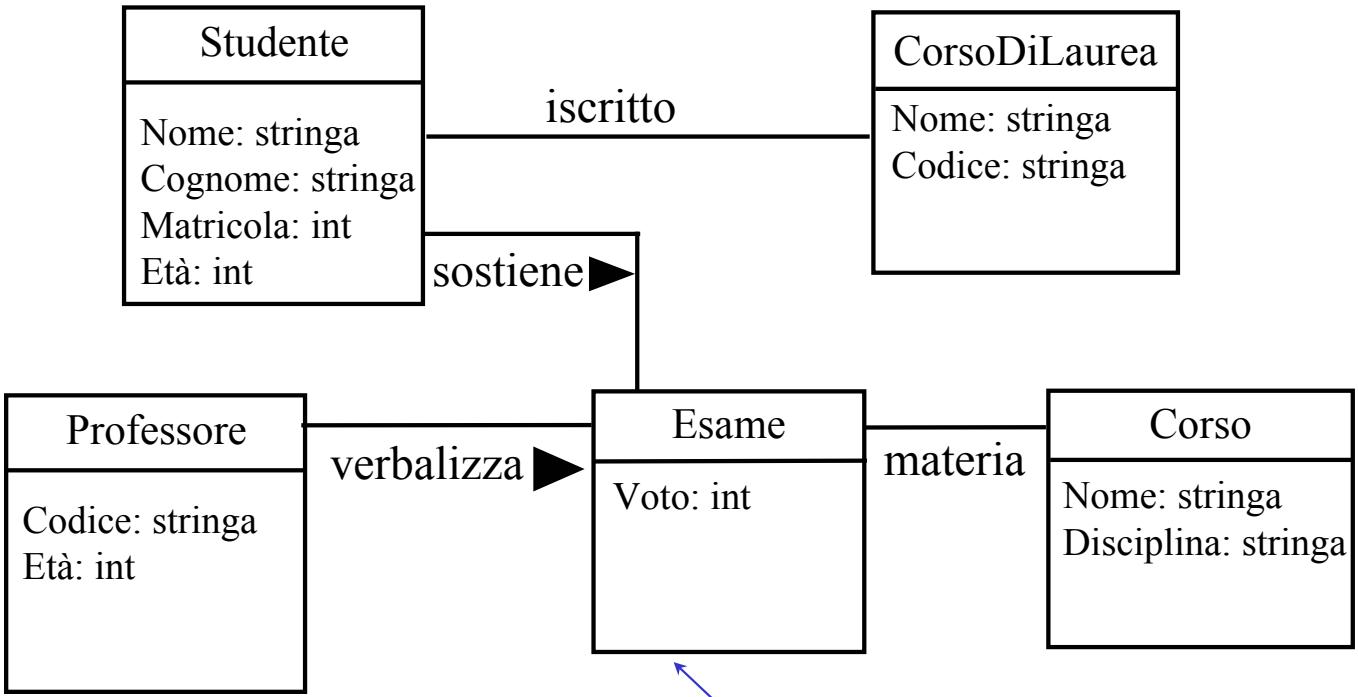
Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

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



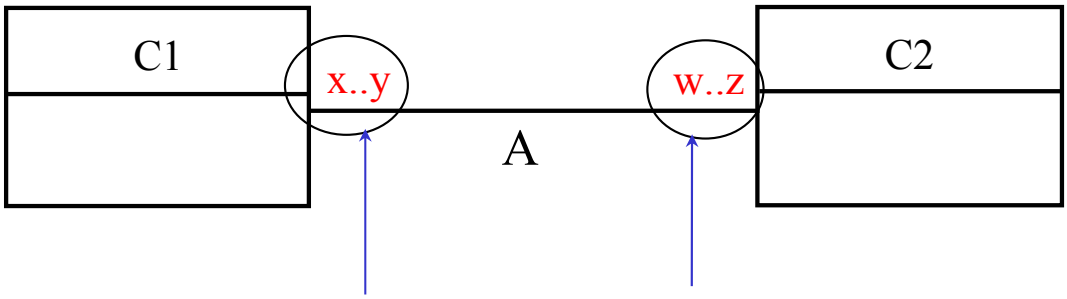
Soluzione scorretta per l'esempio



In questo diagramma, "esame" è una classe. Qual è l'errore?

Molteplicità delle associazioni

- Per specificare con maggiore precisione il significato delle associazioni binarie si possono definire i vincoli di molteplicità (o semplicemente molteplicità) delle associazioni

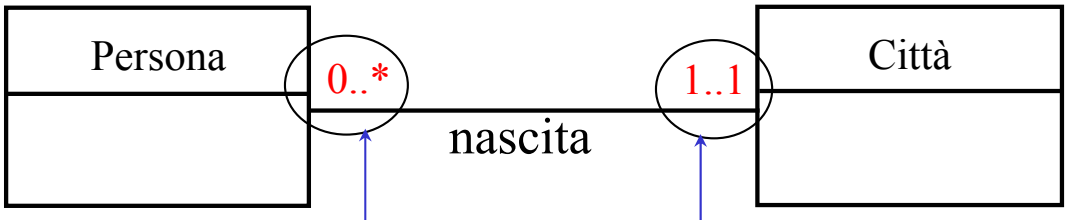


Ogni istanza di C2 è legata ad almeno x e al massimo y istanze di C1 mediante l'associazione A

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

Molteplicità delle associazioni

- Esempio: 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

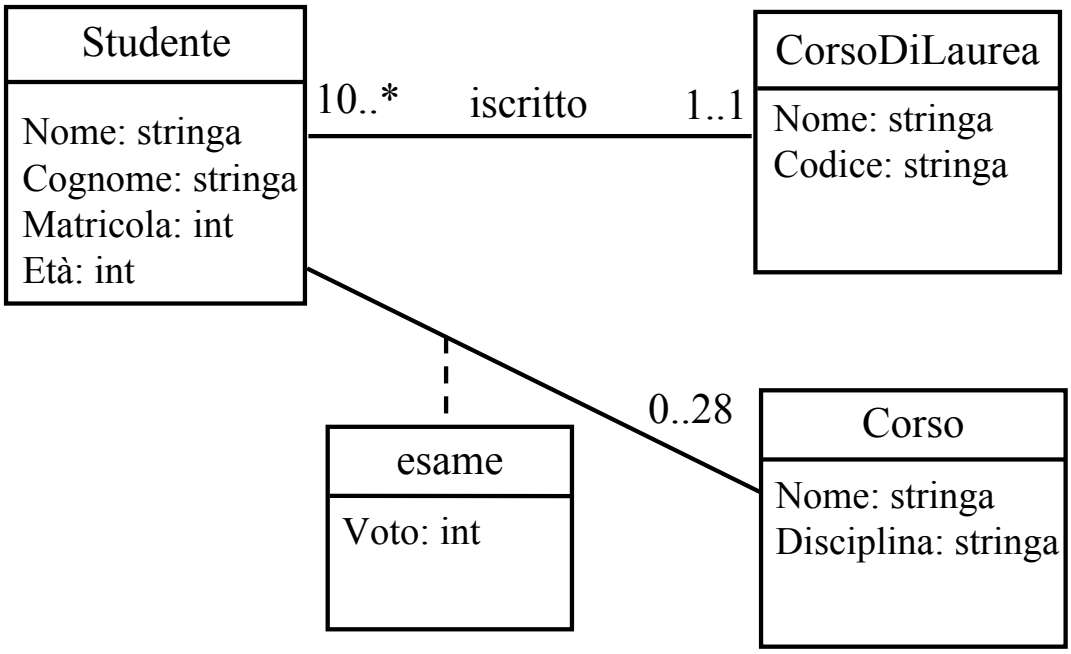
Vincolo di molteplicità: ogni persona ha esattamente una città di nascita

Molteplicità delle associazioni: notazione

- Le molteplicità si definiscono solo per le associazioni n-arie
- Possibili molteplicità:

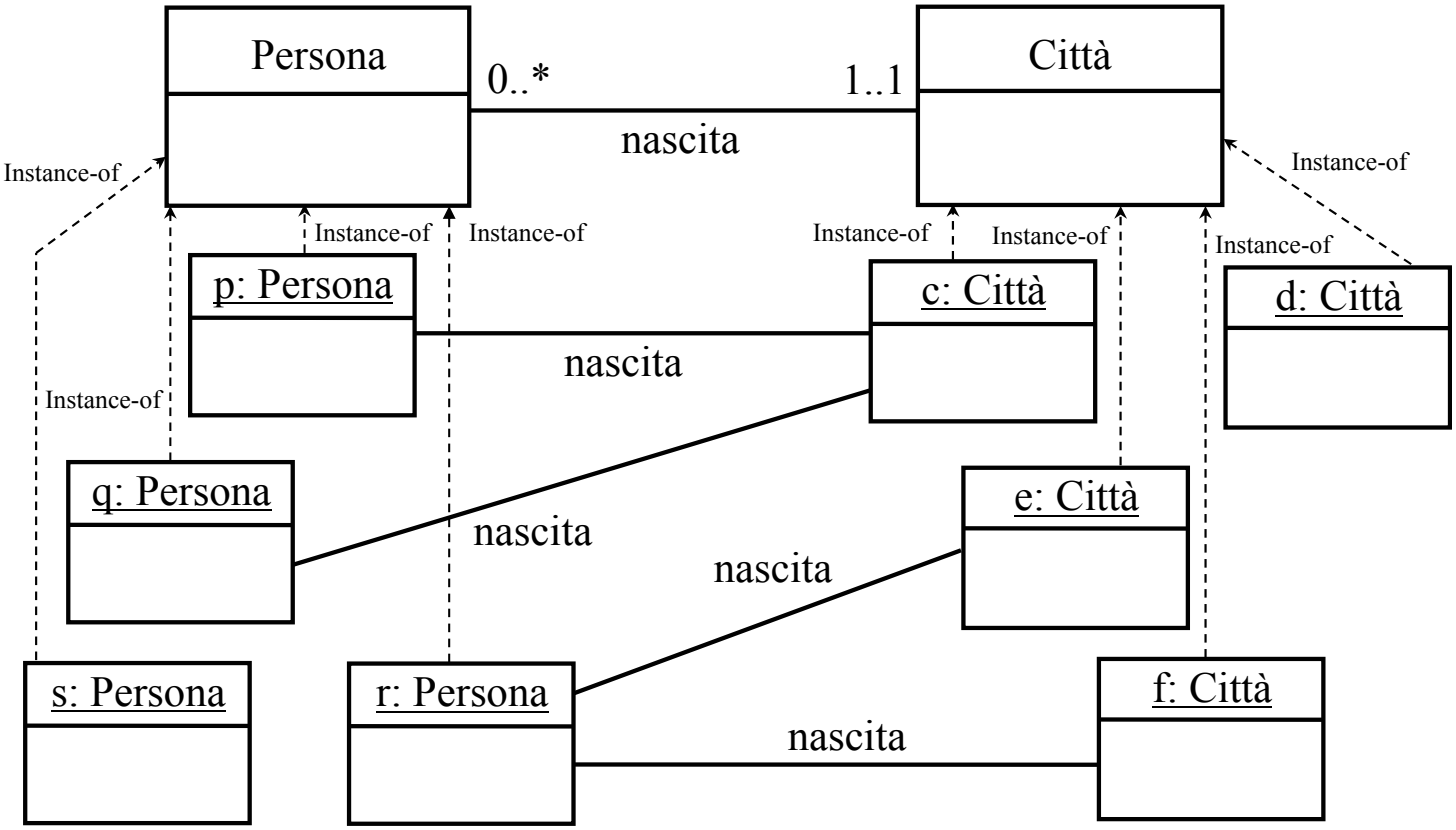
0 .. *	(nessun vincolo, si può evitare di indicare)
0 .. 1	(nessun limite per il minimo, e al massimo una)
1 .. *	(al minimo una, e nessun limite per il massimo)
1 .. 1	(esattamente una)
0 .. y	(nessun limite per il minimo, e al massimo y, con y intero > 0)
x .. *	(al minimo x, con x intero ≥ 0, e nessun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

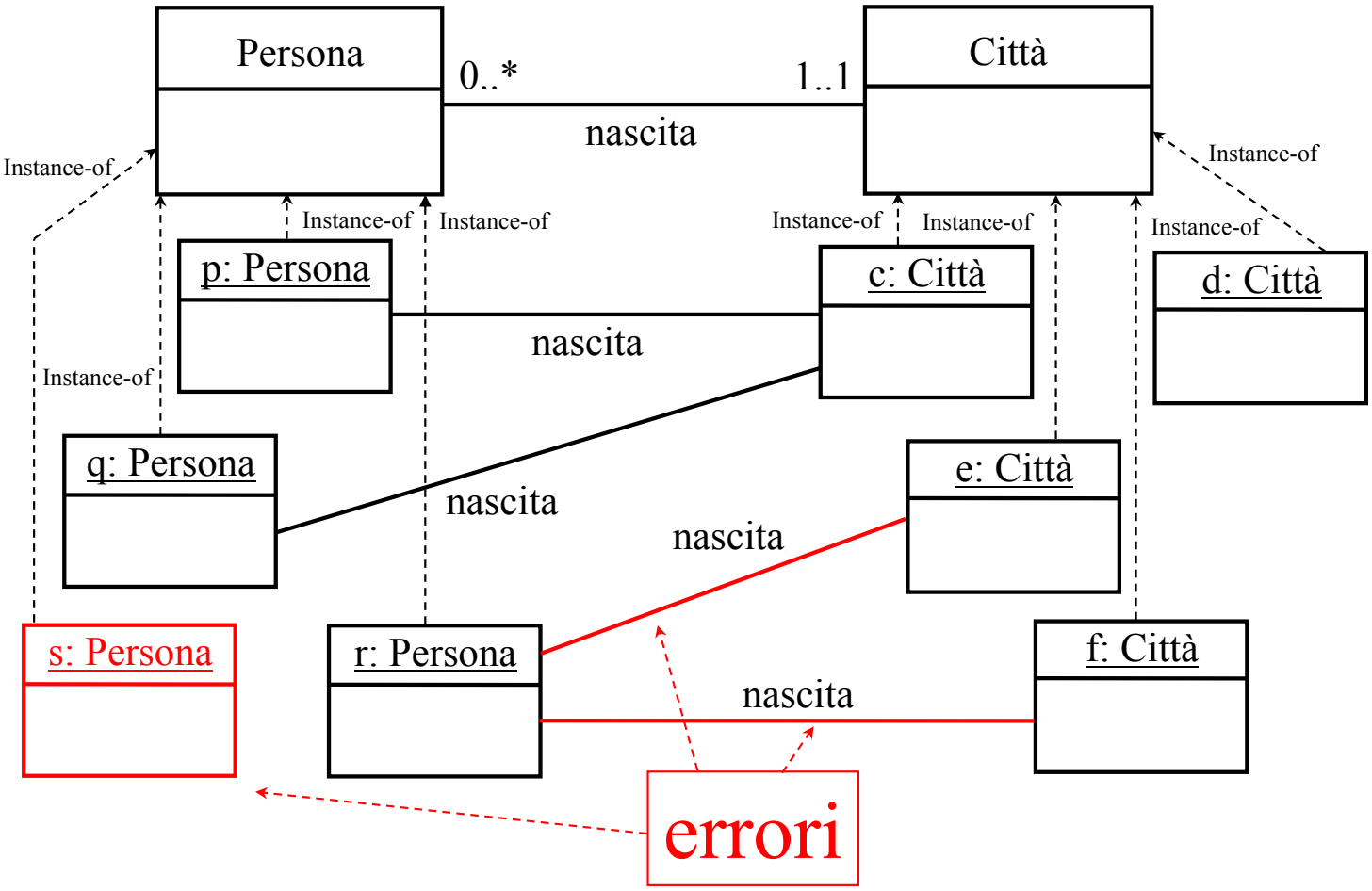


- Ogni studente è iscritto ad un corso di laurea
- Ogni corso di laurea ha almeno 10 iscritti
- Ogni studente può sostenere al massimo 28 esami

