L4.4

Progettazione del Software

Emiliano Casalicchio

Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma – Sede di Rieti

http://www.ce.uniroma2.it/courses/PSW

Seconda Parte

La fase di analisi

- I Class e Object diagram:
  - Generalizzazioni
  - Controllo qualità

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**.

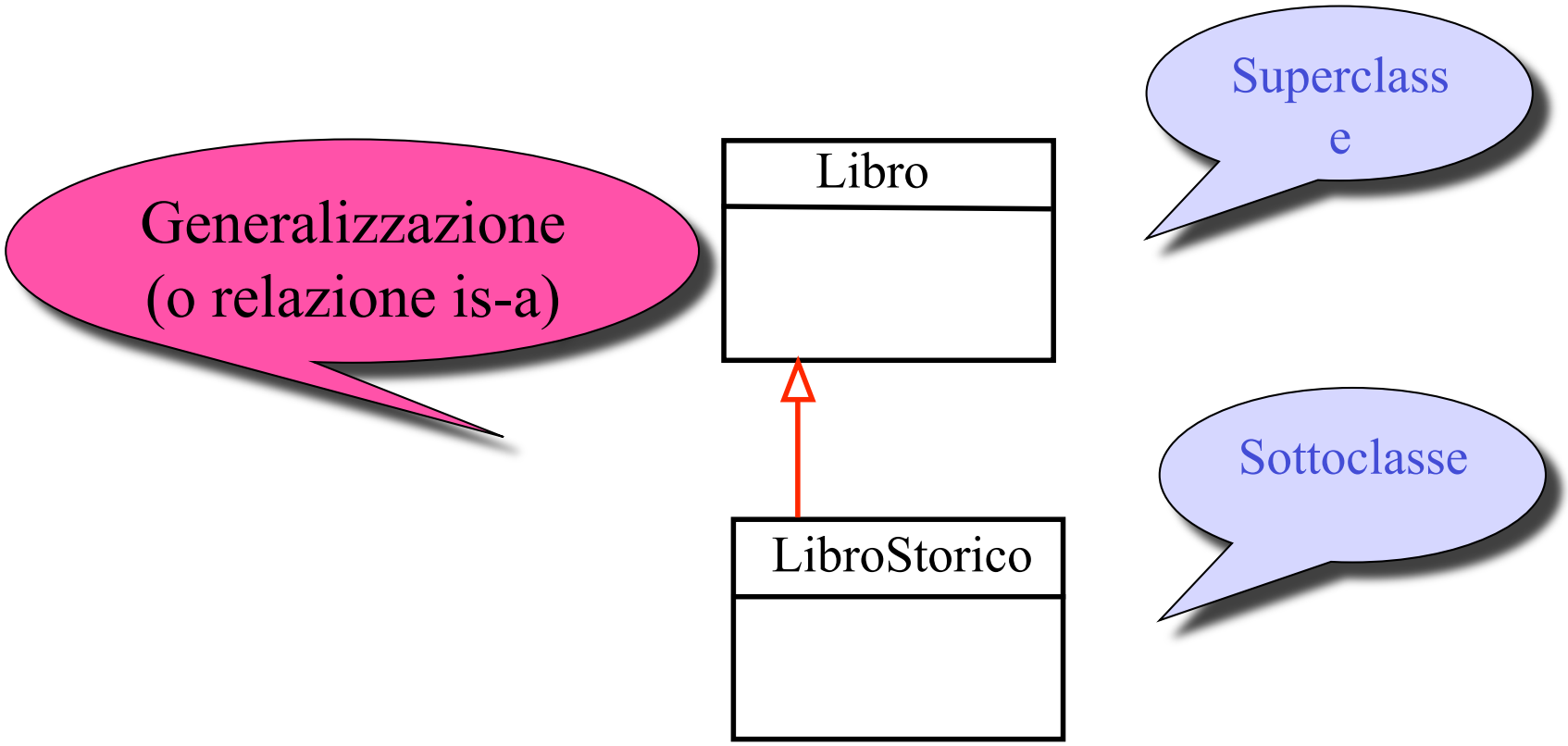
- In UML la relazione is-a si modella mediante la nozione di **generalizzazione**

- 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):

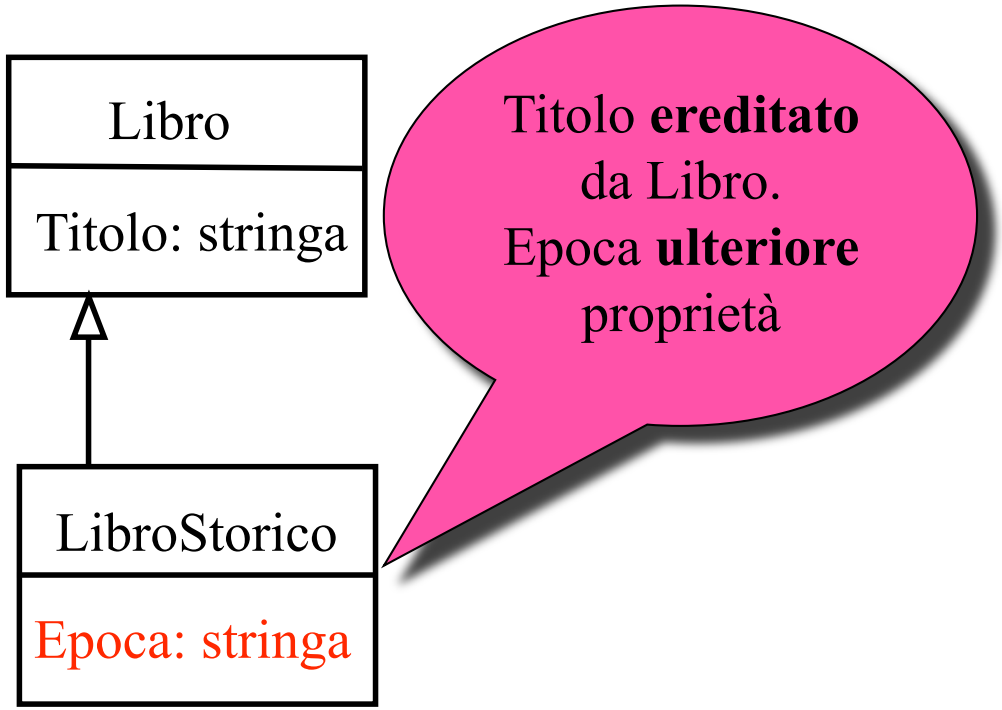


Ereditarietà in UML

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

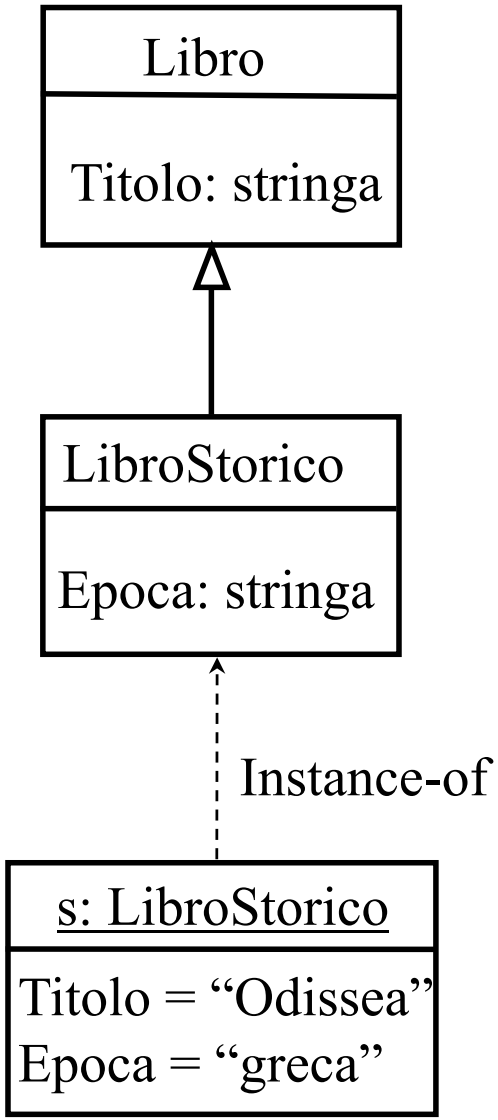
Dal fatto che
1. Ogni istanza di Libro ha un Titolo
2. Ogni istanza di LibroStorico è una istanza di Libro