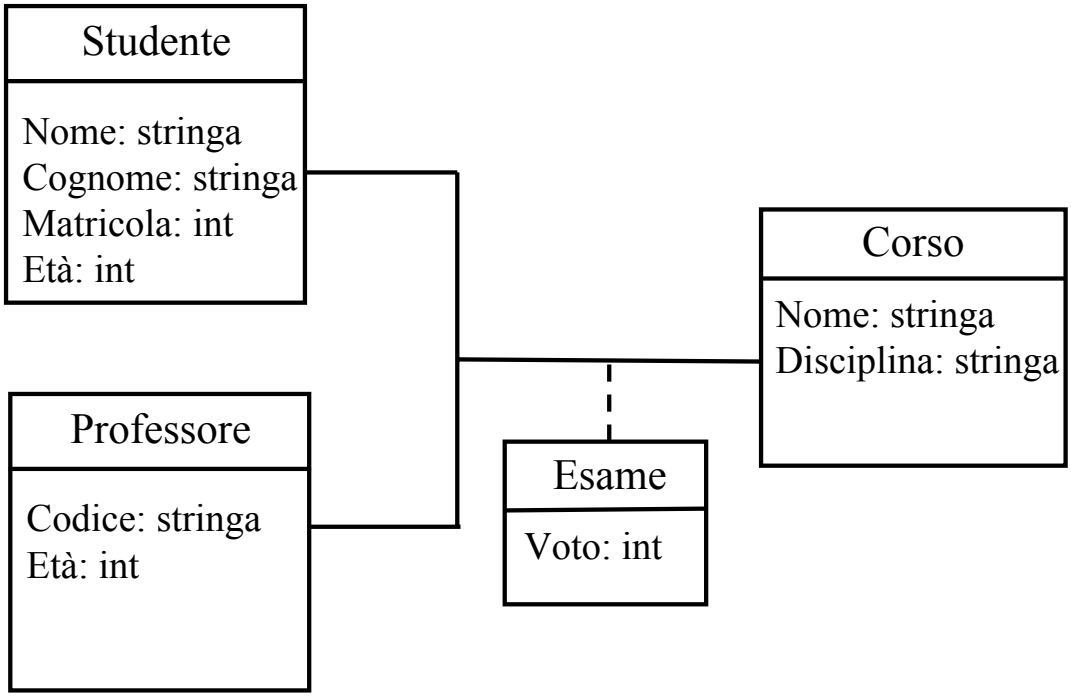
Esercizio 4: individuare gli errori



Soluzione dell'esercizio 4

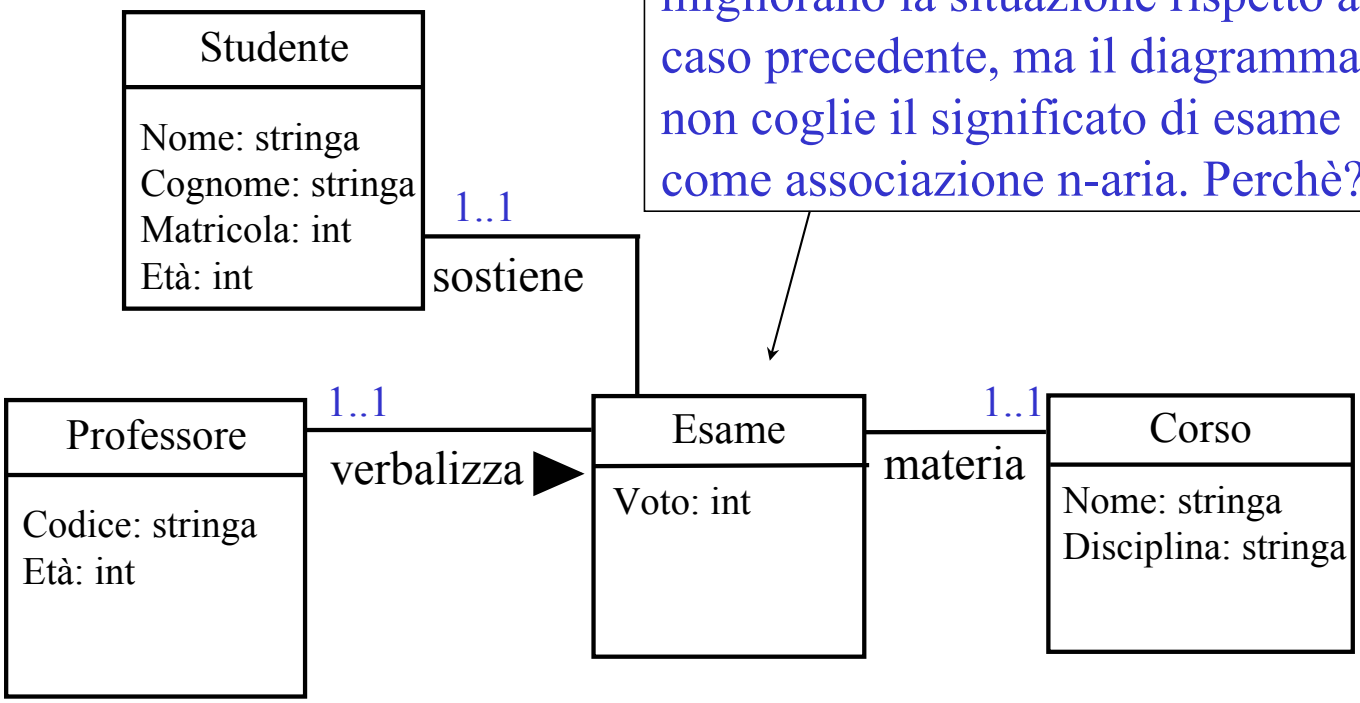


Torniamo alla relazione n-aria



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?



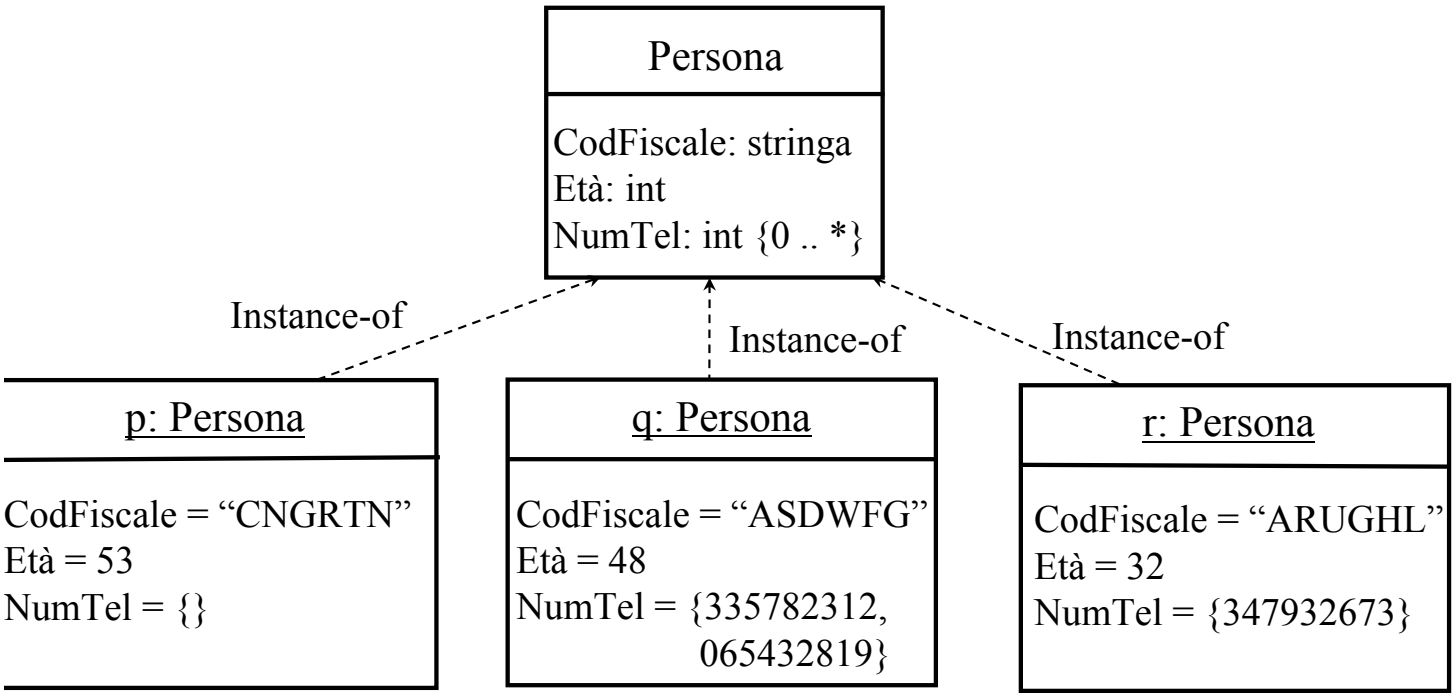
Molteplicità di attributi

- 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
- Al contrario, 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*

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

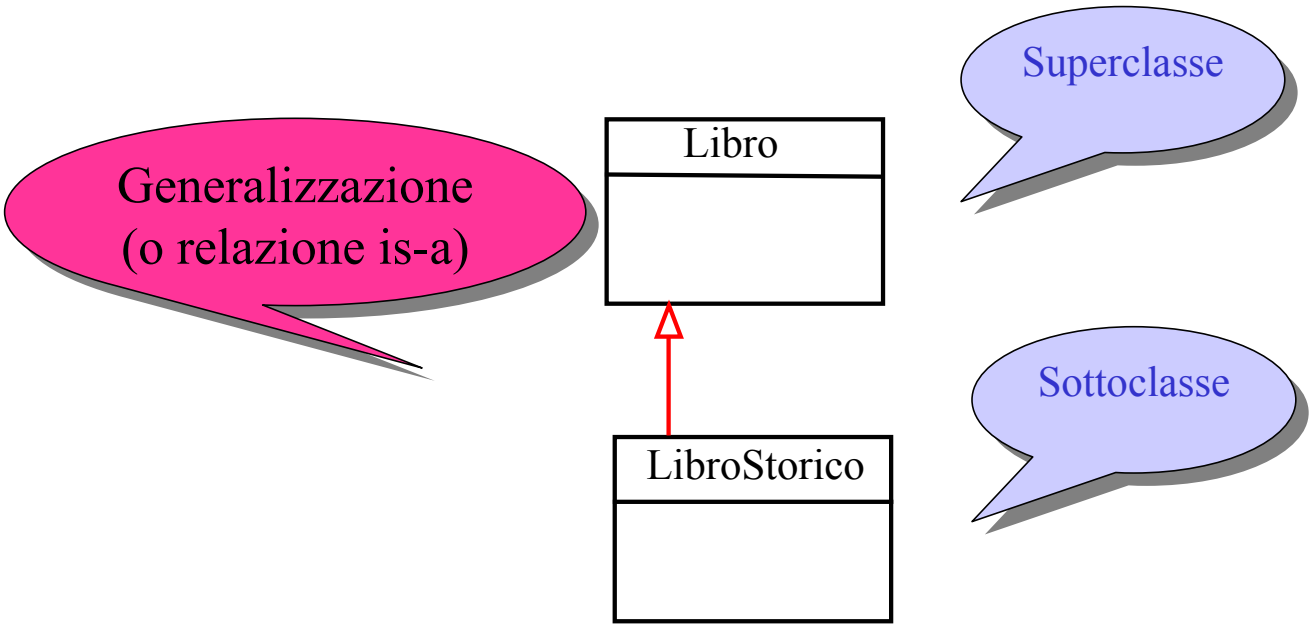


Generalizzazione in UML

- 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

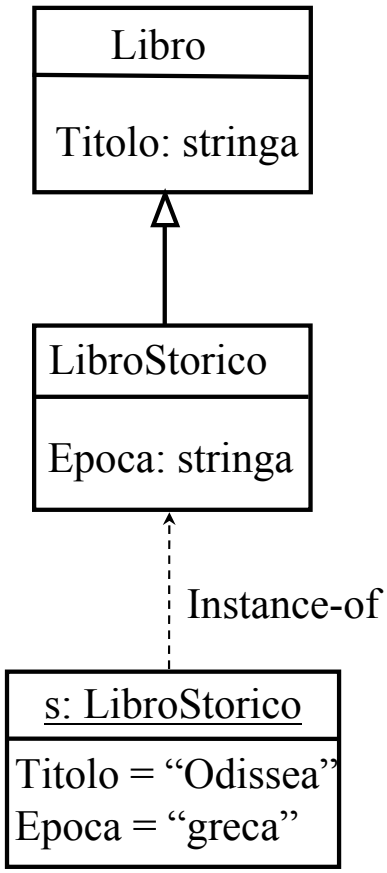
Principio di ereditarietà: ogni proprietà della superclasse è anche una proprietà della sottoclasse, e non si riporta esplicitamente nel diagramma

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

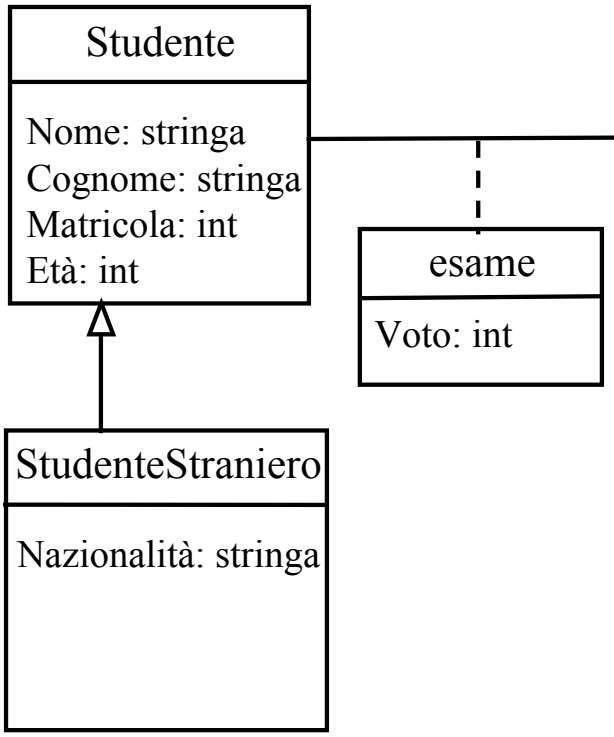
Ragionamento sillogistico (cfr. opera di Aristotele più di due millenni fa)

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

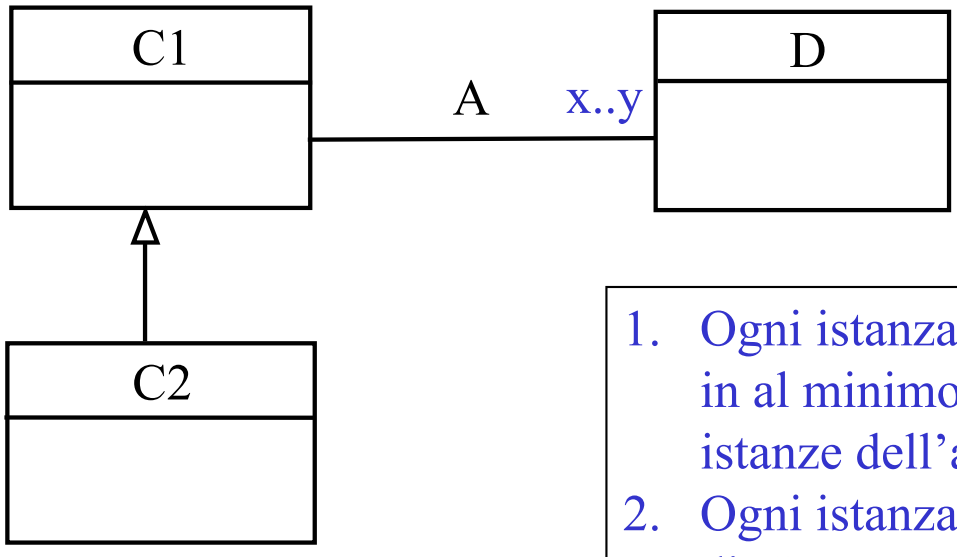


1. Ogni istanza di Studente può essere coinvolta in un numero qualunque di istanze della associazione "esame"
2. Ogni istanza di StudenteStraniero è una istanza di Studente

quindi

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

Ereditarietà sulle molteplicità

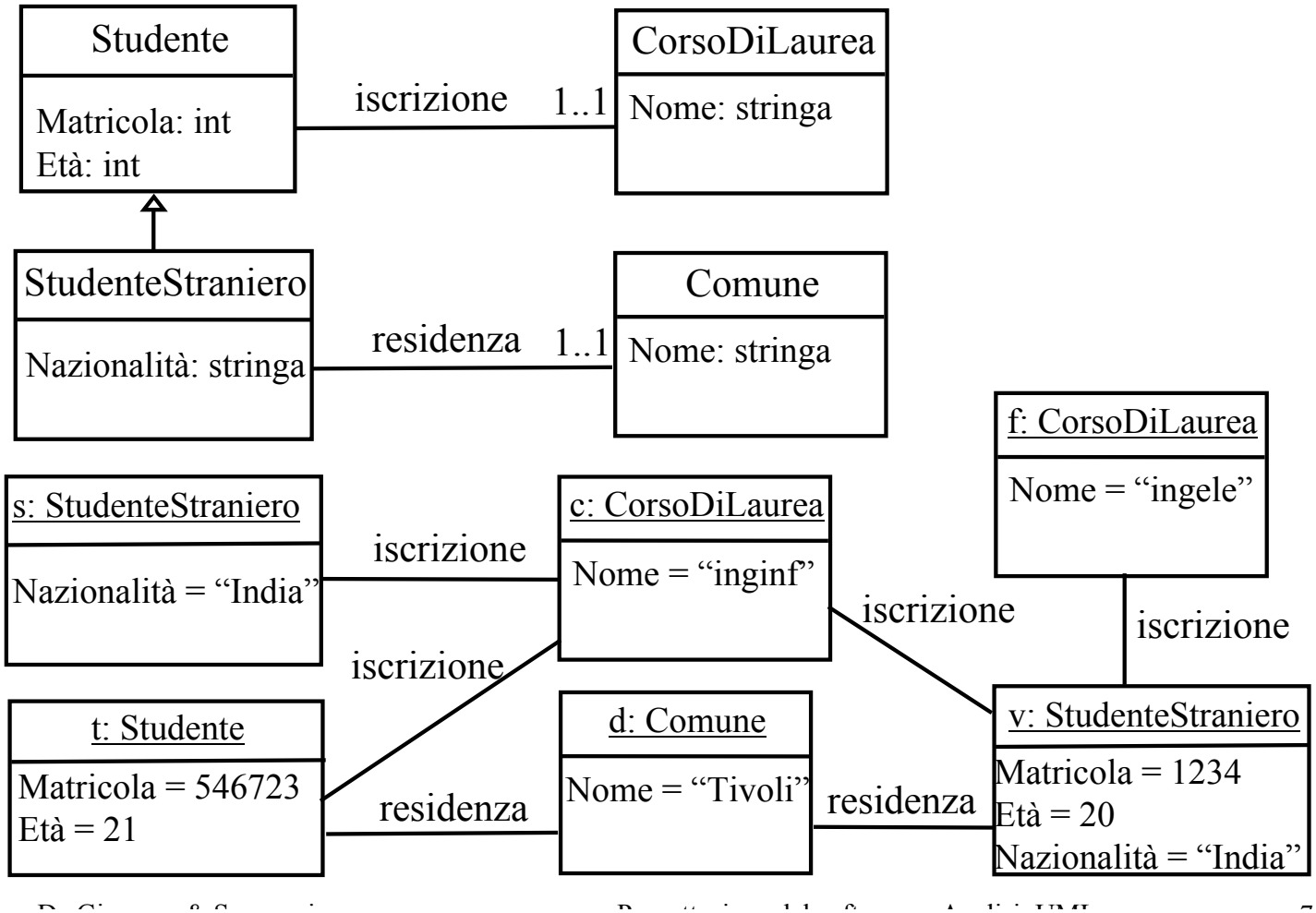


1. Ogni istanza di C_1 è coinvolta in al minimo x e al massimo y istanze dell'associazione A
2. Ogni istanza di C_2 è una istanza di C_1

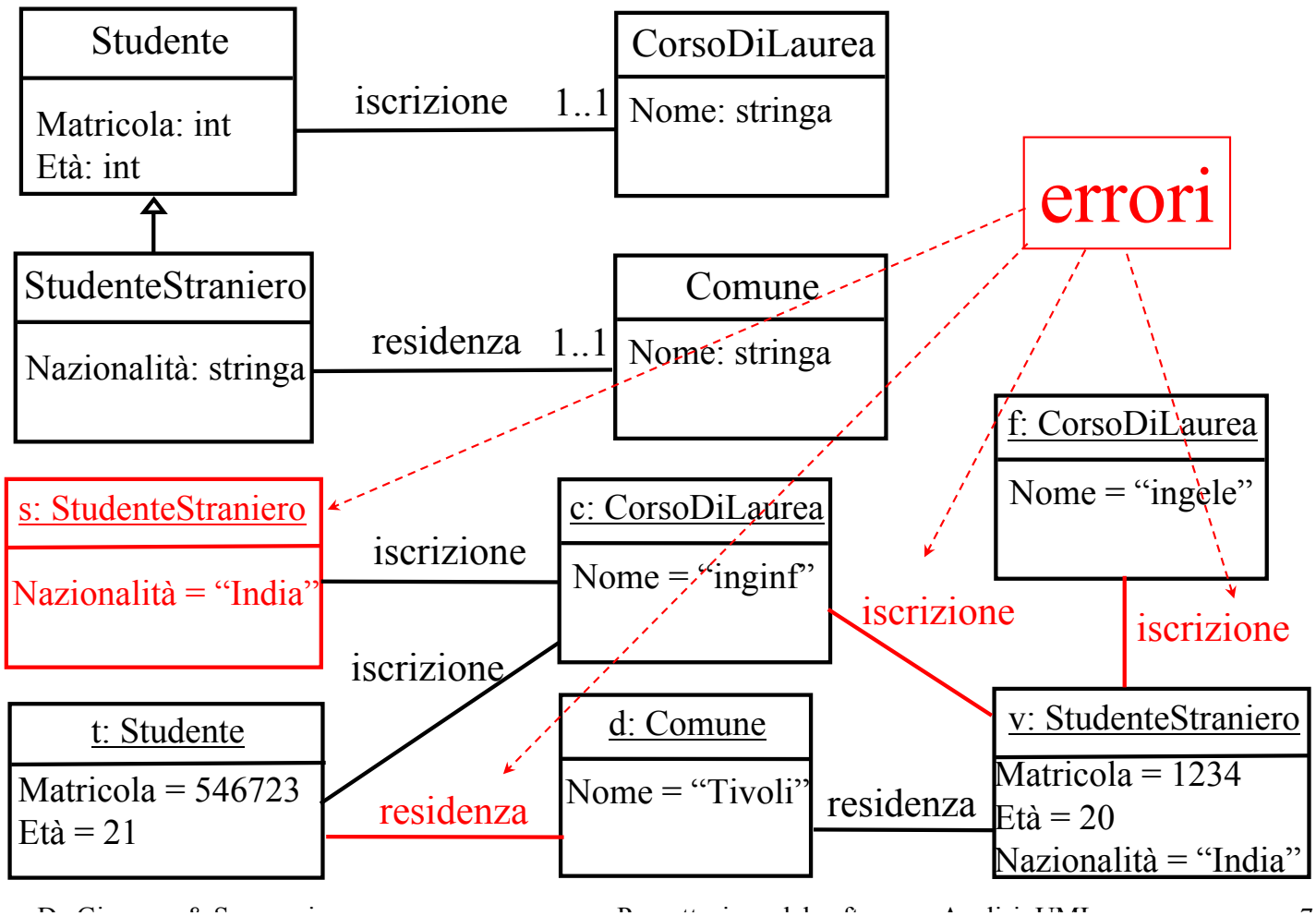
quindi

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

Esercizio 5: individuare gli errori

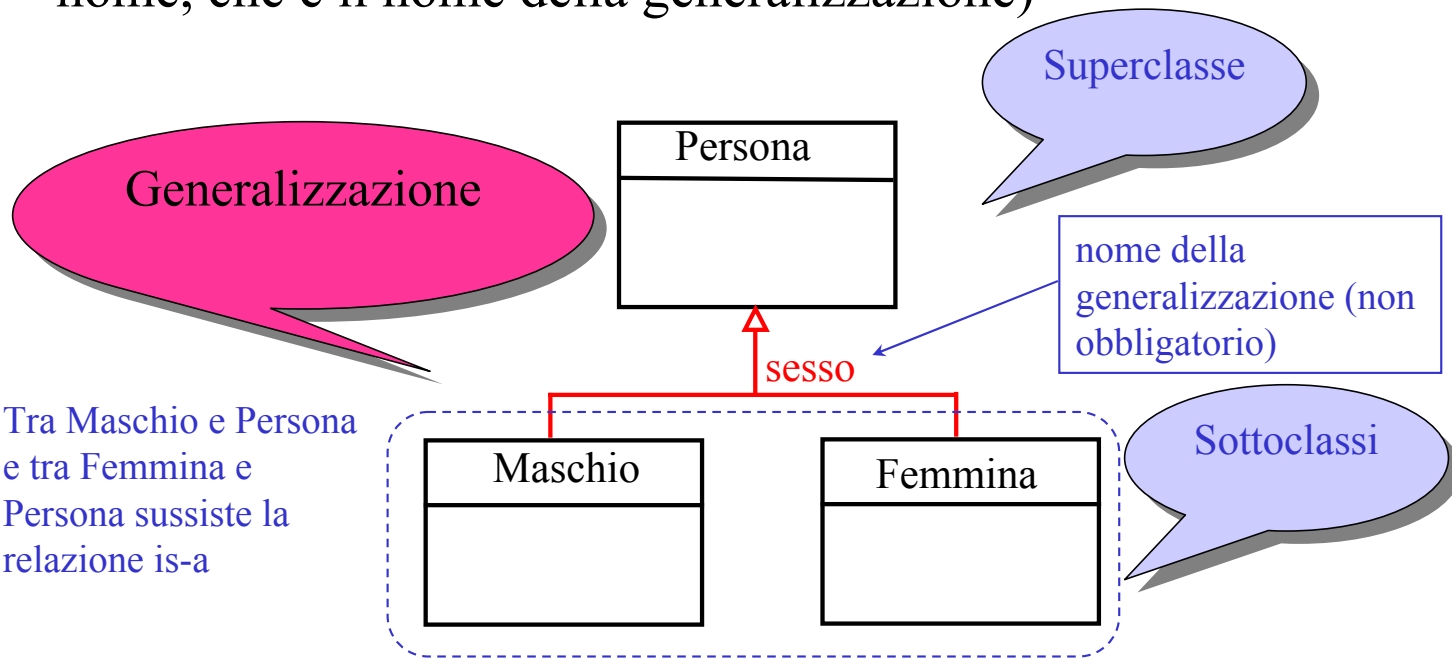


Soluzione dell'esercizio 5



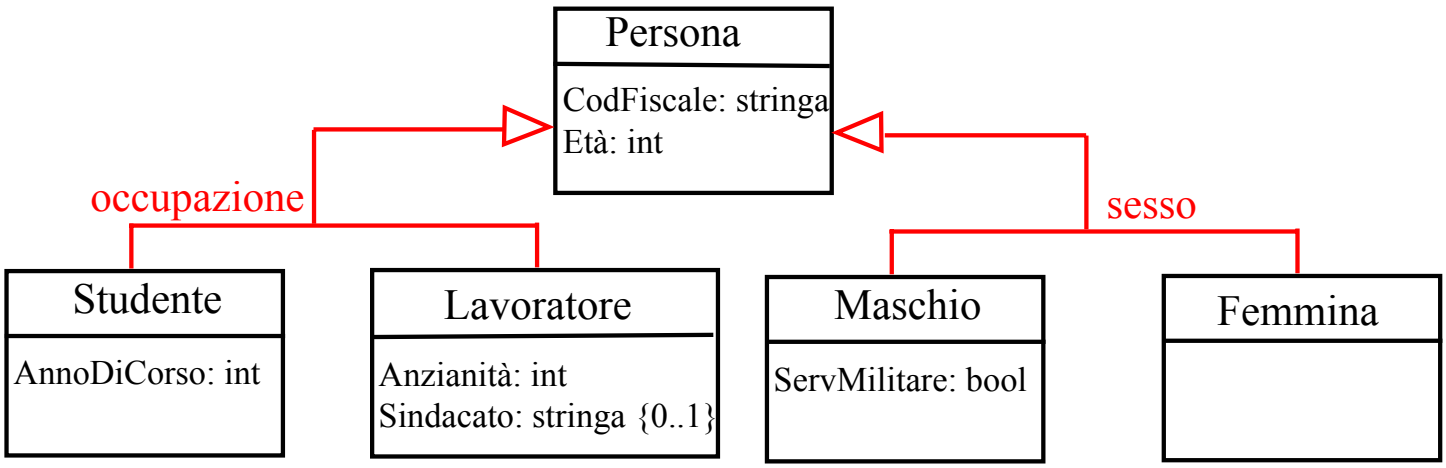
Generalizzazione in UML

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)



Diverse generalizzazioni della stessa classe

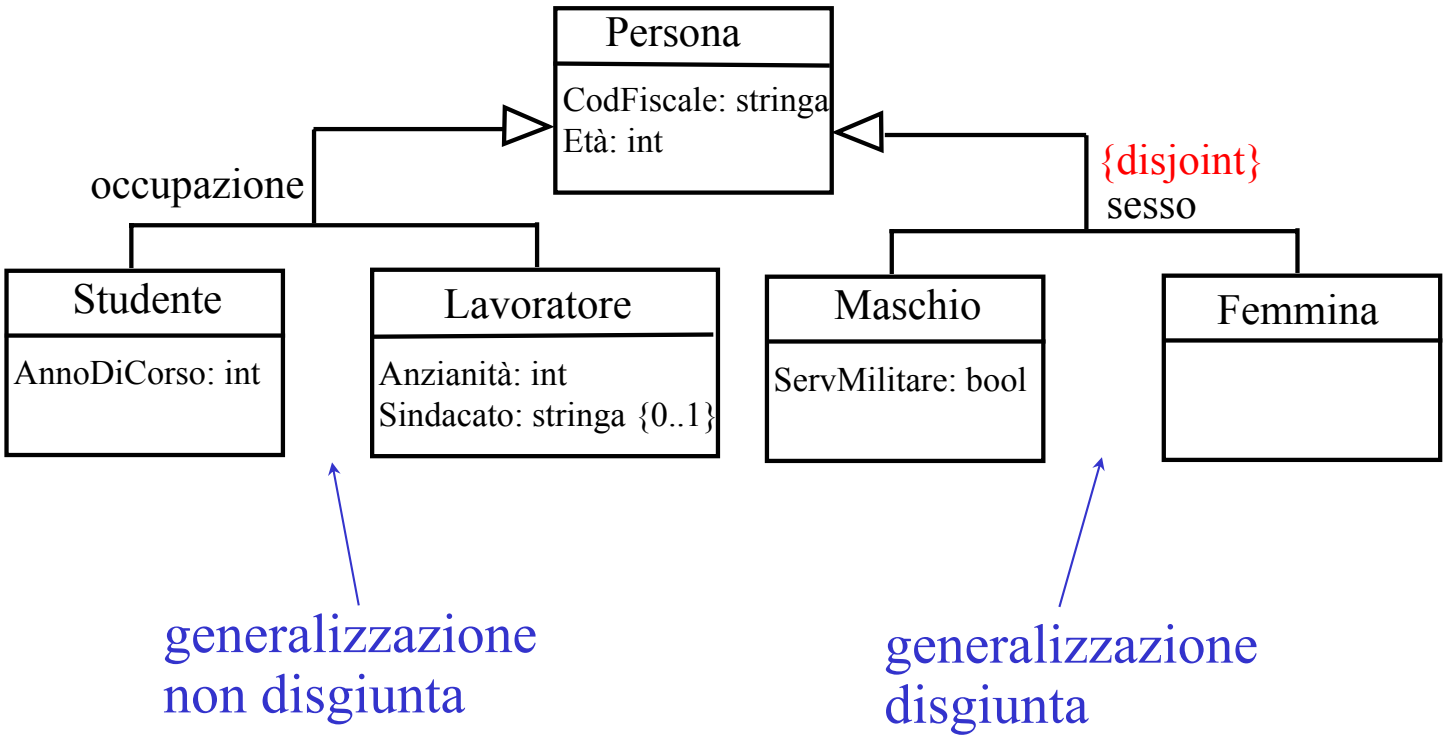
La stessa superclasse può partecipare a diverse generalizzazioni



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

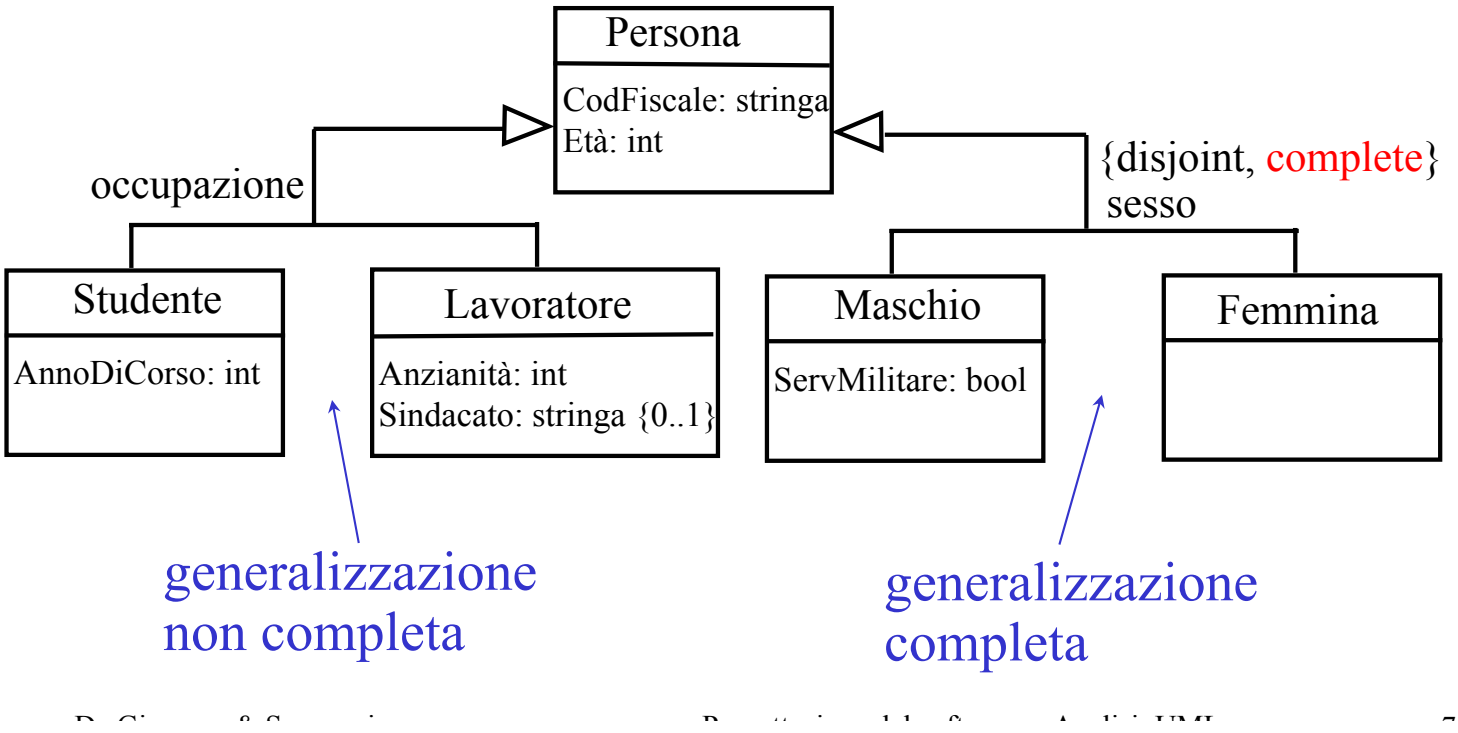
Generalizzazioni disgiunte

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

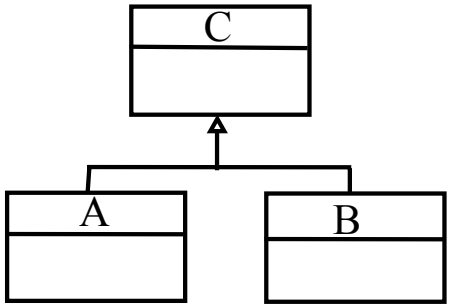
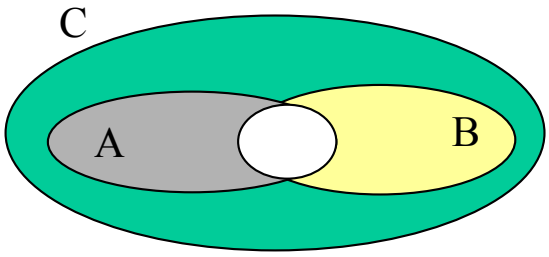
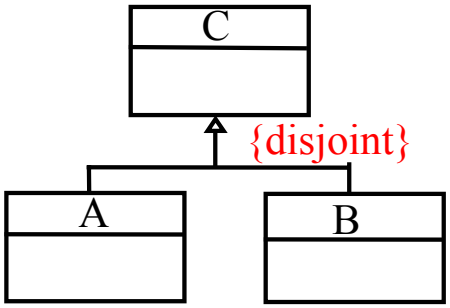
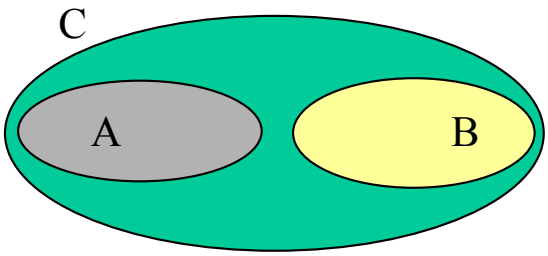