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



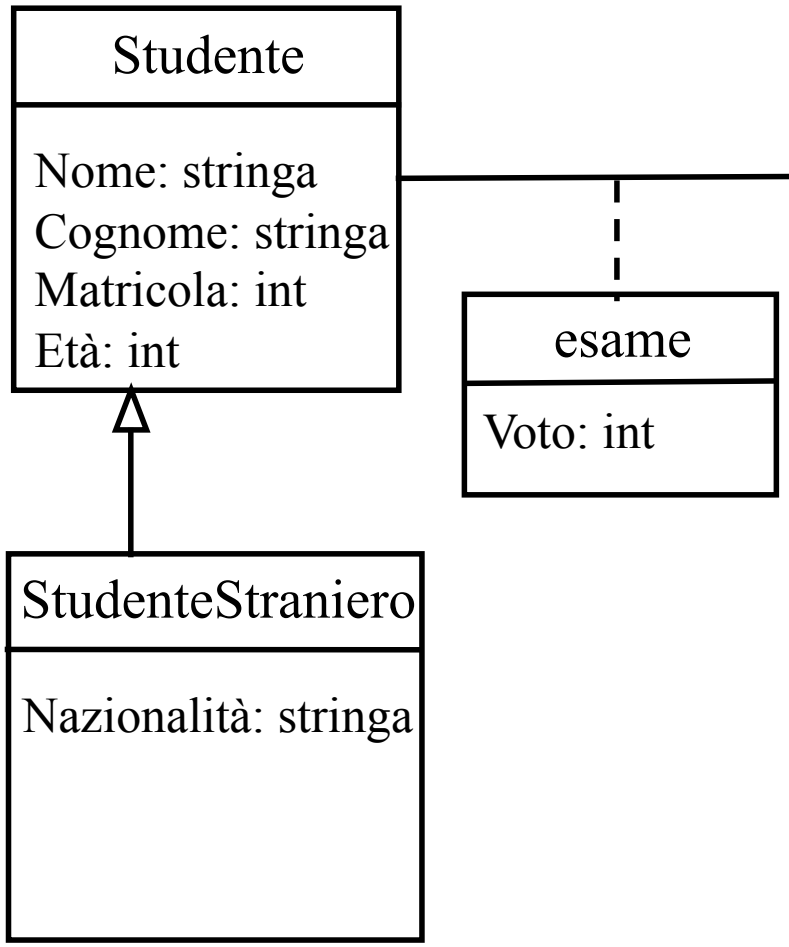
Ragionamento sillogistico (cfr. opera di Aristotele più di due millenni fa)

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

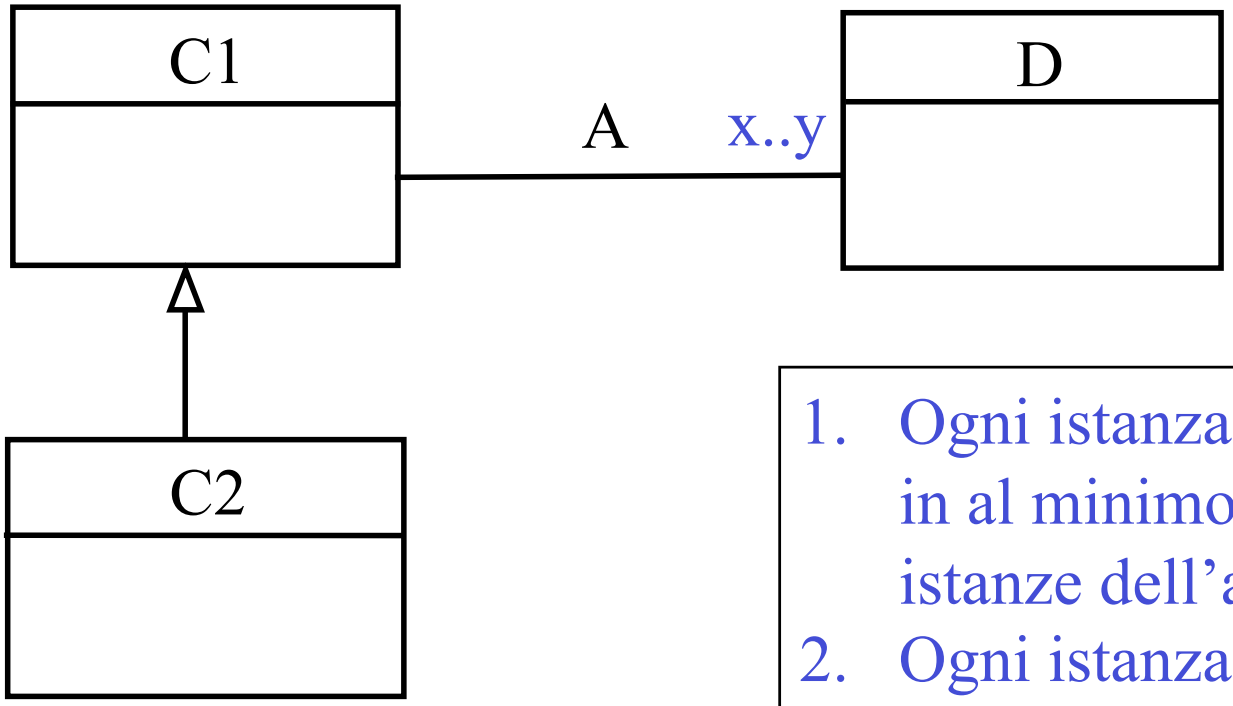


1. Ogni istanza di Studente può essere coinvolta in un numero qualunque di istanze della associazione "esame"
2. Ogni istanza di StudenteStraniero è una istanza di Studente

quindi

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

Ereditarietà sulle molteplicità

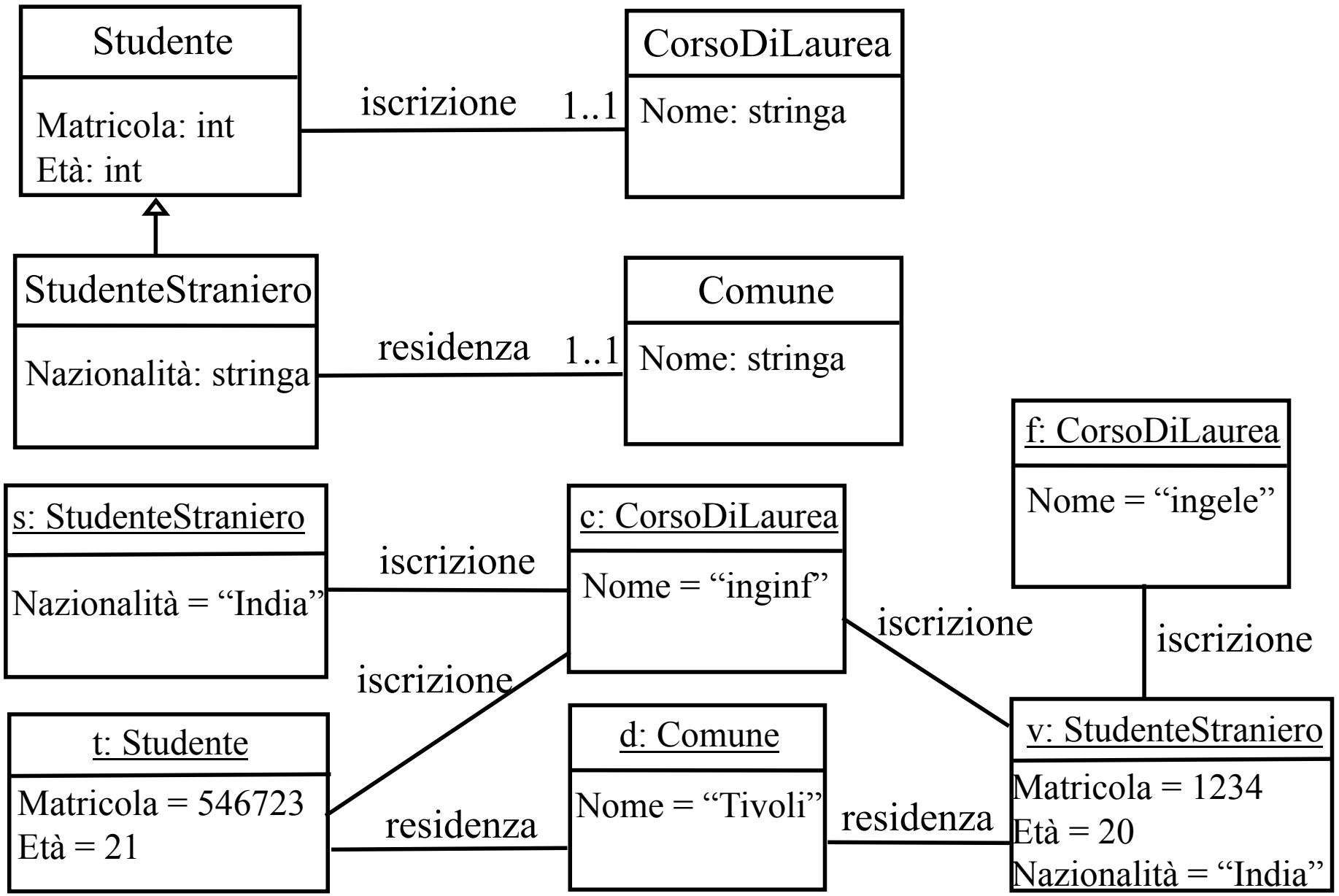


1. Ogni istanza di C_1 è coinvolta in al minimo x e al massimo y istanze dell'associazione A
2. Ogni istanza di C_2 è una istanza di C_1