Generalizzazioni complete

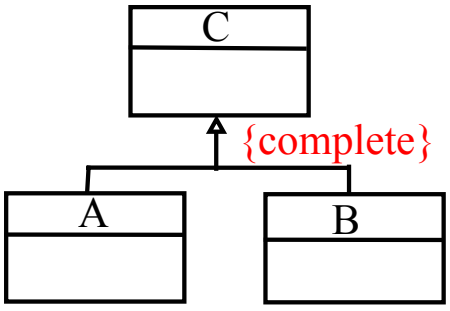
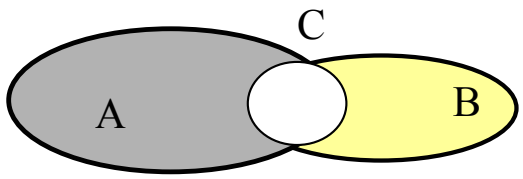
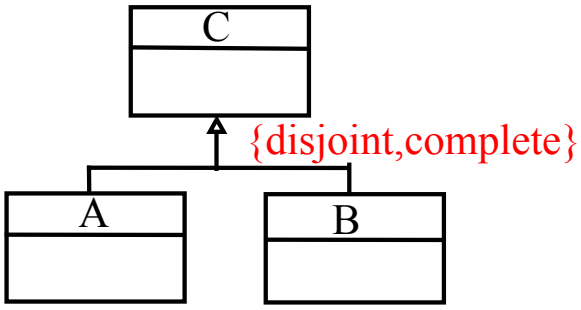
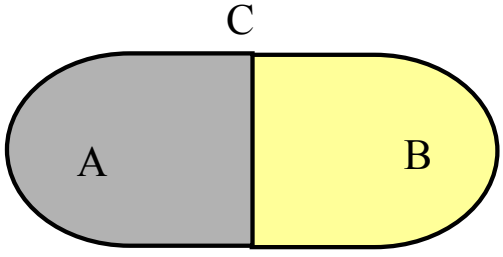
Una generalizzazione può essere completa (l'unione delle istanze delle sottoclassi è uguale all'insieme delle istanze della superclasse) o no



Generalizzazioni

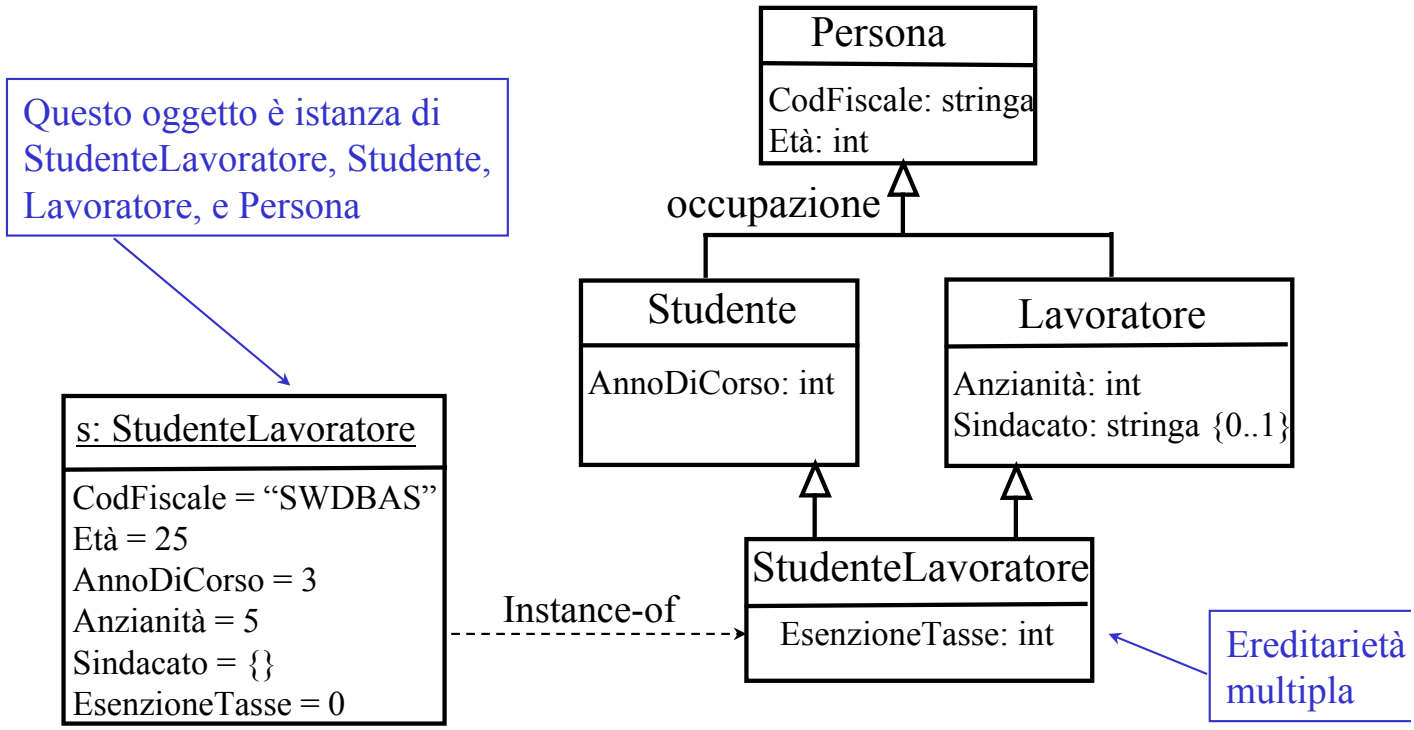
Livello intensionale	Livello estensionale
	
	

Generalizzazioni

Livello intensionale	Livello estensionale
	
	

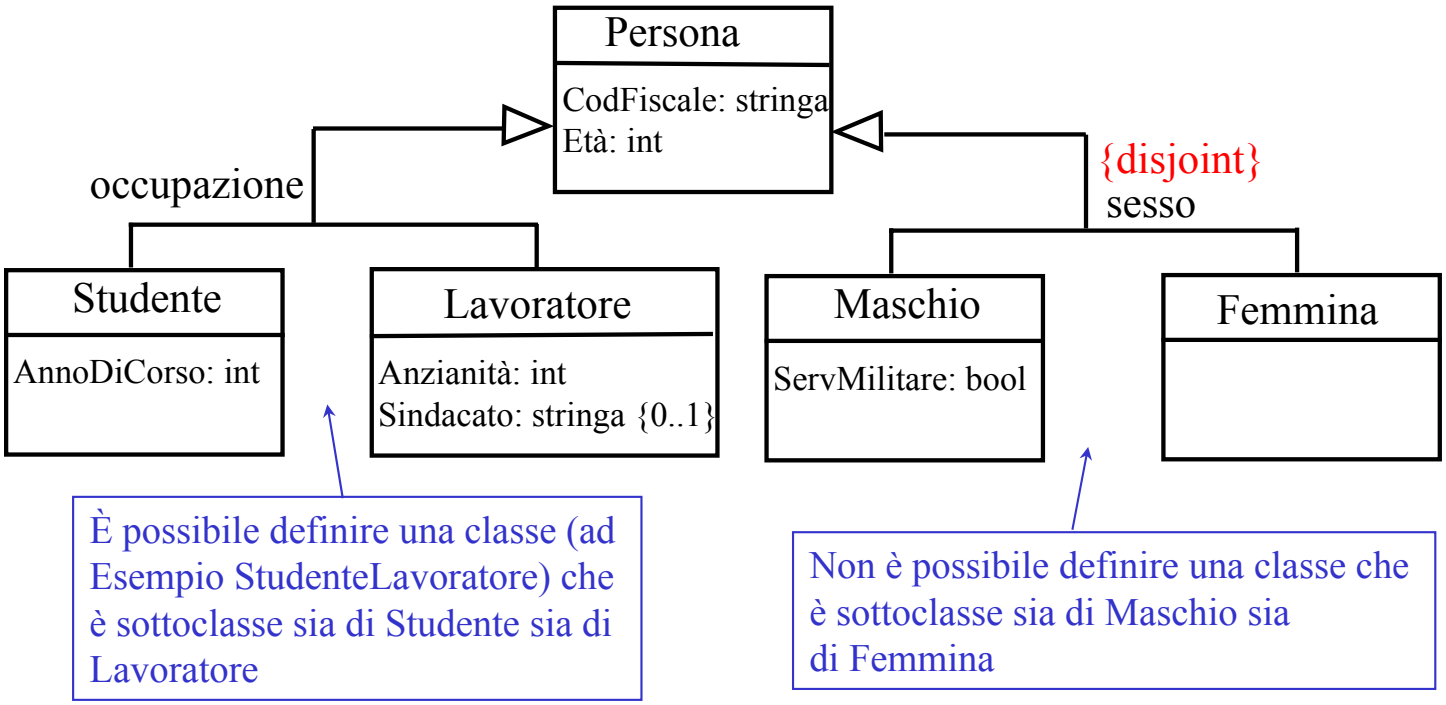
Ereditarietà multipla

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)

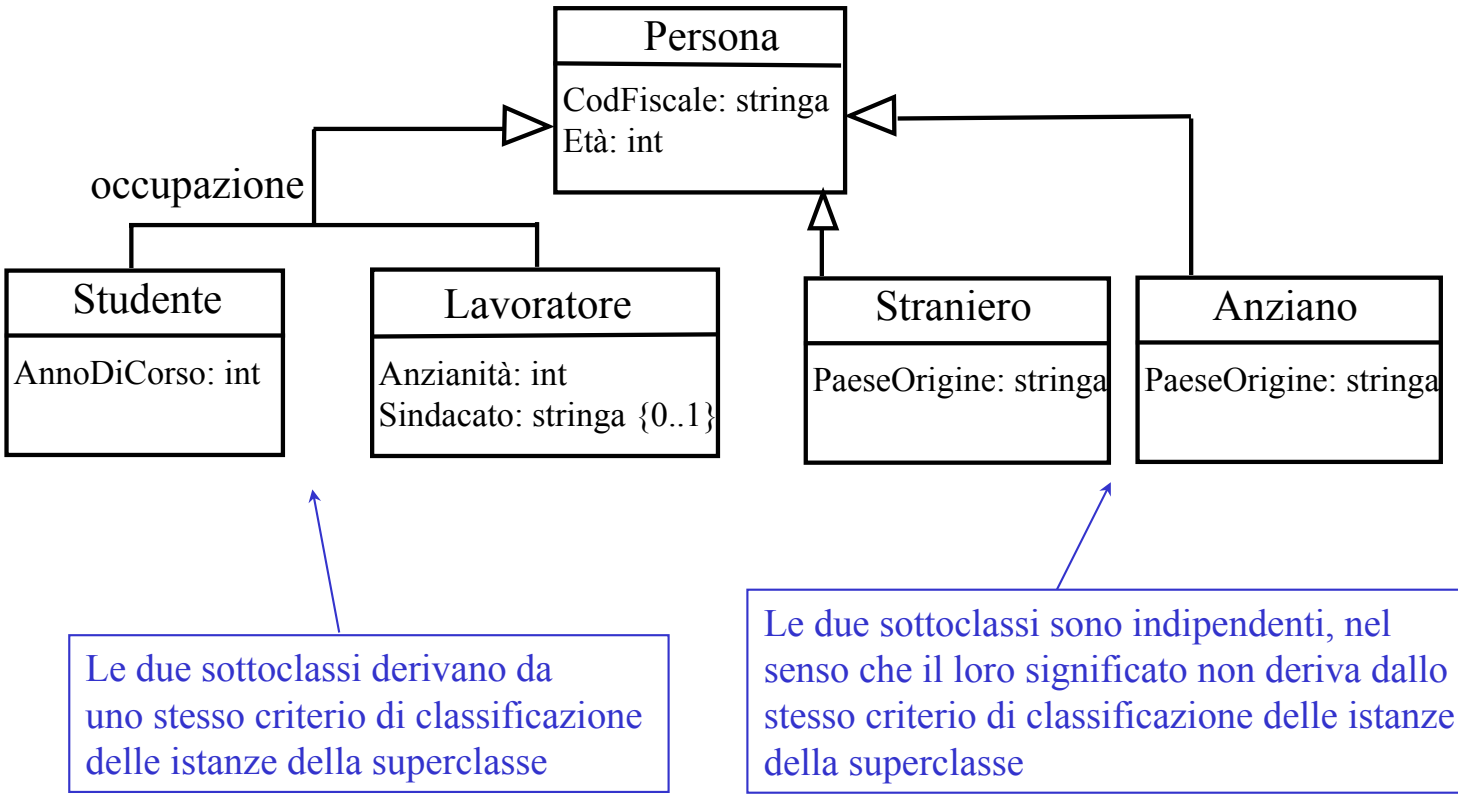


Differenza tra classi disgiunte e non disgiunte

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



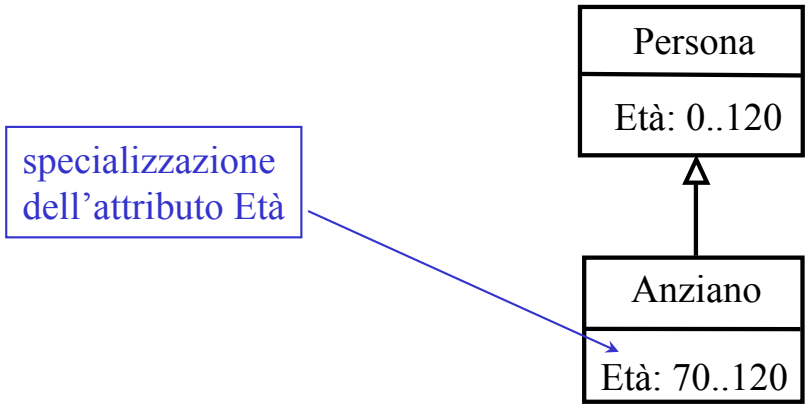
Differenza tra due isa e una generalizzazione



Specializzazione

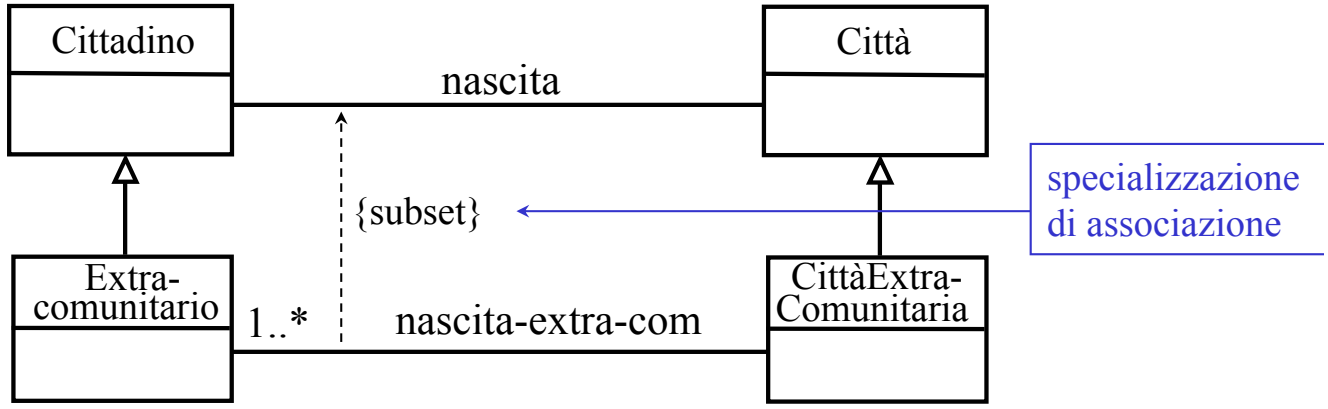
In una generalizzazione la sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche specializzare le proprietà ereditate dalla superclasse.

- 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.



Specializzazione

- 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_2 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_2 a R
  - Definire eventualmente molteplicità più specifiche (molteplicità minima non minore, e molteplicità massima non maggiore) su R_2 rispetto alle corrispondenti molteplicità definite su R



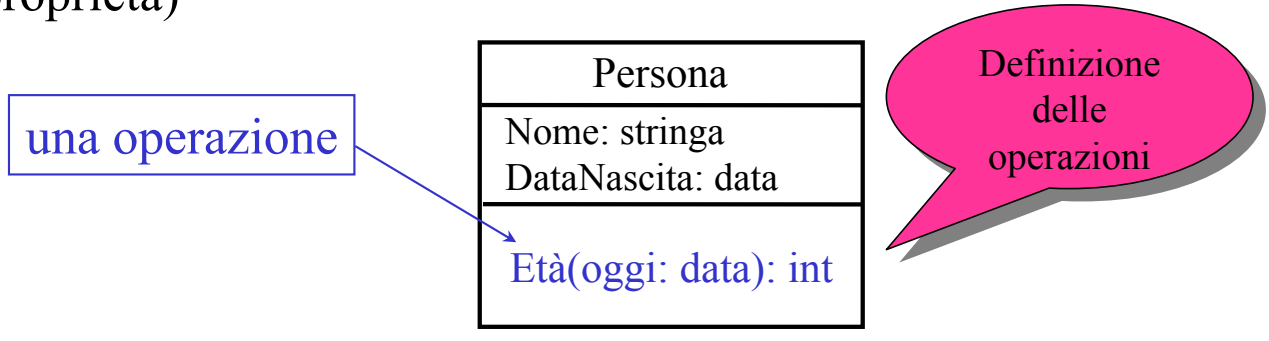
Operazioni

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),

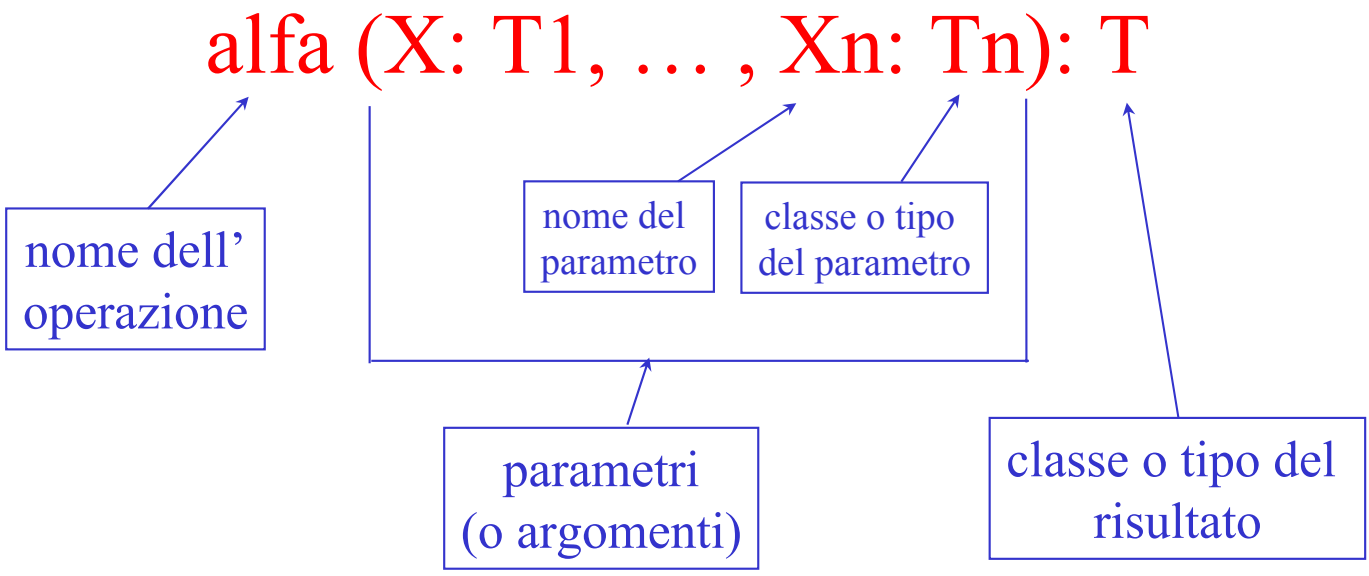
• o per calcolare le proprietà

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



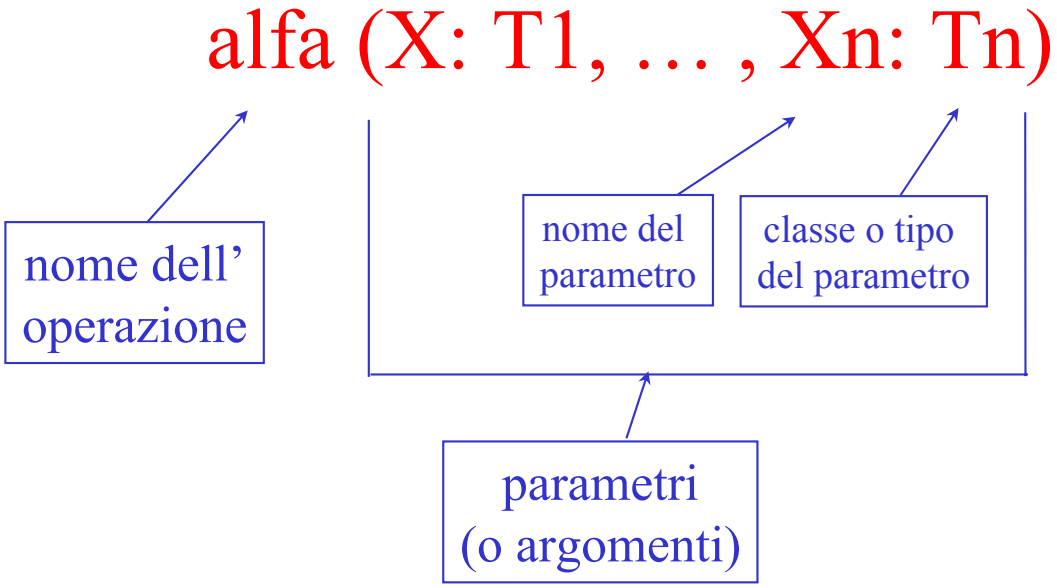
Definizione di una operazione

In una classe, una operazione si definisce specificando i parametri e il tipo del risultato (se c'è), e non il metodo (cioè non la specifica delle istruzioni che realizzano l'operazione)



Definizione di una operazione

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

- 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 in gioco implicitamente un oggetto (l'oggetto di invocazione) della classe in cui l'operazione è definita

- 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 6

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

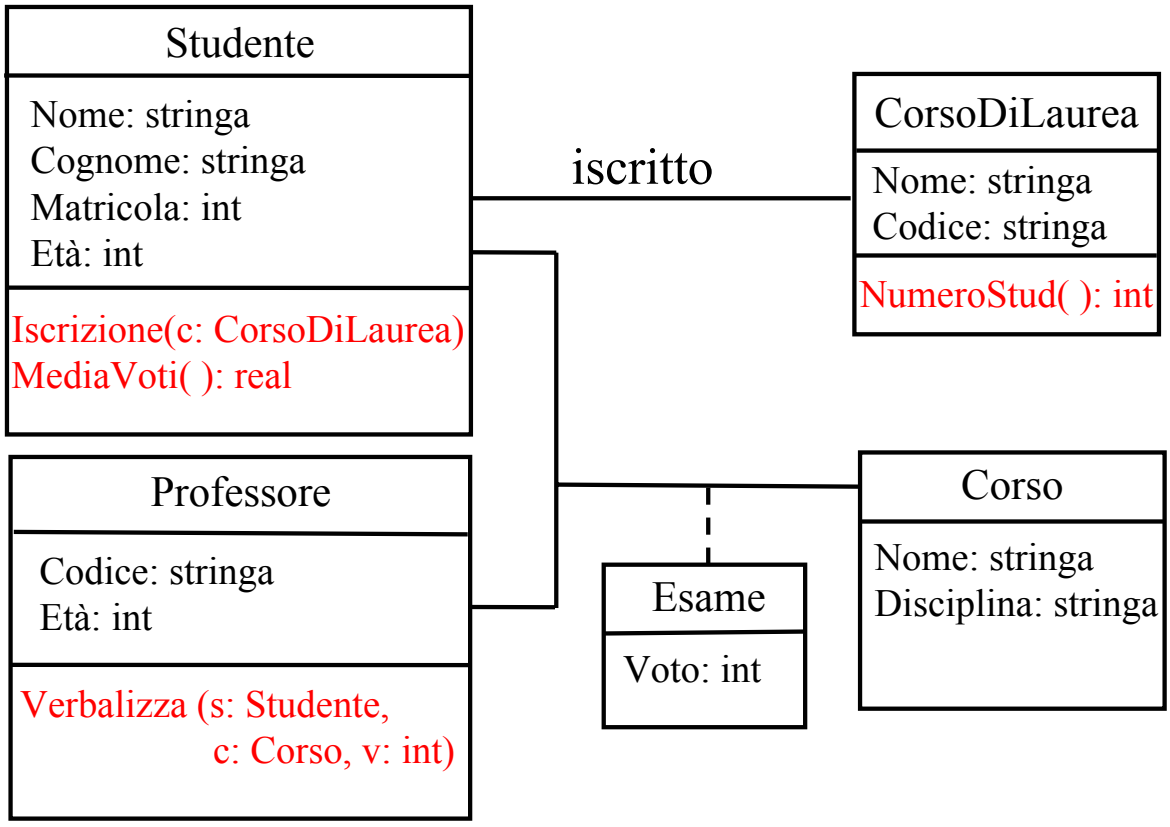
Si vogliono modellare gli studenti (con nome, cognome, numero di matricola, età), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame, con il professore che ha verbalizzato l'esame, ed il voto conseguito. Di ogni corso di laurea interessa il codice e il nome. Di ogni corso 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).

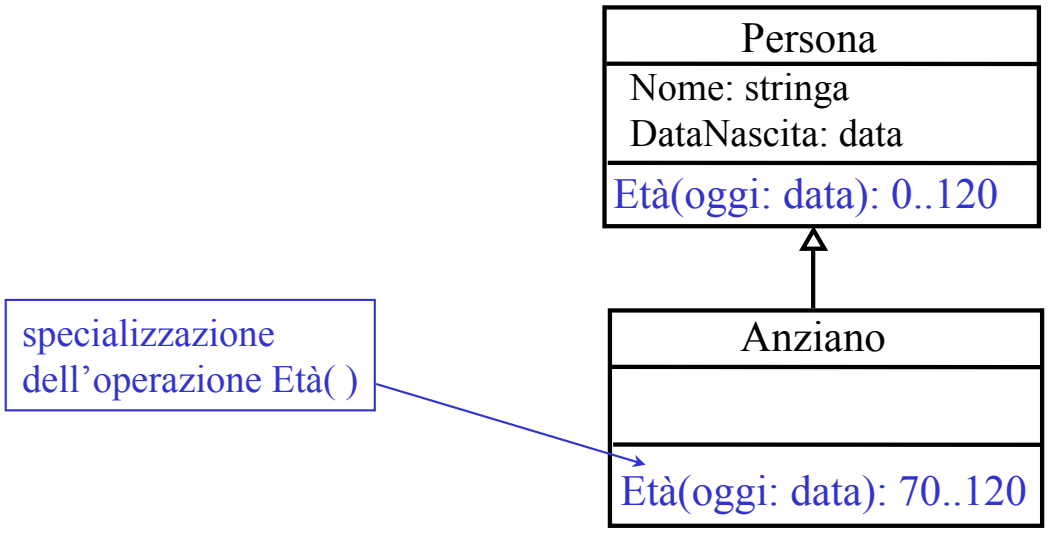
La segreteria vuole periodicamente calcolare la media dei voti di uno studente, e il numero di studenti di un corso di laurea.

Esercizio 6: soluzione



Specializzazione di operazioni

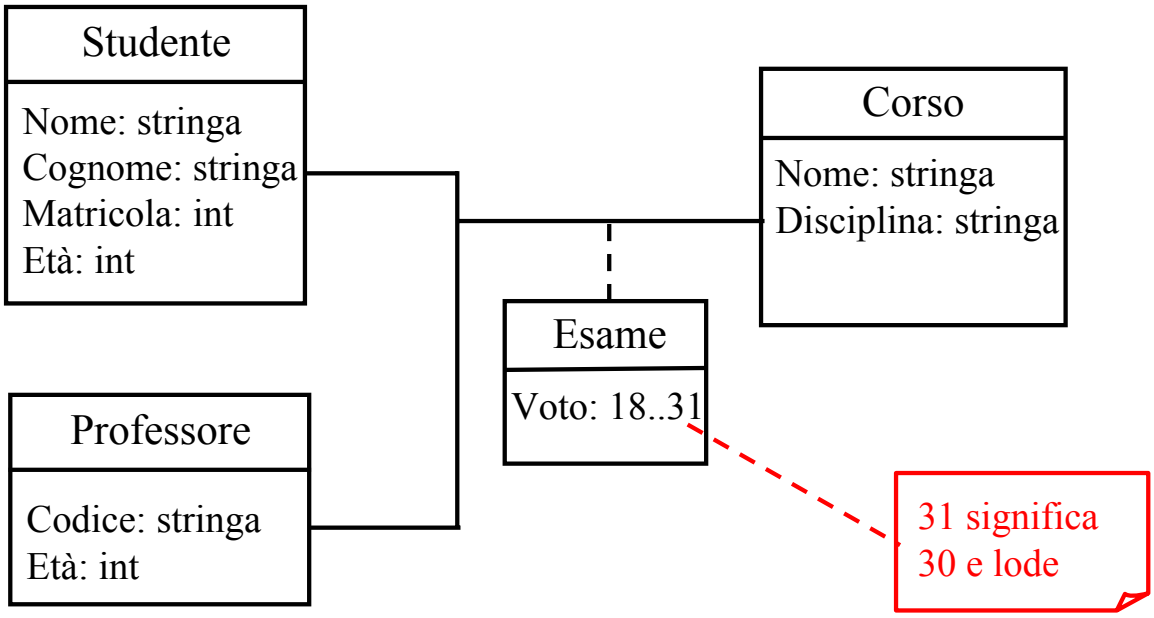
Oltre agli attributi e alle associazioni, anche le operazioni si possono specializzare nelle sottoclassi. Una operazione si specializza specializzando i parametri e/o il tipo di ritorno.



Si noti che il metodo associato ad una operazione specializzata in una sottoclasse è in genere diverso dal metodo associato alla stessa operazione nella superclasse

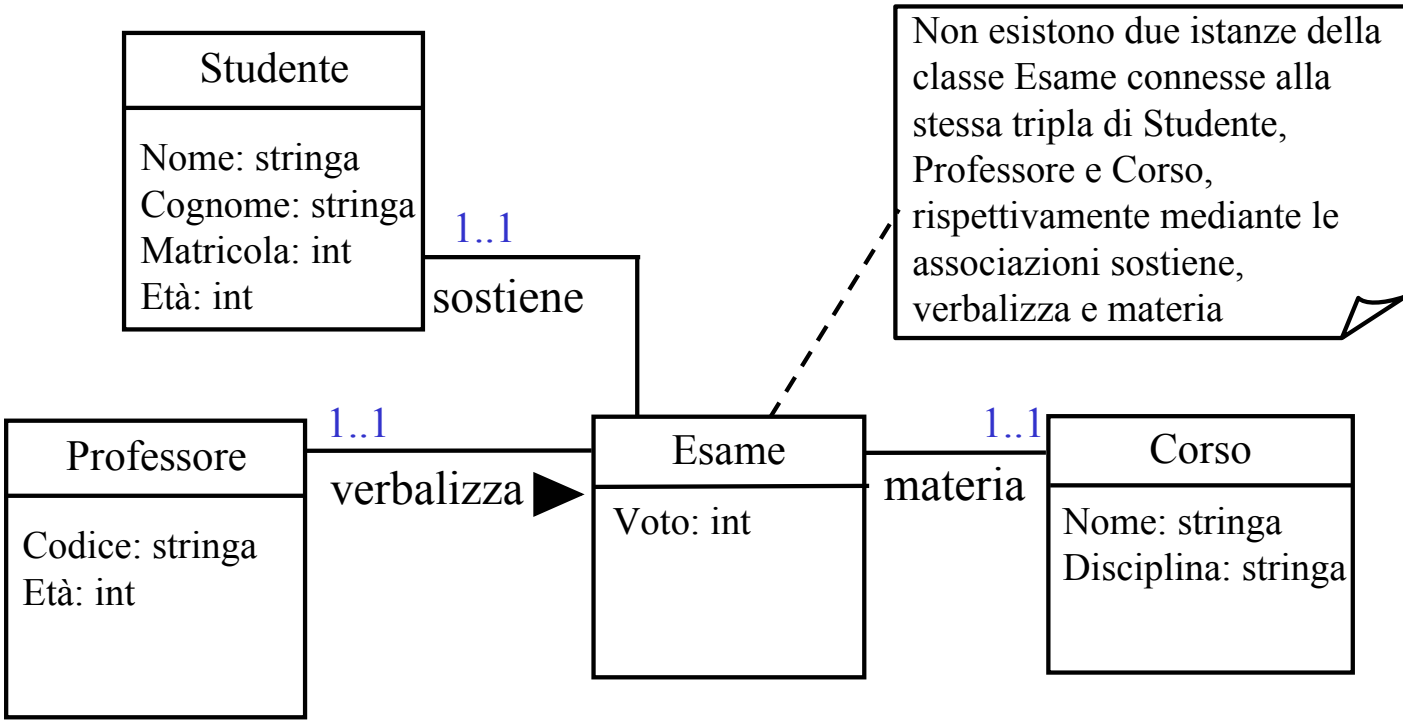
Commenti in UML

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



Osservazione sui tipi

- 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, ad esempio tipi definibili attraverso costruttori come Record, Insieme, Liste, Vettore, ecc.
- Ad esempio, si può usare il tipo indirizzo come record con campi "strada" (di tipo stringa) e "numero civico" (di tipo int), oppure il tipo data come record con campi giorno (di tipo 1..31), mese (di tipo 1..12) e anno (di tipo int).

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

- *Individua le classi e gli oggetti di interesse*
- *Individua gli attributi delle classi*
- *Individua le associazioni tra classi*
- *Individua gli attributi delle associazioni*
- *Determina le molteplicità di associazioni e attributi*
- *Individua le generalizzazioni, partendo o dalla classe più generale e scendendo nella gerarchia, oppure dalle classi più specifiche e risalendo nella gerarchia*
- *Determina le specializzazioni*
- *Individua le operazioni ed associale alle classi*
- *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?*
- *Verificare che le generalizzazioni non formino cicli*
- *Le specializzazioni sono corrette?*
- *Si possono applicare ulteriori generalizzazioni?*
- *Ci sono classi che sono sottoinsiemi di classi disgiunte?*

Scelta tra attributi e classi

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

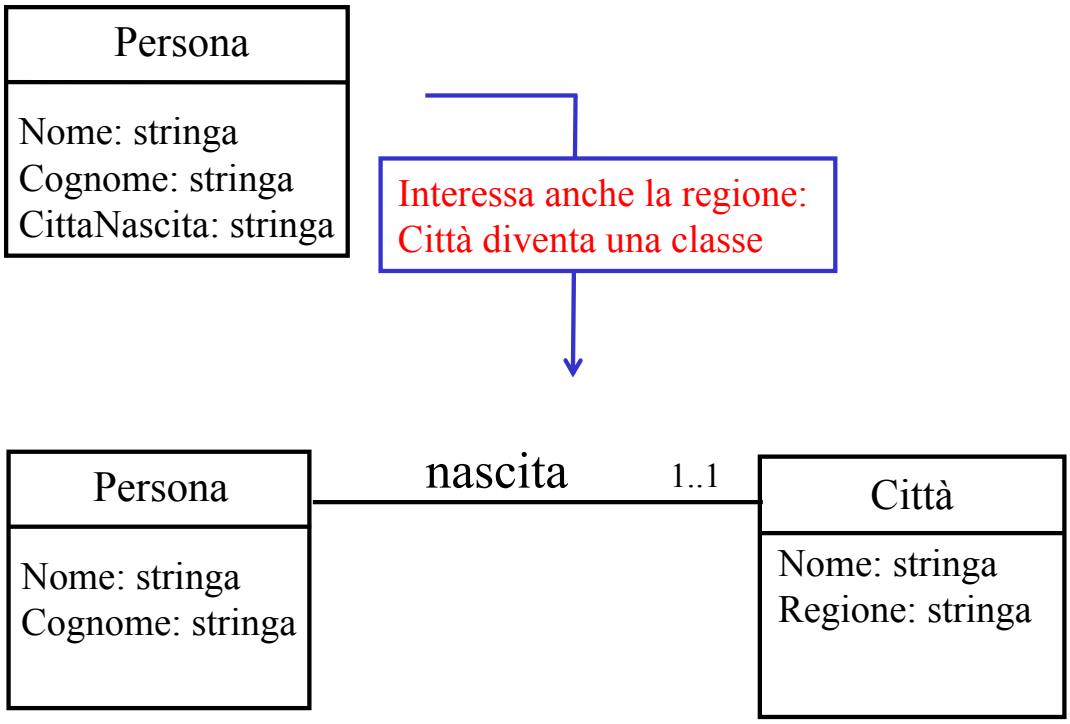
	Classe	Tipo
Istanze	**oggetti**	**valore**
Istanze identificate da	**identificatore di oggetto**	**valore**
Uguaglianza	**basata su identificatore**	**basata su valore**
Realizzazione	**da progettare**	**tipicamente predefinita, oppure basata su strutture di dati predefinite**