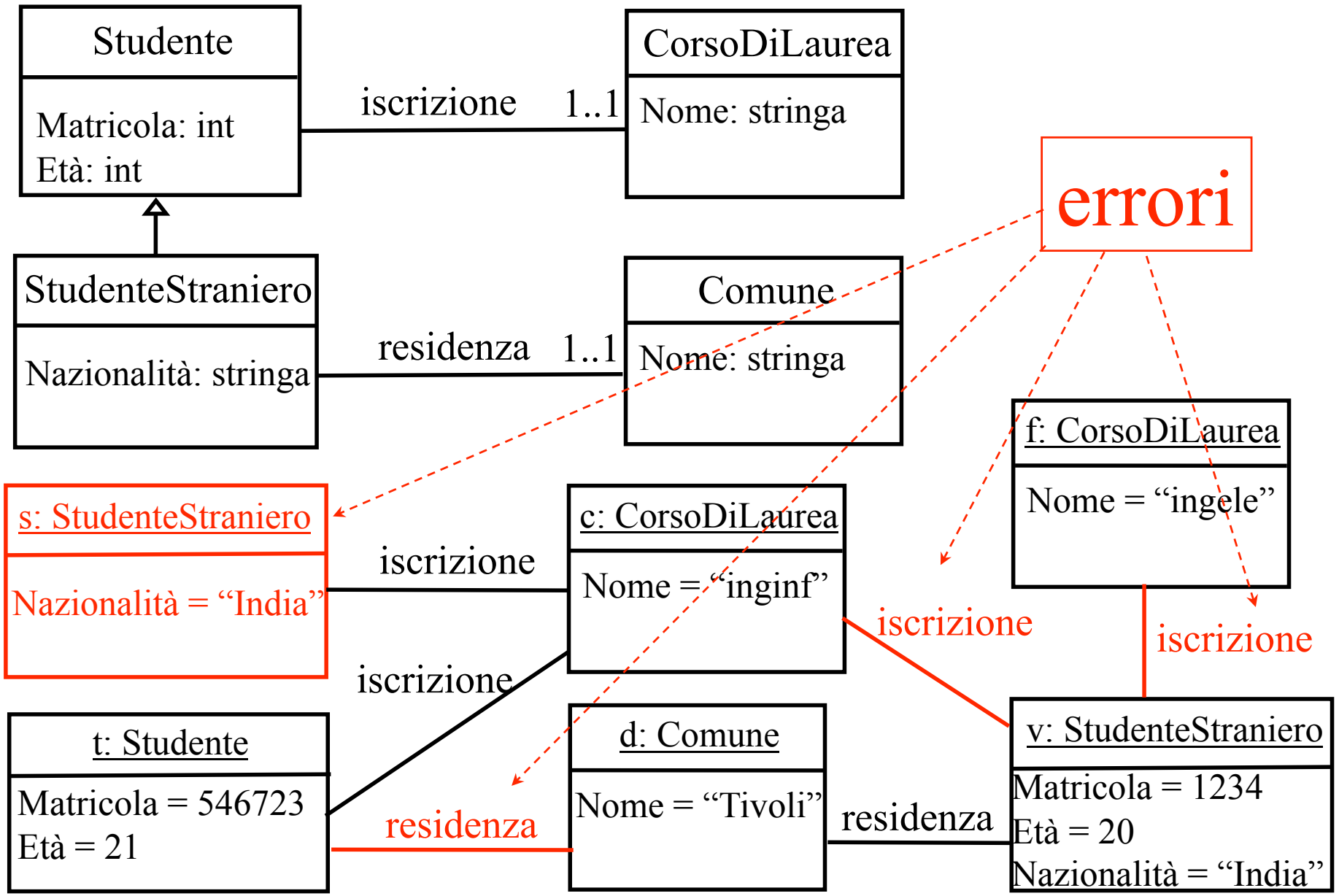
quindi

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

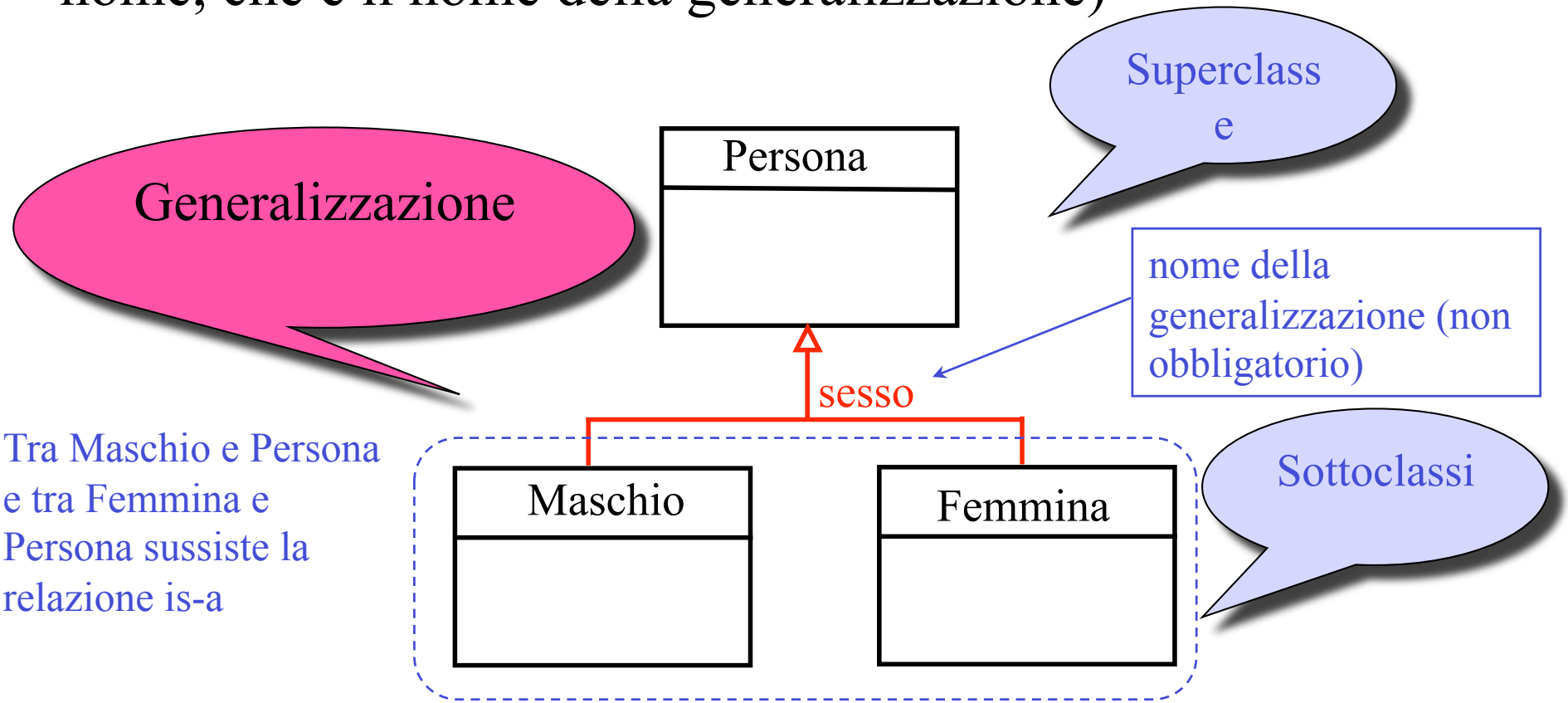


Soluzione dell'esercizio 5



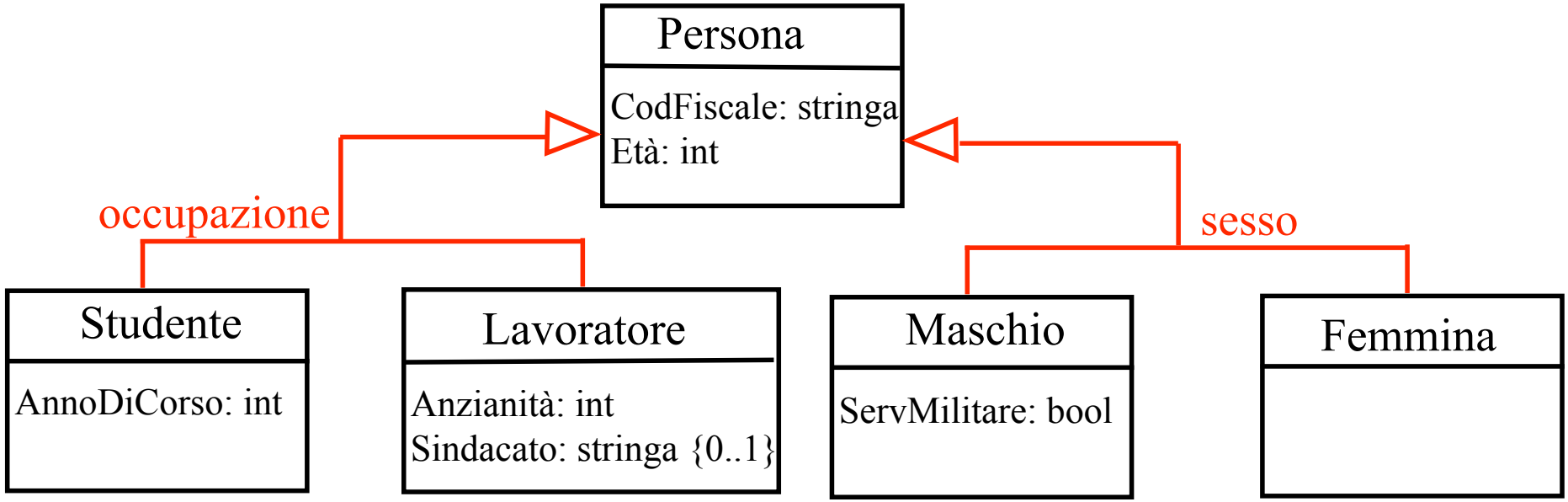
Generalizzazione in UML

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

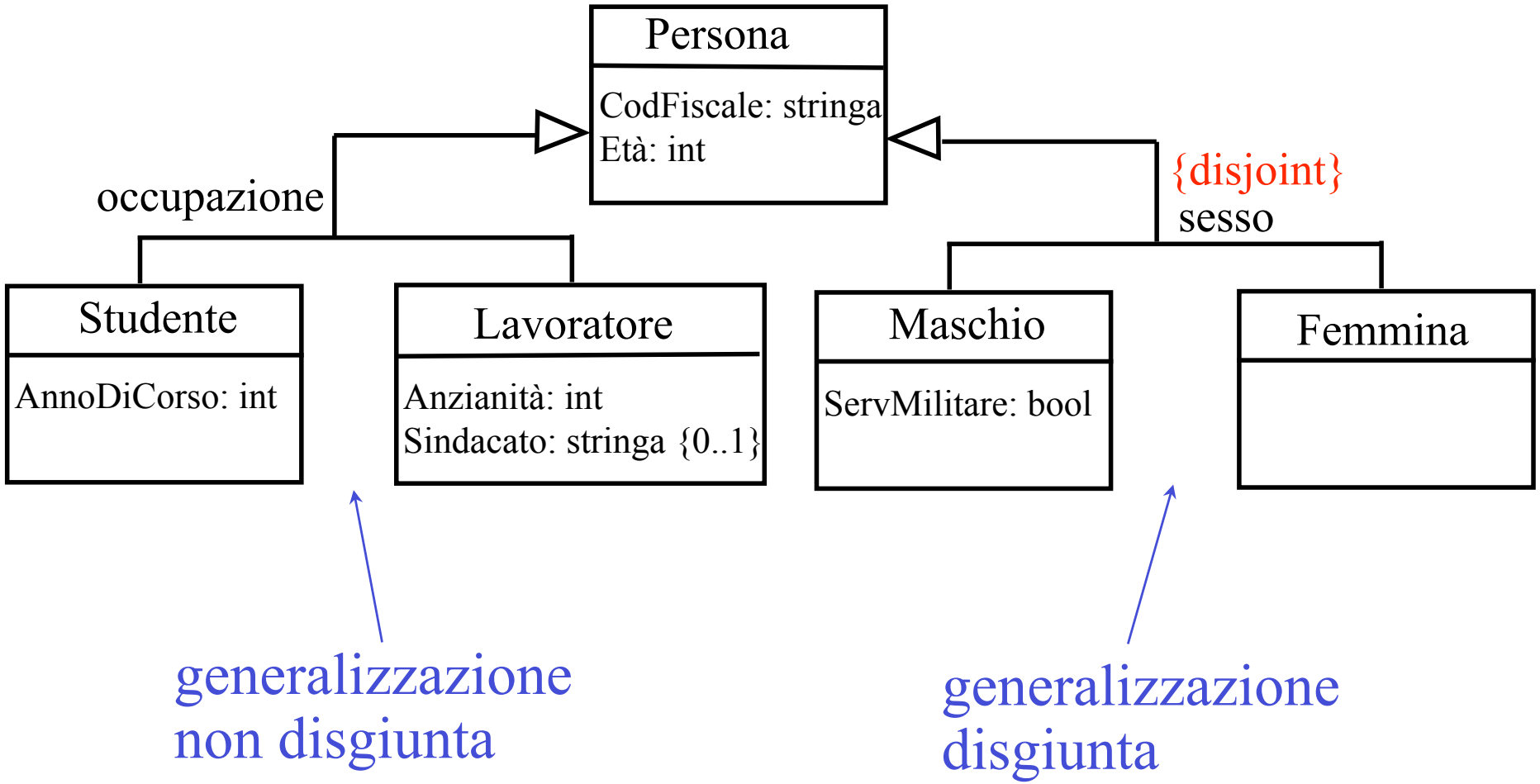
La stessa superclasse può partecipare a diverse generalizzazioni