Scelta tra attributi e classi

- 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
  - un attributo
    - Se le sue istanze non hanno vita propria
    - Se ha senso solo per rappresentare proprietà di altri concetti

Scelta tra attributi e classi

Le scelte possono cambiare durante l'analisi. Esempio:

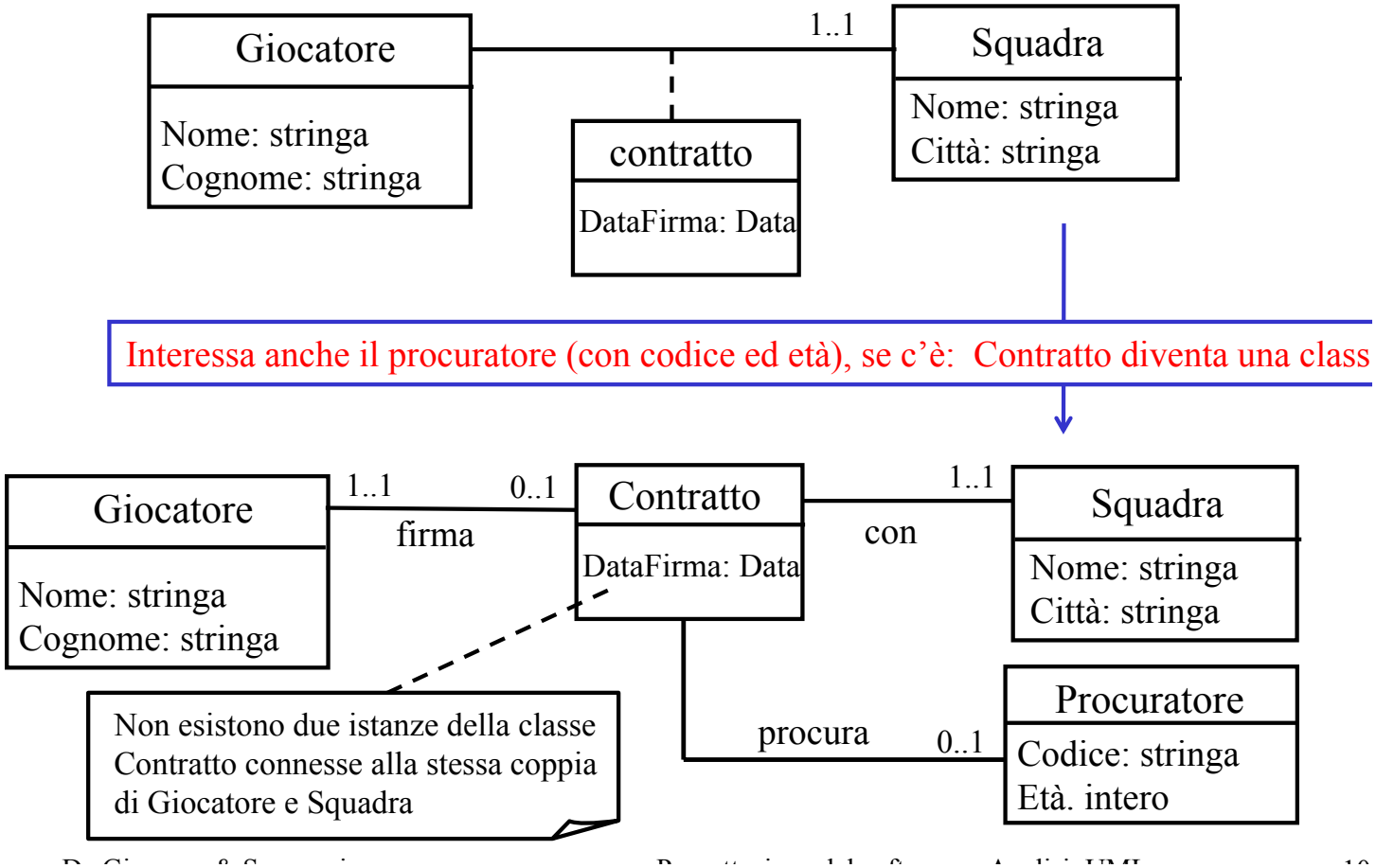


Scelta tra classi e associazione

- 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 instanze ad altre associazioni

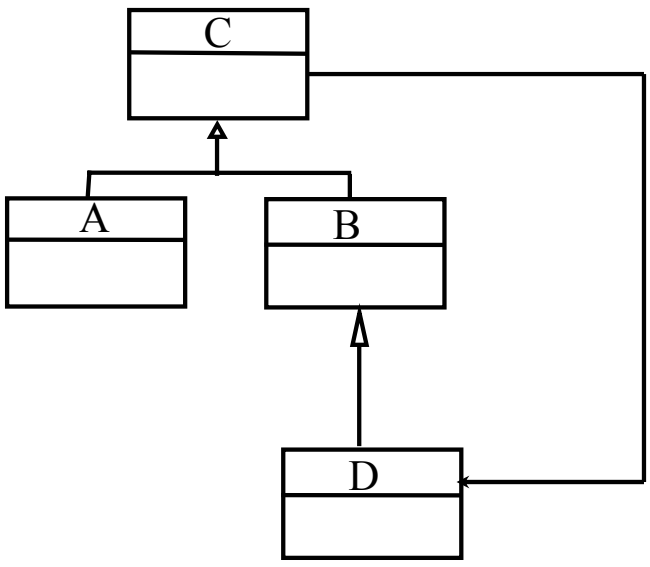
Scelta tra classi e associazioni

Le scelte possono cambiare durante l'analisi. Esempio:



Verifiche sulle generalizzazioni

- Il grafo delle generalizzazioni non può contenere cicli!



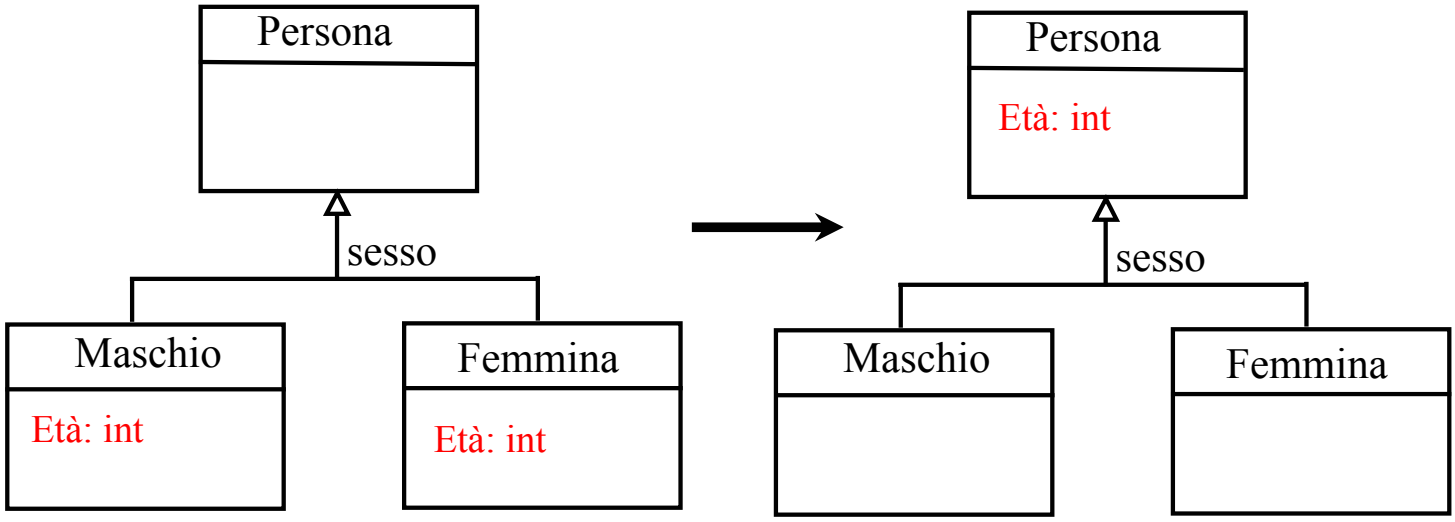
Ciclo nel grafo delle generalizzazioni:
Le classi C, B e D hanno le stesse istanze!

Verifiche sulle specializzazioni

- Specializzazione di un attributo: Se una classe C_1 ha un attributo A di tipo T_1, e 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.
- 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 devono essere definite molteplicità uguali o più ristrette che per R
  - C_4 è una sottoclasse di C_3 (al limite C_3 e C_4 sono uguali)

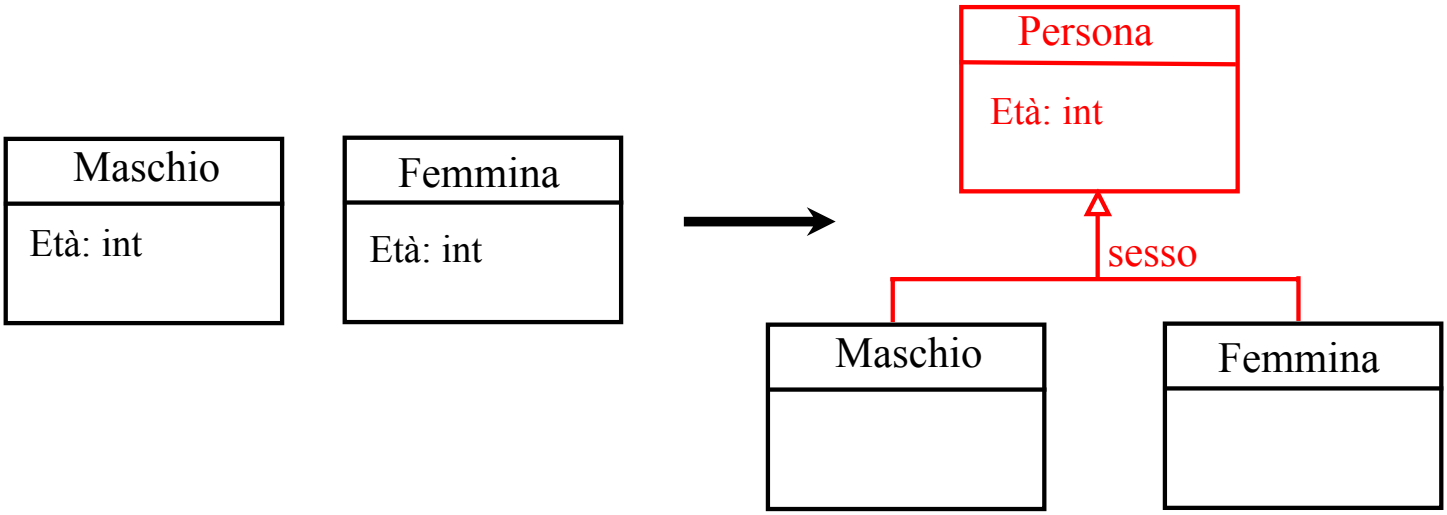
Si possono applicare ulteriori generalizzazioni?

È bene verificare se gli attributi sono stati associati alle classi giuste in una generalizzazione



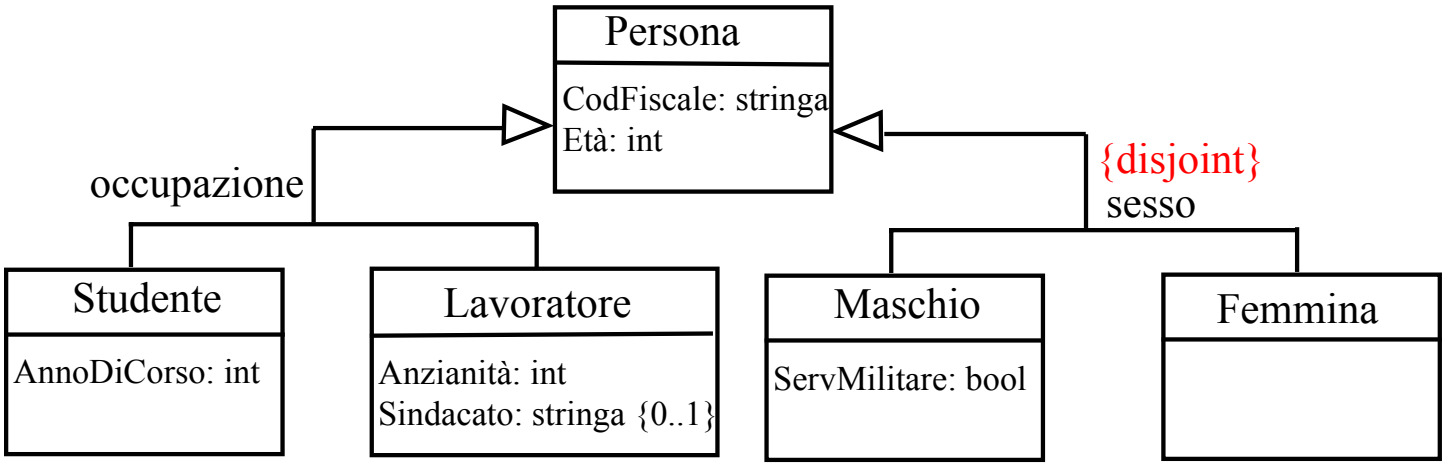
Si possono applicare ulteriori generalizzazioni?

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



Il problema delle classi disgiunte

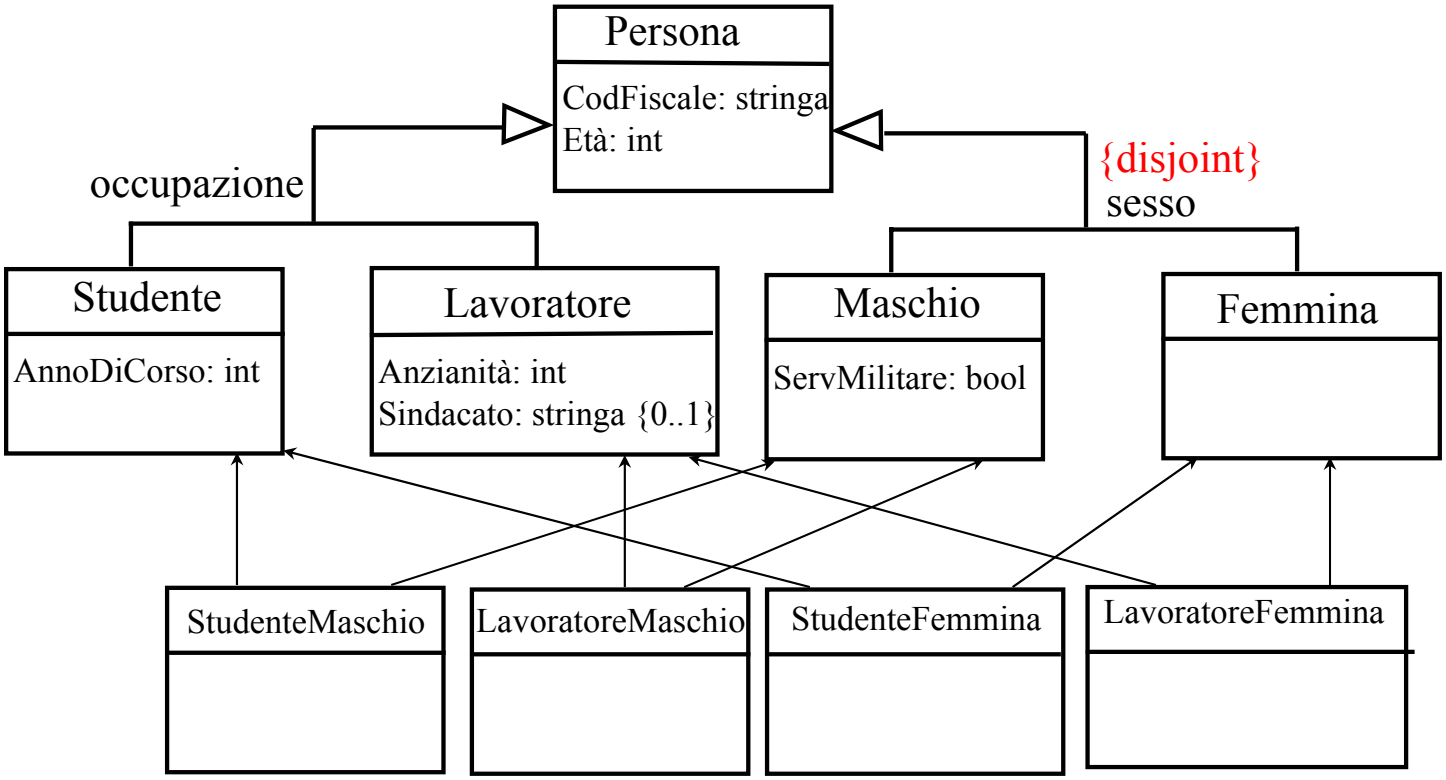
Talvolta è necessario ristrutturare la gerarchia di generalizzazione