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

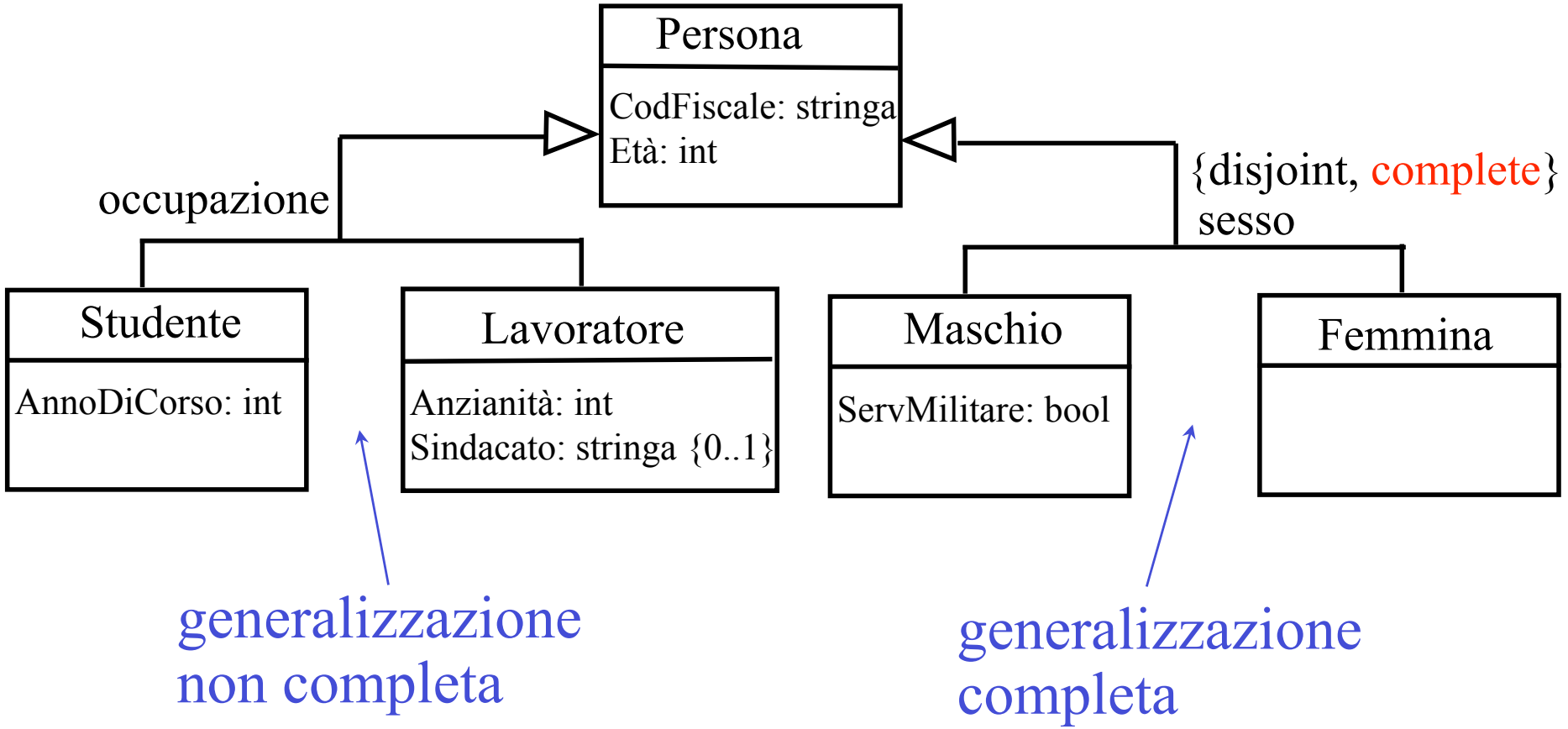
Generalizzazioni disgiunte

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

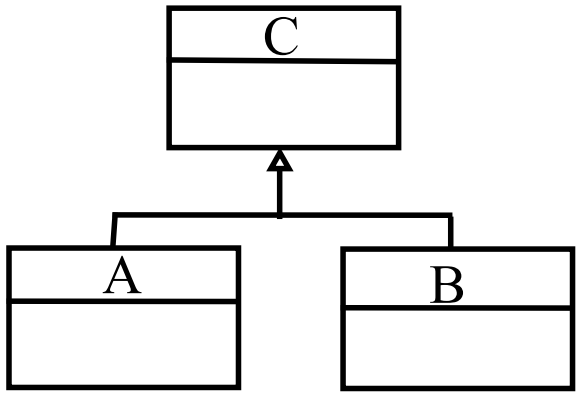
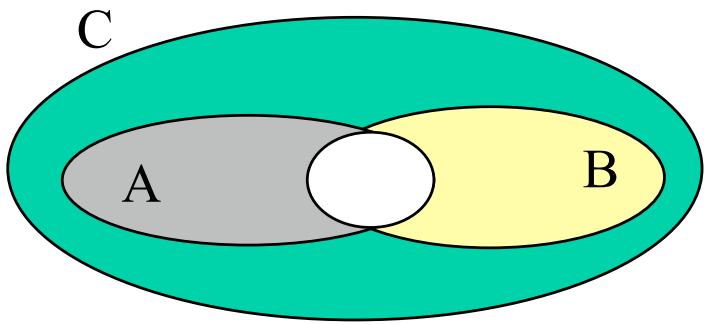
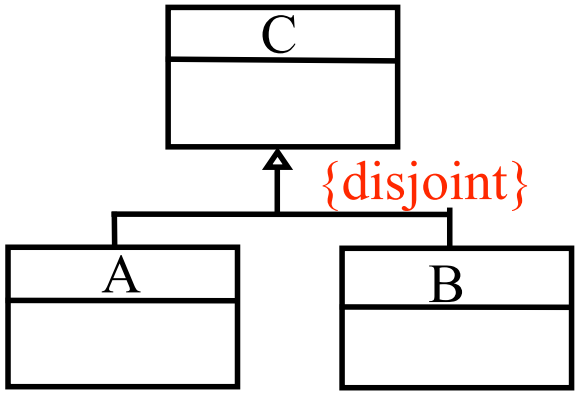
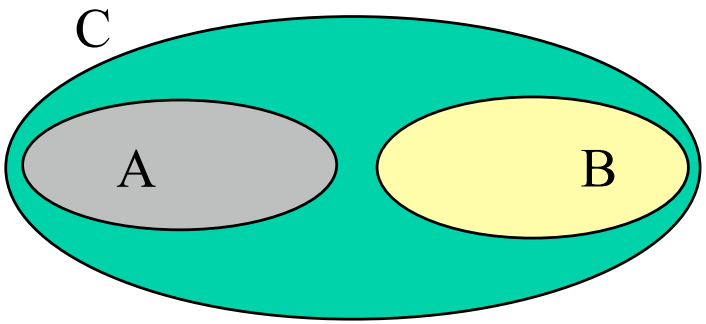