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

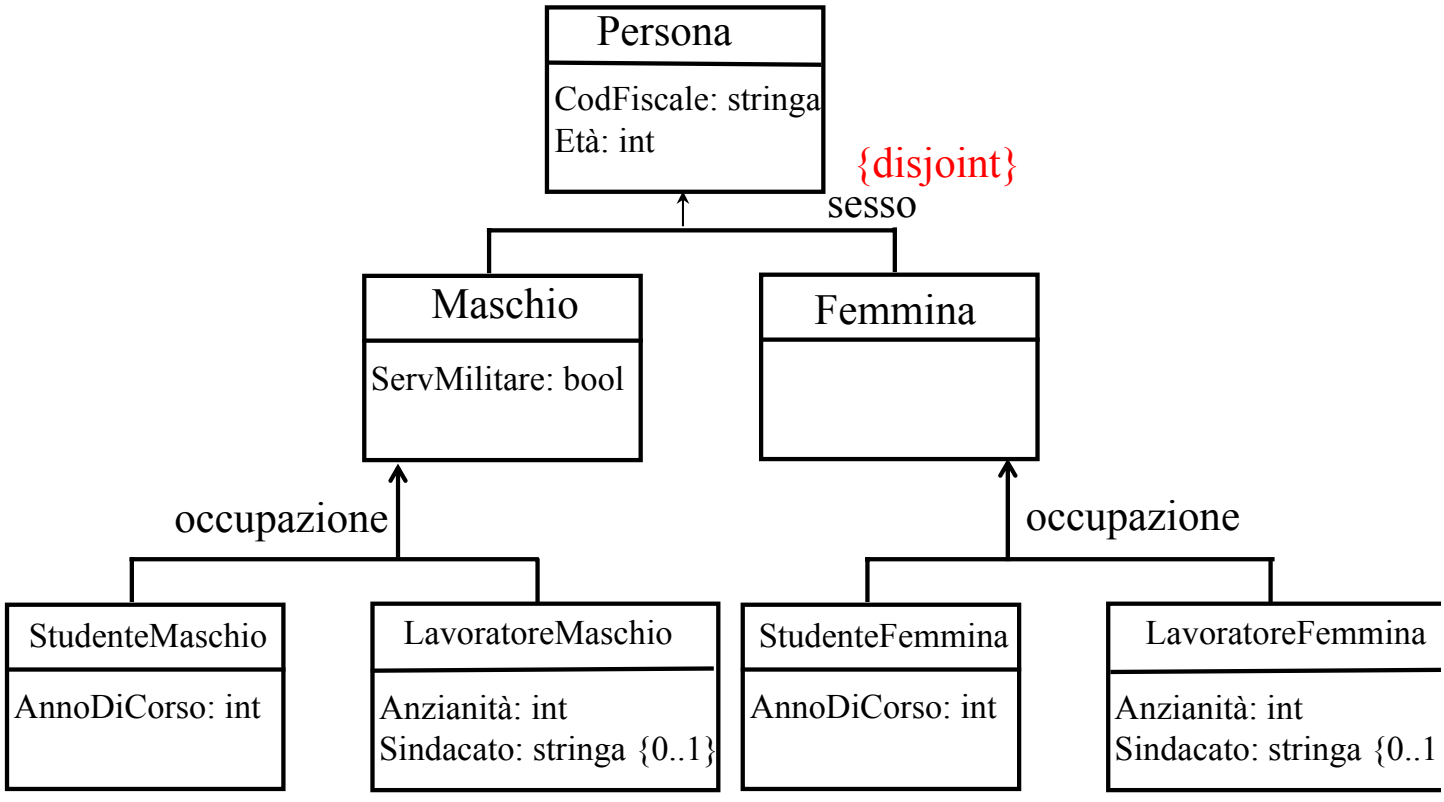
Il problema delle classi disgiunte

Lo schema si può ristrutturare in due modi. Primo modo:



Il problema delle classi disgiunte

Lo schema si può ristrutturare in due modi. Secondo modo:



Il diagramma degli use case

Ricordiamo che lo schema concettuale è costituito da:

- Diagramma delle classi e degli oggetti
- Diagramma degli use case
- Diagramma degli stati e delle transizioni

Il diagramma degli use-case descrive le funzionalità fondamentali che il sistema deve realizzare, in termini di scenari di utilizzo del sistema

Use case

Uno use case rappresenta una tipica interazione tra un utente ed il sistema software da realizzare. Un Use case cattura una qualche funzione visibile dall'utente, e la sua descrizione si ottiene attraverso l'interazione tra analista ed utente in fase di analisi.

In altre parole, uno use case definisce un particolare modo di utilizzare il sistema, il quale offre servizi e funzionalità in risposta a eventi prodotti da attori esterni.

Use case

Uno use case modella un processo (o un insieme di processi) che è trasversale rispetto alle classi, cioè coinvolge più classi allo stesso livello, e sarebbe una forzatura modellarlo come una operazione di una singola classe

Uno use case è in genere composto da diverse operazioni, che non vengono definite in modo dettagliato nel diagramma. Vedremo, quando parleremo di "specifica di uno use case", come queste operazioni vengono definite

Use Case: Attori

Uno use case è formulato sulla base delle funzionalità offerte dal sistema così come sono percepite dagli utenti.

Oltre agli use case, un altro componente fondamentale del diagramma degli use case è l'attore. Un attore è un ruolo che un utente (una persona o un sistema esterno) gioca interagendo con il sistema.

Lo stesso utente può essere rappresentato da più di un attore (può giocare più ruoli). Più utenti possono essere rappresentati dallo stesso attore.

Use case: diagramma

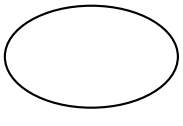
Un diagramma degli use case è un grafo i cui nodi possono essere

- Attori
- Use Case

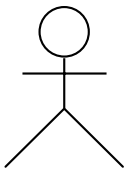
mentre gli archi rappresentano

- la comunicazione tra gli attori e gli use case,
- i legami d'uso tra use case
- l'estensione di uno use case da parte di un altro
- la generalizzazione di uno use case da parte di un altro

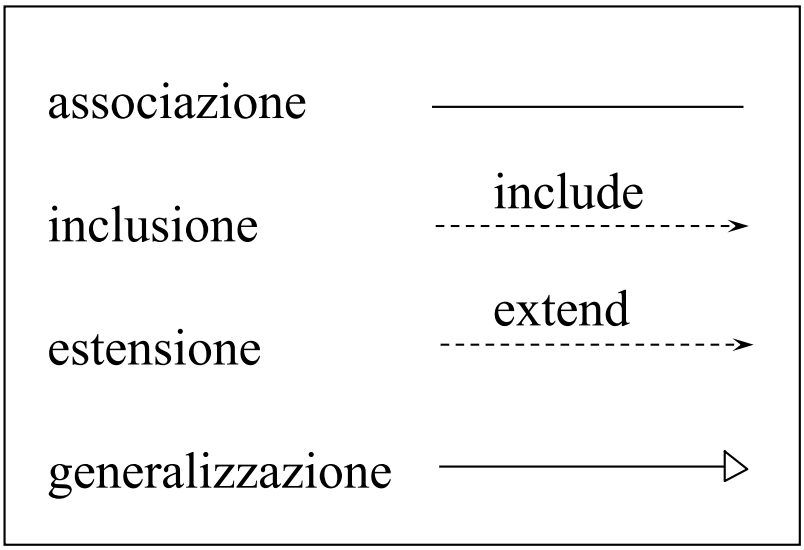
Componenti di un diagramma degli use case



Use Case



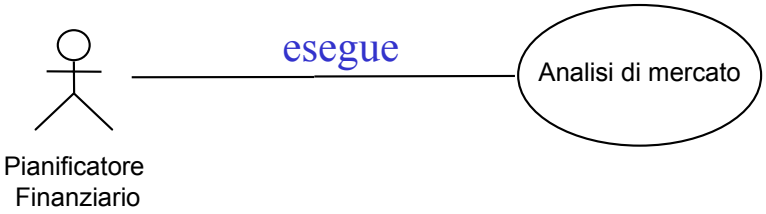
Attore



Relazioni nel diagramma degli use case

Associazione

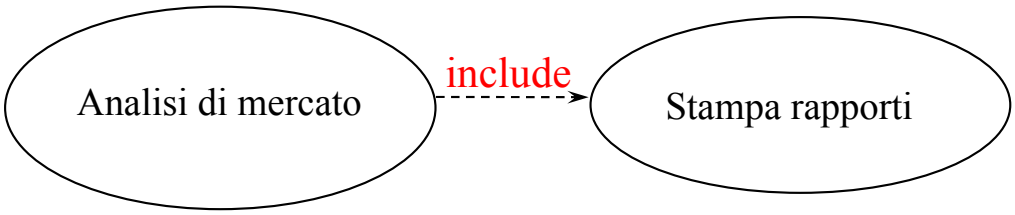
La partecipazione di un attore ad uno use case è rappresentata da un arco di associazione (con un nome) tra il simbolo dell'attore e il simbolo di use case. In generale, questo significa che l'attore "comunica" con lo use case.



Esempio di diagramma degli use case

Inclusione

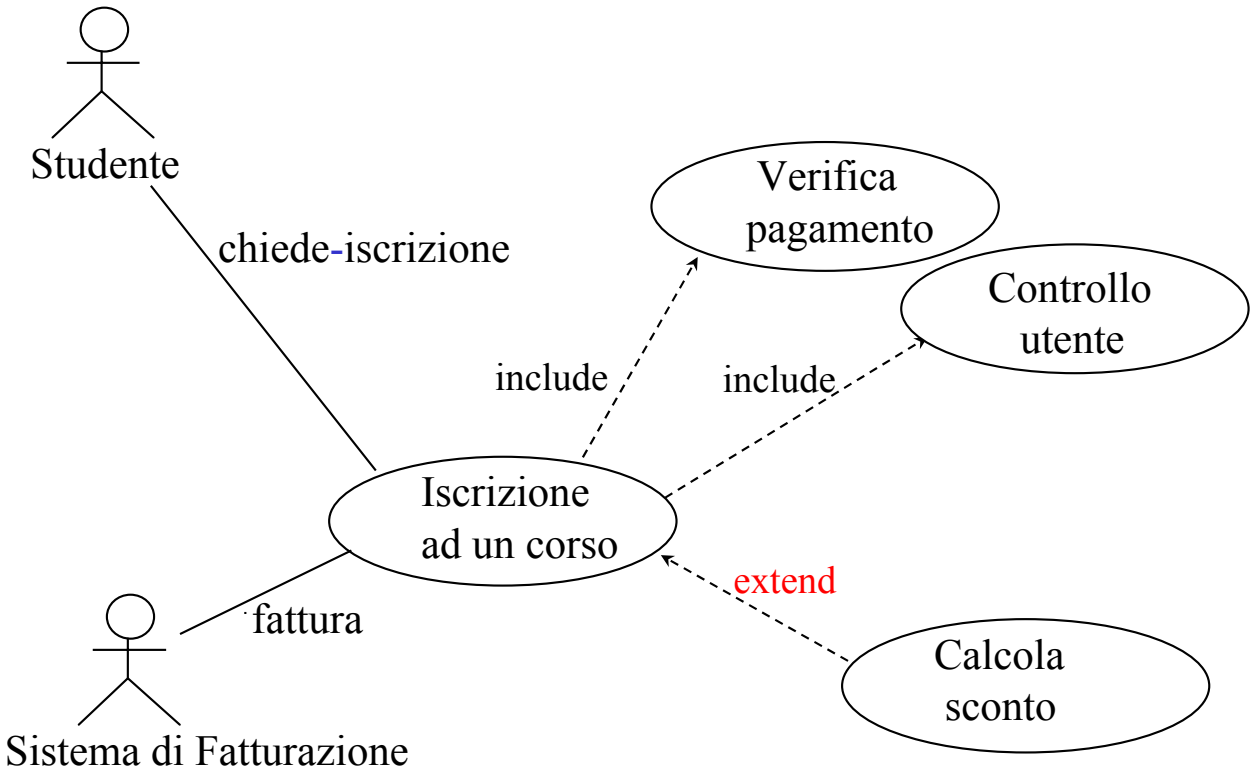
Una relazione d'inclusione tra use case è rappresentata da una freccia tra lo use case che include e quello che è incluso. La freccia è etichettata con "include". Una relazione d'inclusione da uno use case *A* ad uno use case *B*, indica che ogni istanza dello use case *A* includerà anche il comportamento specificato per lo use case *B*. In altre parole, qualche funzionalità di *A* richiede di servirsi di qualche funzionalità di *B*.



Estensione

La relazione "estende" tra use case è rappresentata da una freccia etichettata con "extend" dallo use case che definisce l'estensione allo use case base. La relazione "extend" tra uno use case *A* ed uno use case *B* indica che ogni istanza di *B*, in condizioni particolari, viene esteso con le funzionalità di una istanza di *A*. Per uno stesso use case, i comportamenti definiti attraverso diverse estensioni possono occorrere all'interno di ogni singola sua istanza.

Esempi di extend e include



Generalizzazione

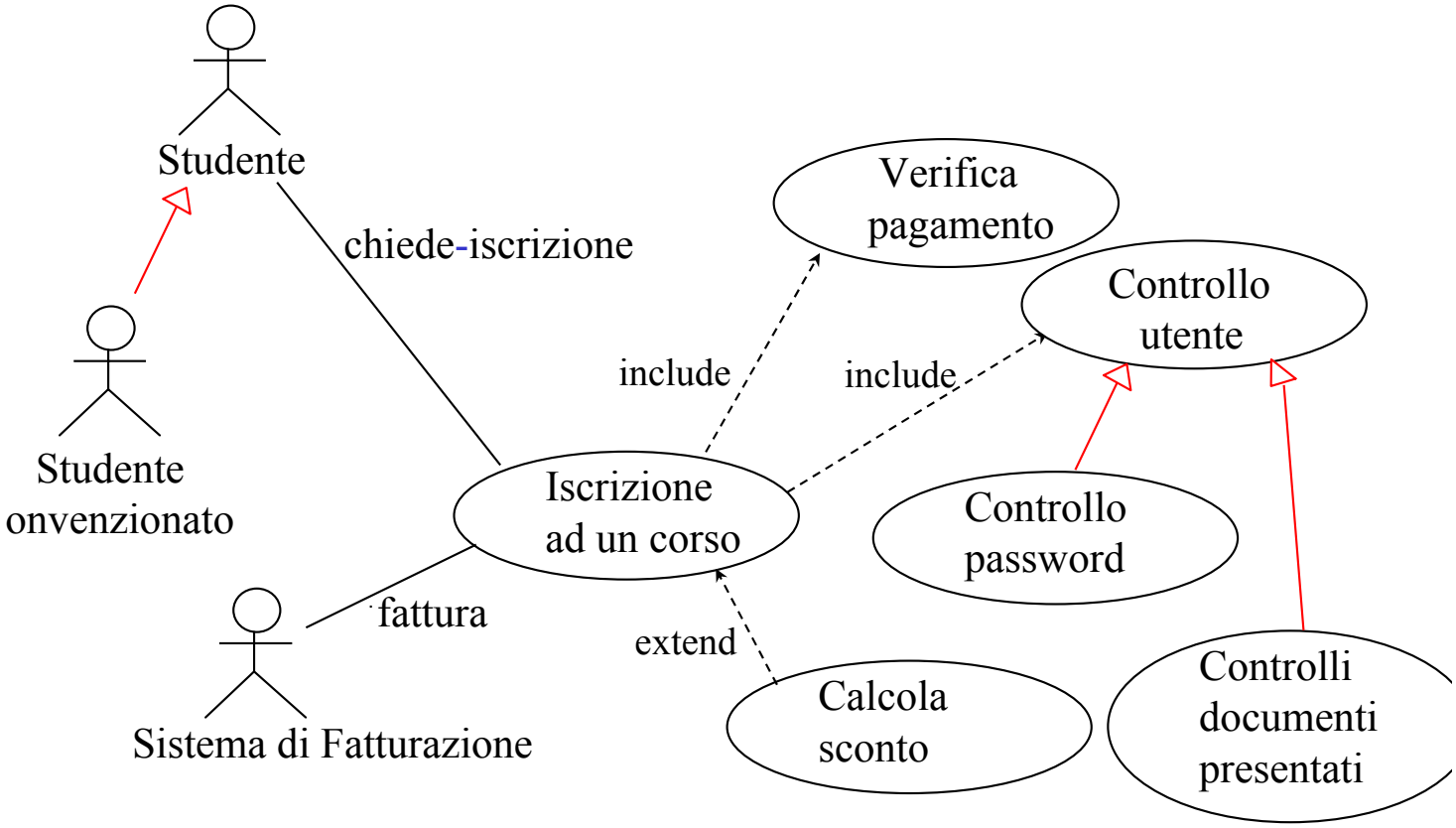
La generalizzazione si applica sia ad attori sia a use case.

Un attore *A* è generalizzazione di un attore *B* quando *B* è visto come un caso particolare di *A*.

Analogamente, uno use case *A* è generalizzazione di uno use case *B* quando *B* è visto come un caso particolare di *A*.

Il concetto di generalizzazione è simile a quello usato nel diagramma delle classi.

Esempi di extend, include e generalizzazione



Aspetti metodologici nella costruzione del diagramma degli use case

Un metodo comunemente usato per costruire il diagramma degli Use Case prevede i seguenti passi

- *Individua gli use case generali di interesse*
- *Individua gli attori*
- *Individua le associazioni tra attori e use case*
- *Individua altri use case, più specifici, se ce ne sono*
- *Determina le relazioni "include" tra use case*
- *Individua le generalizzazioni tra attori e tra use case,*
- *Individua le relazioni "extend" tra use case*
- *Controllo di qualità*

Correggi, modifica, estendi

Controllo di qualità del diagramma degli use case

- *Sono stati colti tutti gli aspetti insiti nei requisiti?*
- *Sono state individuate tutte le associazioni e tutte le generalizzazioni?*
- *Ci sono ridondanze nel diagramma?*
- *È alta la coesione dei singoli use case?*
- *È basso l'accoppiamento tra diversi use case?*

Esercizio 7

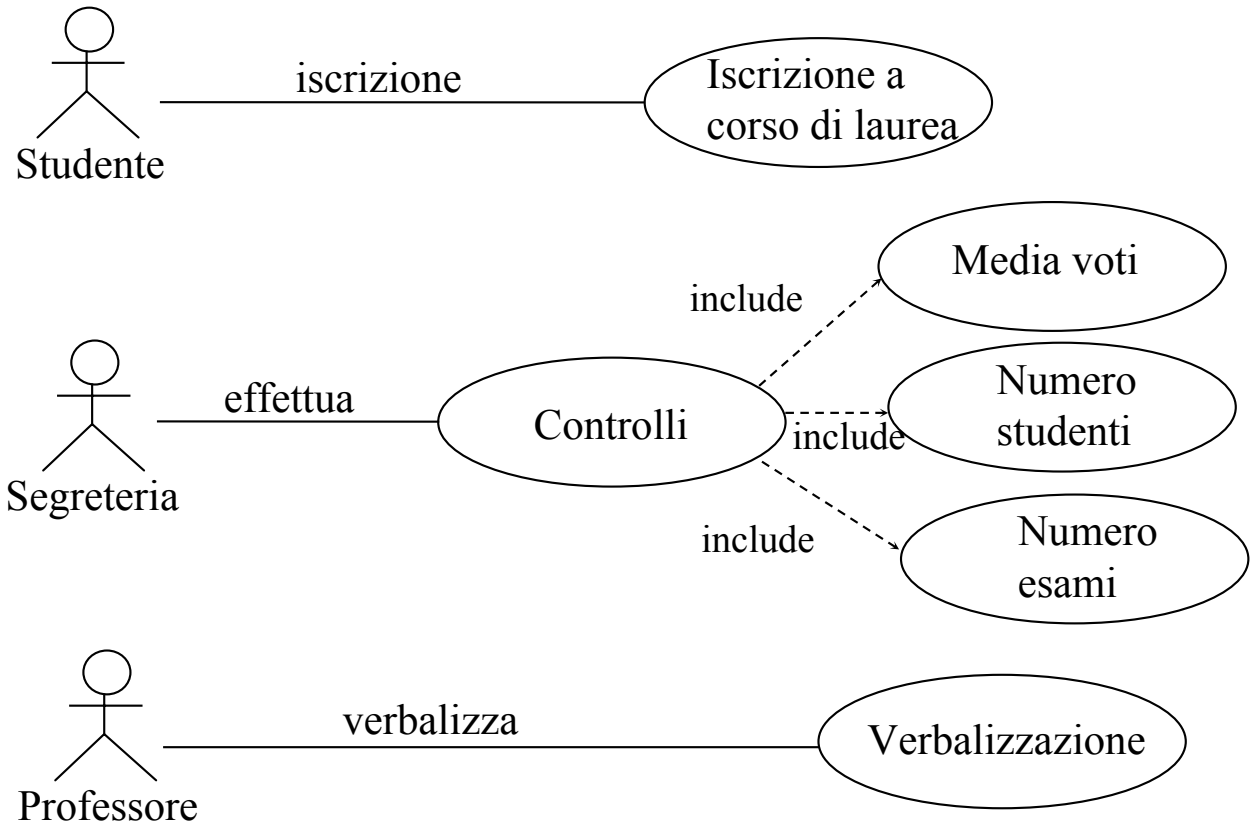
Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

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, il numero di studenti di un corso di laurea, e la media del numero di esami sostenuti per gli studenti di un corso di laurea.