Generalizzazioni complete

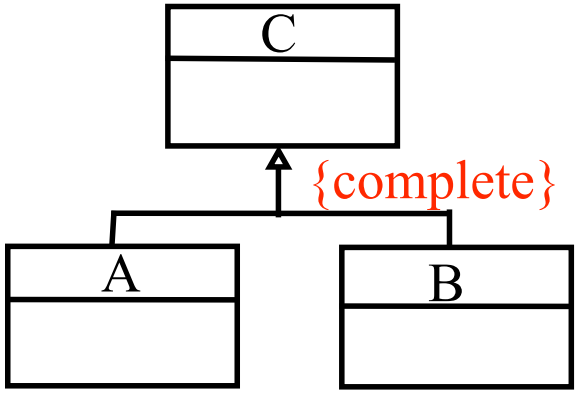
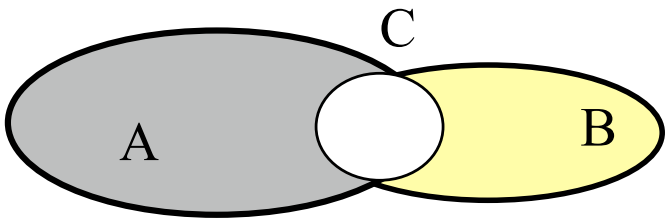
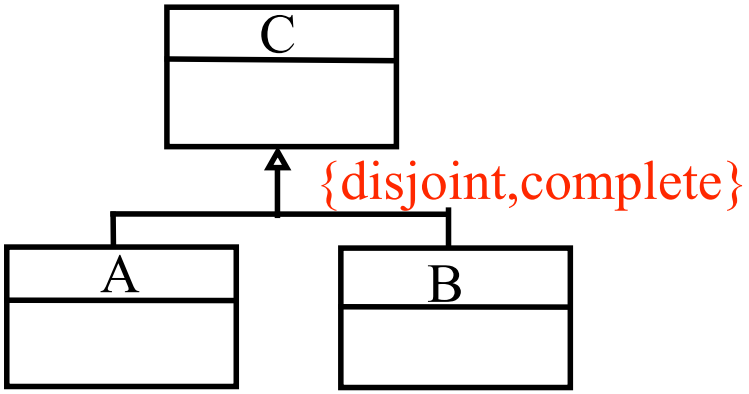
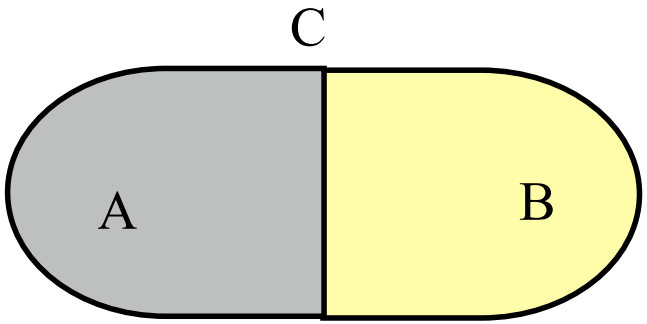
Una generalizzazione può essere completa (l'unione delle istanze delle sottoclassi è uguale all'insieme delle istanze della superclasse) o no



Generalizzazioni

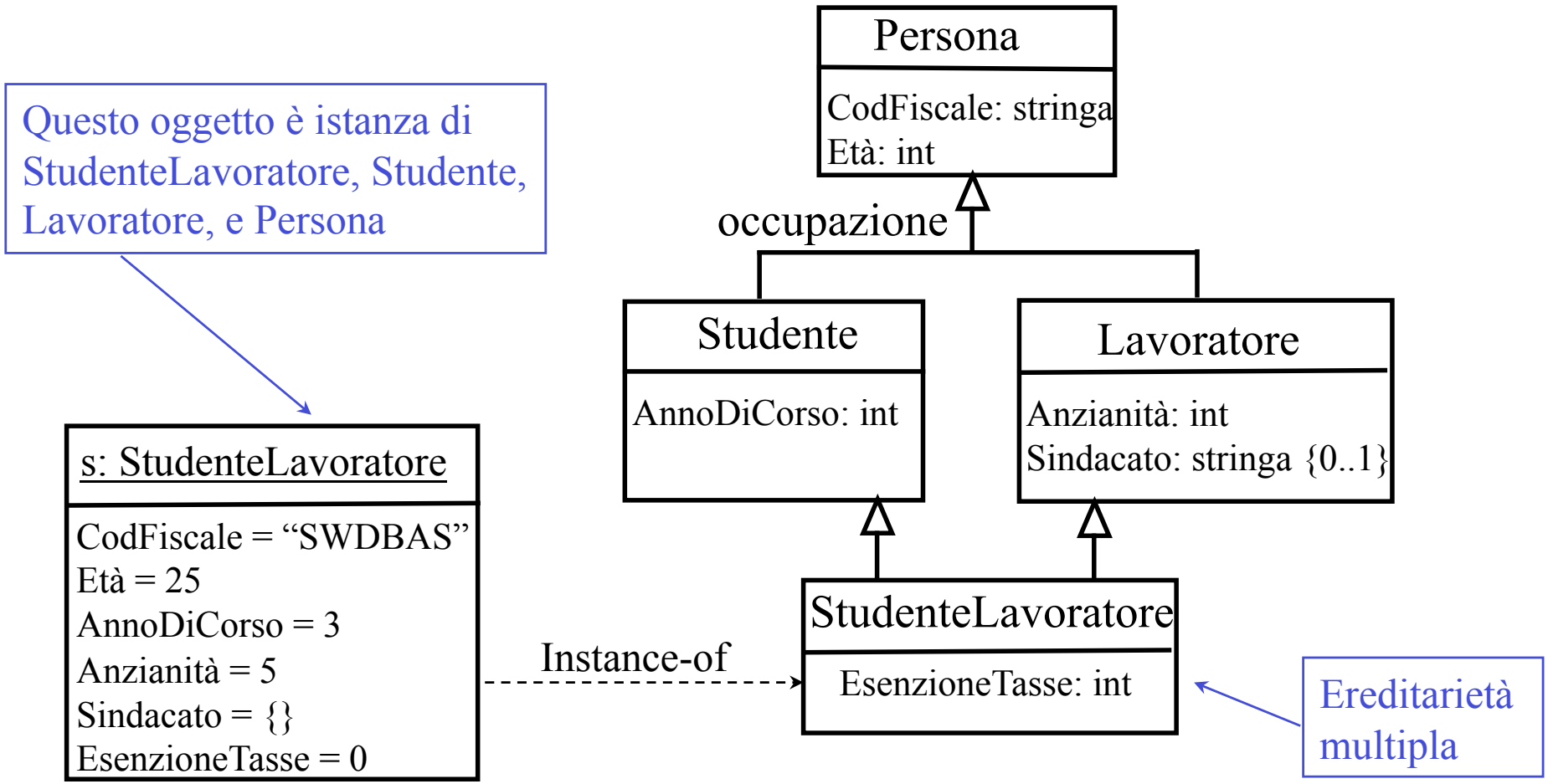
Livello intensionale	Livello estensionale
	
	

Generalizzazioni

Livello intensionale	Livello estensionale
	
	

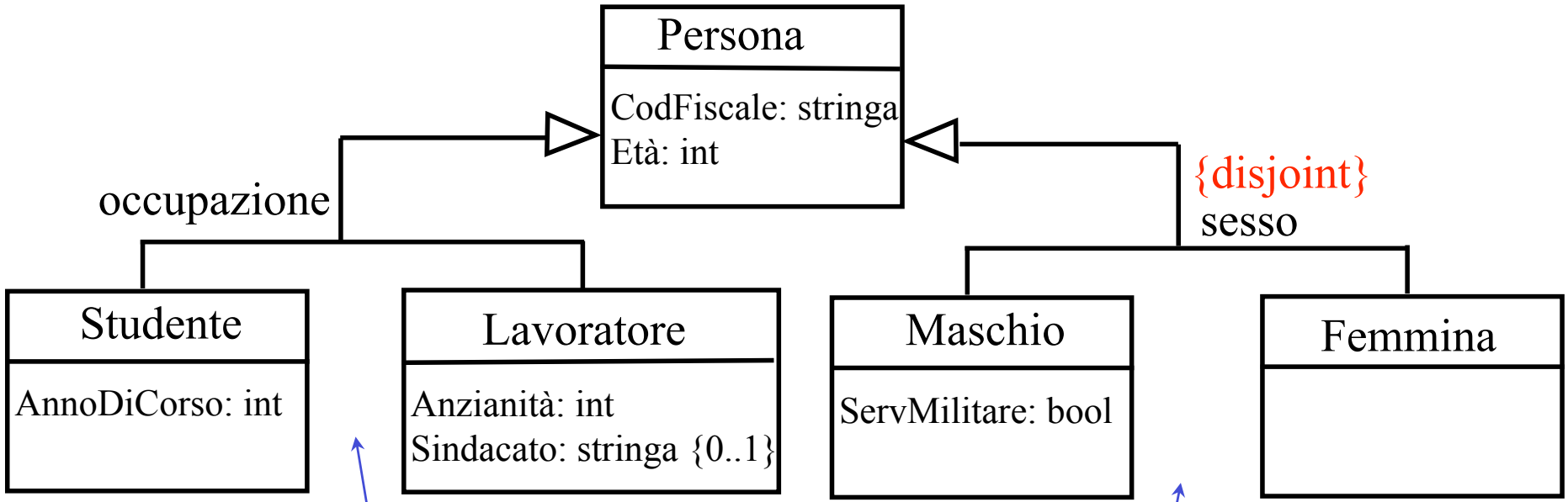
Ereditarietà multipla

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

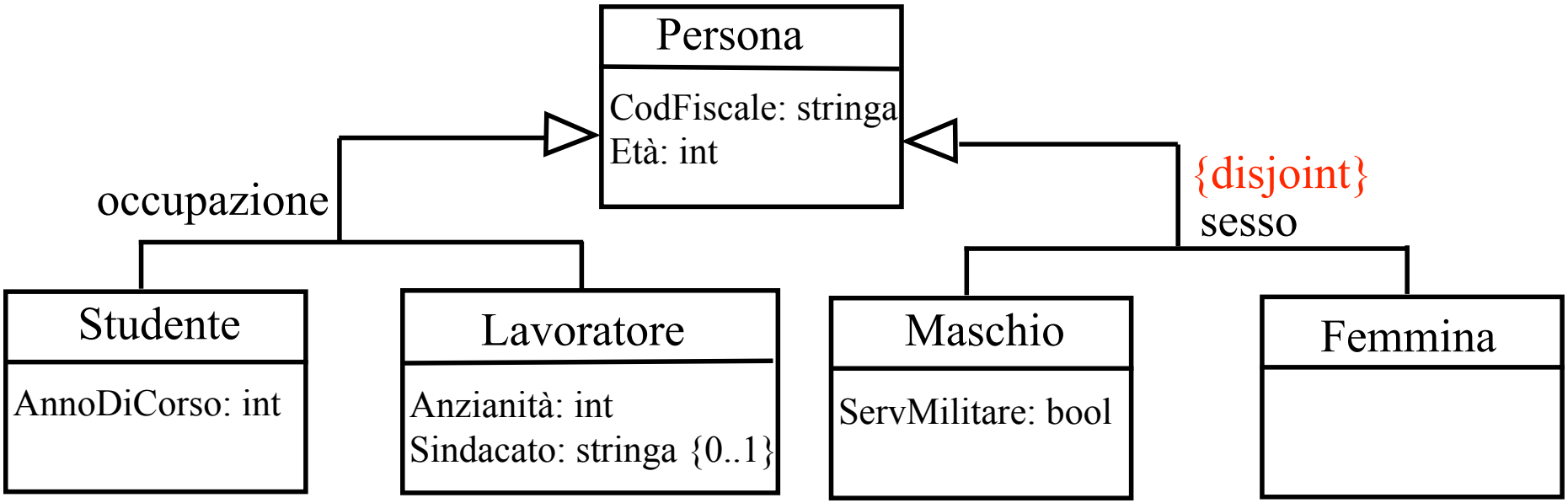


È possibile definire una classe (ad Esempio **StudenteLavoratore**) che è sottoclasse sia di Studente sia di Lavoratore

Non è possibile definire una classe che è **sottoclasse sia di Maschio sia di Femmina**

Il problema delle classi disgiunte (1)

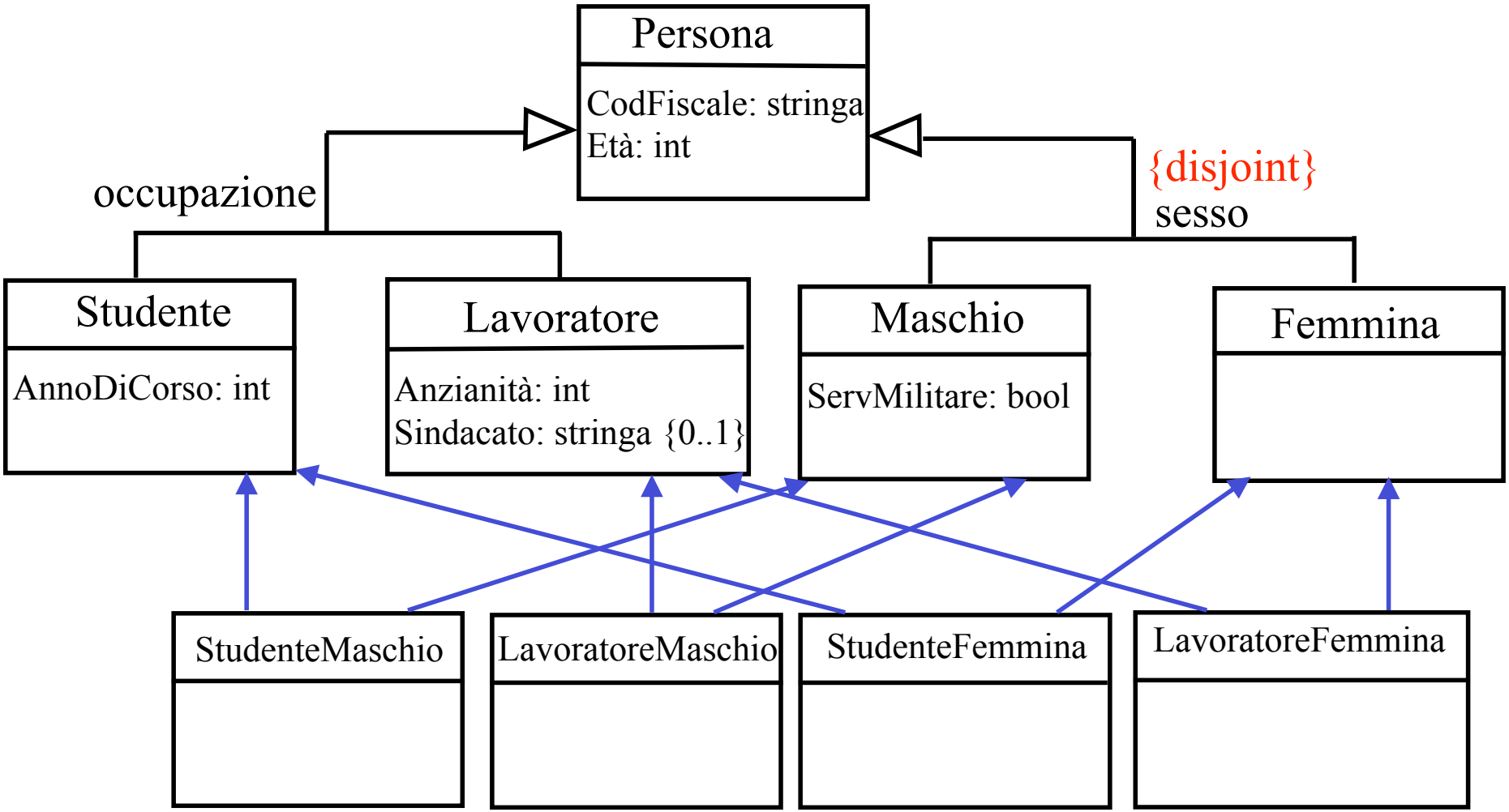
Consideriamo la gerarchia di generalizzazione seguente:



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)