Esercizio 7: soluzione



Il diagramma degli stati e delle transizioni

Il diagramma degli stati e delle transizioni viene definito per una classe, ed intende descrivere l'evoluzione tipica di un oggetto generico di quella classe.

Uno stato rappresenta una situazione in cui un oggetto ha un insieme di proprietà considerate stabili

Una transizione modella un cambiamento di stato ed è denotata da:

Evento [Condizione] / Azione

Il diagramma degli stati e delle transizioni

Il significato di una transizione del tipo di quella qui mostrata è:

- Se l'oggetto si trova nello stato S_1, e
- Se si verifica l'evento E, e
- Se la condizione C è verificata
- Allora viene eseguita l'azione A e l'oggetto passa nello stato S_2.

Esempio di diagramma degli stati e delle transizioni

Descriviamo il diagramma degli stati e delle transizioni relativa ad una classe "Caldaia". In questo diagramma ogni transizione è caratterizzata solamente da eventi e condizioni (i cambiamenti di stato non hanno bisogno di azioni perché sono automatici)

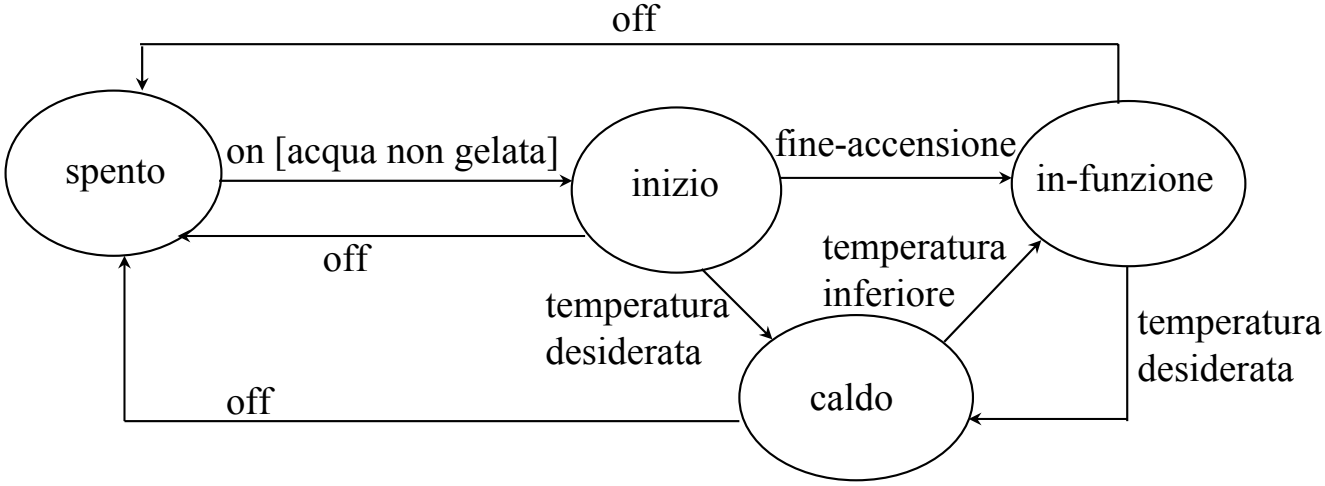
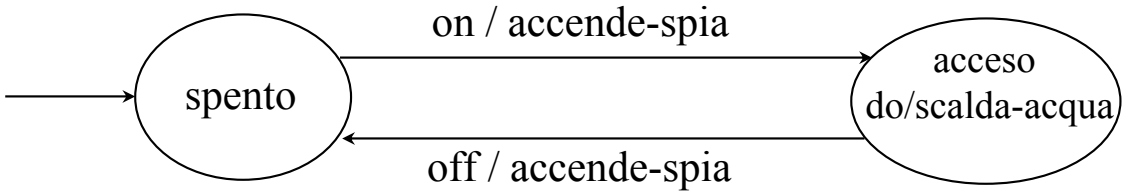


Diagramma degli stati e delle transizioni

- Una freccia non etichettata che parte dal "vuoto" ed entra in uno stato indica che lo stato è inziale
- Una freccia non etichettata che esce da uno stato e finisce nel "vuoto" indica che lo stato è finale
- Alcune volte vogliamo rappresentare dei processi che l'oggetto esegue senza cambiare stato. Questi processi si chiamano attività, e si mostrano negli stati con la notazione:

do / attività

Esempio:



Aspetti metodologici nella costruzione del diagramma degli stati e delle transizioni

Un metodo comunemente usato per costruire il diagramma degli stati e delle transizioni prevede i seguenti passi

- *Individua gli stati di interesse*
- *Individua le transizioni*
- *Individua le attività*
- *Determina gli stati iniziali e finali*
- *Controllo di qualità*

Correggi, modifica, estendi

Controllo di qualità del diagramma degli stati e delle transizioni

- *Sono stati colti tutti gli aspetti insiti nei requisiti?*
- *Ci sono ridondanze nel diagramma?*
- *Ogni stato può essere caratterizzato da proprietà dell'oggetto?*
- *Ogni azione e ogni attività può corrispondere ad una operazione della classe?*
- *Ogni evento e ogni condizione può corrispondere ad un evento o condizione verificabile per l'oggetto?*

La specifica

Lo schema concettuale viene alla fine corredato da

- una specifica per ogni Classe
- una specifica per ogni use case

La specifica di una classe ha lo scopo di definire precisamente il comportamento di ogni operazione della classe

La specifica di uno use case ha lo scopo di definire precisamente il comportamento di ogni operazione di cui lo use case è costituito

Specifica di una classe

La specifica di una classe C ha la seguente forma:

InizioSpecificaClasse C

Specifica della operazione 1

…

Specifica della operazione N

FineSpecifica

Specifica di uno use case

La specifica di uno use case si fornisce facendo la lista delle operazioni (una o più) che costituiscono lo use case stesso, e fornendo poi la specifica di ogni operazione. La specifica di uno use case D ha la seguente forma:

InizioSpecificaUseCase D

Specifica della operazione 1

…

Specifica della operazione N

FineSpecifica

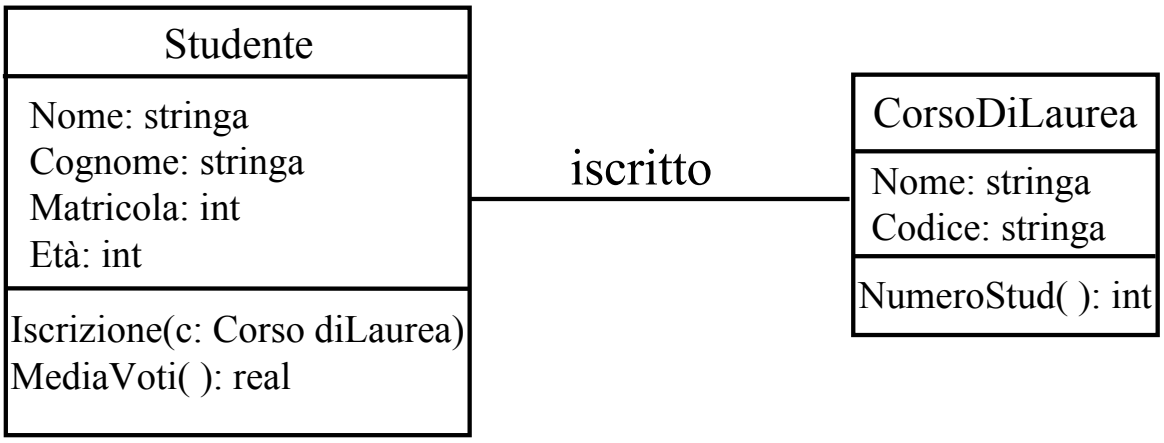
Specifica di una operazione

Che sia una operazione di una classe o una operazione di uno use case, la specifica di una operazione ha la seguente forma:

alfa (X: T1, … , Xn: Tn): T
pre: *condizione*
post: *condizione*

- alfa (X: T1, … , Xn: Tn): T è la segnatura dell'operazione (T può mancare),
- pre rappresenta la precondizione dell'operazione, cioè l'insieme delle condizioni (a parte quelle già stabilite dalla segnatura) che devono valere prima di ogni esecuzione della operazione
- post rappresenta le postcondizioni della operazione, cioè l'insieme delle condizioni che devono valere alla fine di ogni esecuzione della operazione

Esempio di specifica di una operazione

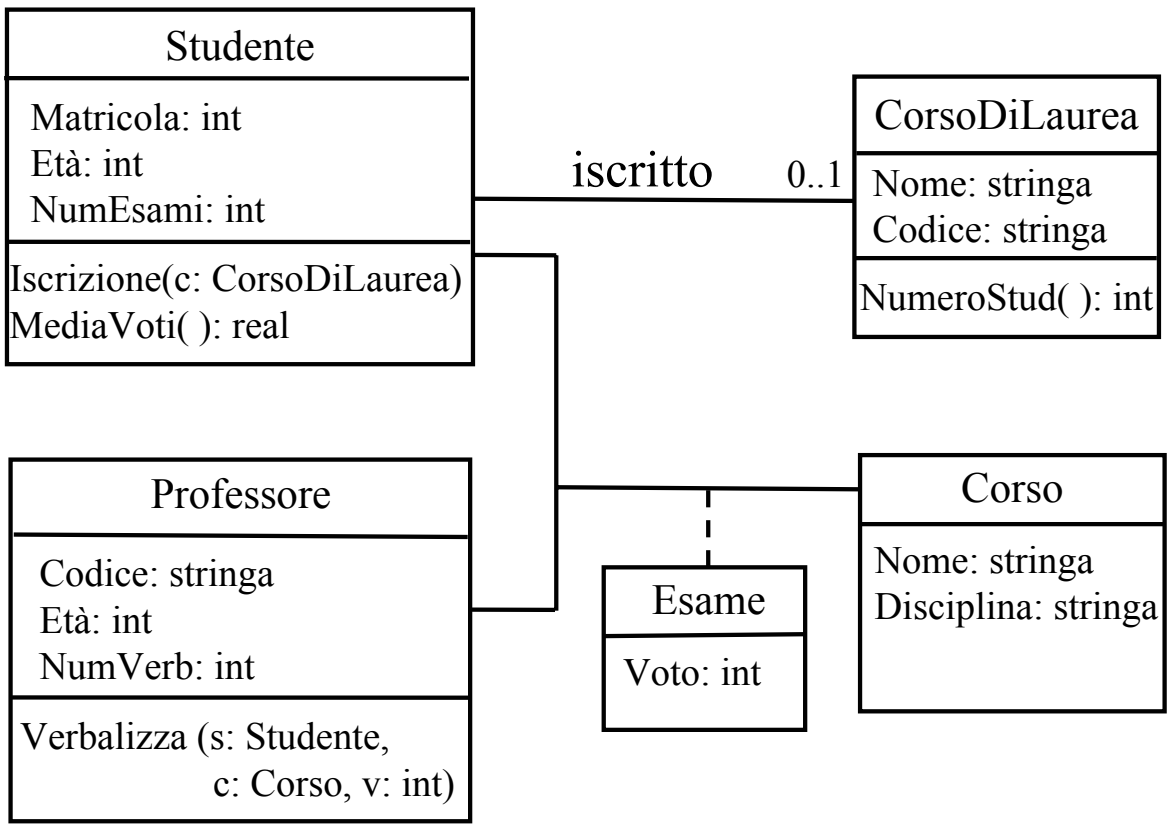


InizioSpecificaClasse CorsoDiLaurea
NumeroStud() : int
pre : nessuna
post : result è uguale al numero di studenti iscritti nel corso di laurea this
FineSpecifica

Precondizioni e postcondizioni

- Nella specifica di una operazione, nella precondizione si usa "this" per riferirsi all'oggetto di invocazione della operazione
- Nella specifica di una operazione, nella postcondizione si usa
  - "this" per riferirsi all'oggetto di invocazione della operazione nello stato corrispondente alla fine della esecuzione della operazione
  - "result" per riferirsi al risultato restituito dalla esecuzione della operazione
  - pre(alfa) per riferirsi al valore della espressione alfa nello stato corrispondente alla precondizione

Esempio di diagramma delle classi

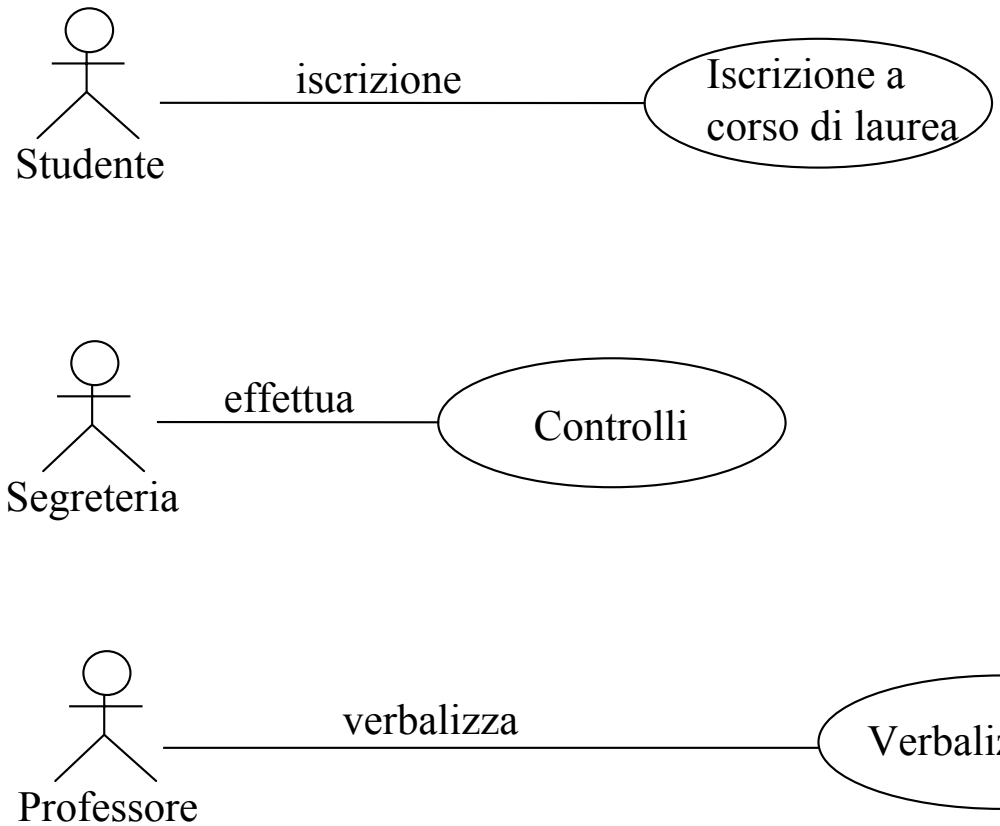


Esempio di specifica di classi

InizioSpecificaClasse Professore
    Verbalizza(s: Studente, c: Corso, v: int)
        pre : s non ha ancora sostenuto l'esame c, e $18 \leq v \leq 31$
        post : this, s e c sono collegati da un link di tipo Esame, con voto v. Inoltre vale che s.NumEsami = pre(s.NumEsami) + 1, e this.NumVerb = pre(this.NumVerb) + 1
FineSpecifica

InizioSpecificaClasse Studente
    Iscrizione(c: CorsoDiLaurea)
        pre : this non è iscritto ad alcun CorsoDiLaurea
        post : this è iscritto al CorsoDiLaurea c,
    MediaVoti() : real
        pre : this è iscritto ad un CorsoDiLaurea, e this.NumEsami > 0
        post : result è la media dei voti degli esami sostenuti da this
FineSpecifica

Esempio di diagramma di use case



Esempio di specifica di use case

InizioSpecificaUseCase Controlli

MediaVoti(s: Studente): real

pre : s è iscritto ad un CorsoDiLaurea, e s.NumEsami > 0

post : result = s.MediaVoti()

NumeroStudenti(c: CorsoDiLaurea): int

pre : nessuna

post : result = c.NumeroStud()

MediaEsami(c: CorsoDiLaurea): real

pre : c ha almeno uno studente iscritto

post : result è la media del numero di esami sostenuti dagli studenti iscritti al corso di laurea c

FineSpecifica