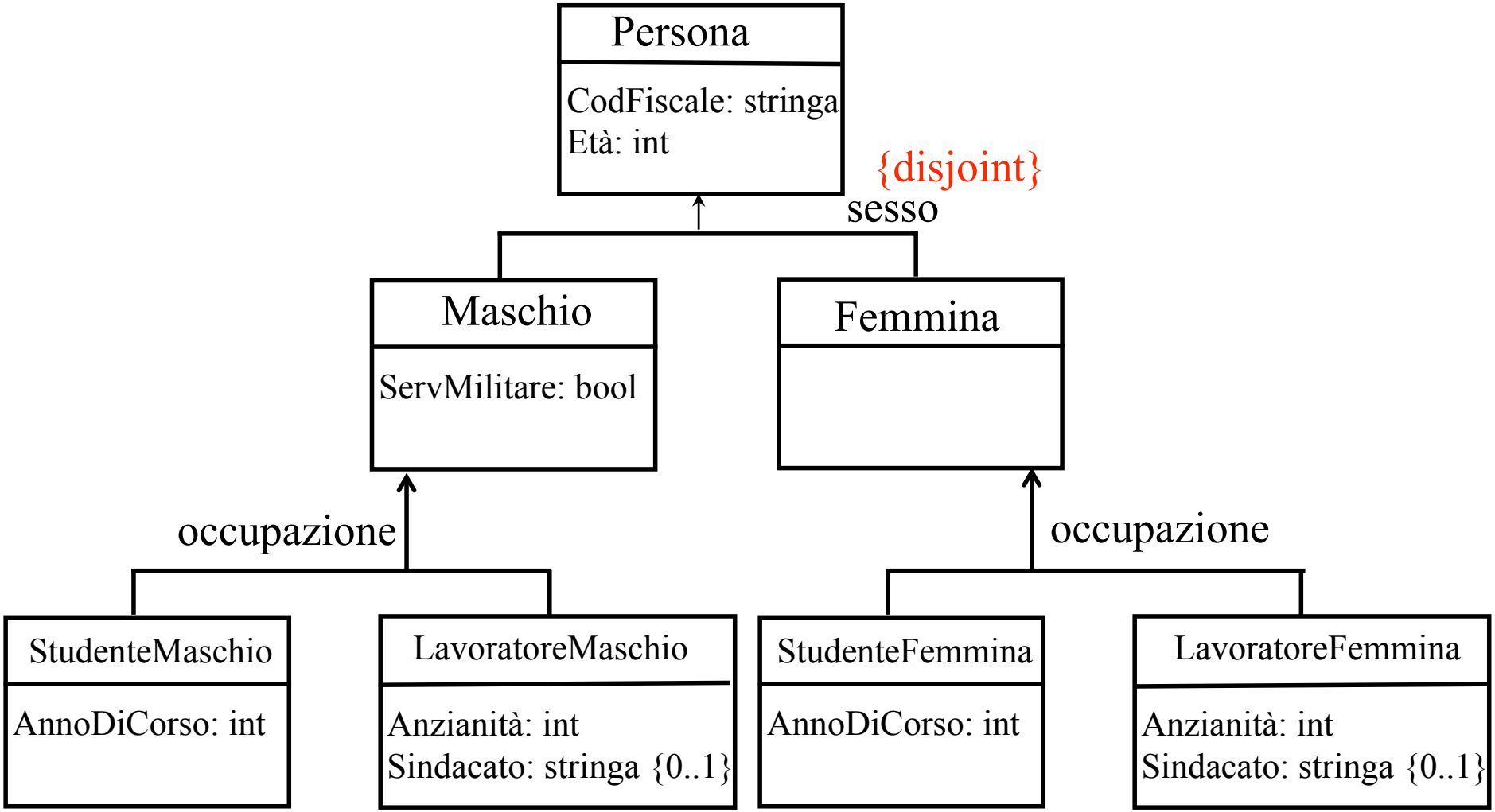
Il problema delle classi disgiunte (2)

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

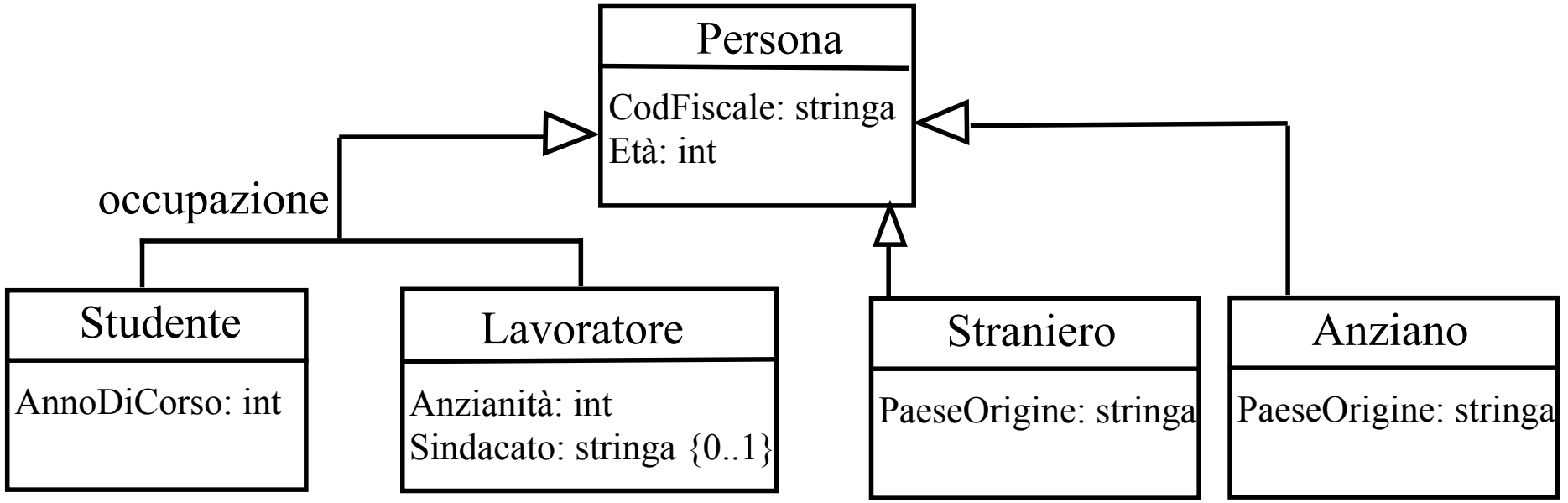


Il problema delle classi disgiunte (3)

Secondo modo:



Differenza tra due isa e una generalizzazione



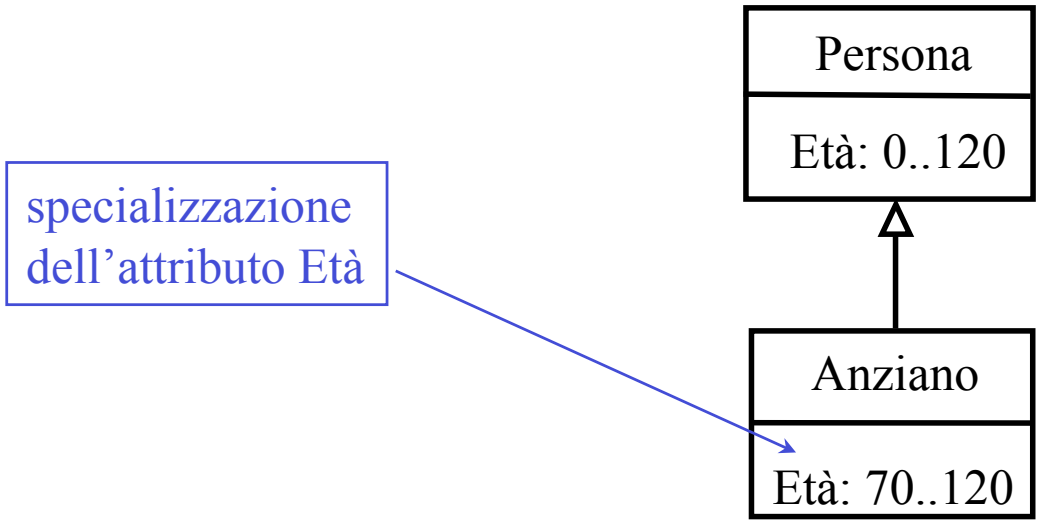
Le due sottoclassi derivano da **uno stesso criterio** di classificazione delle istanze della superclasse

Le due sottoclassi sono indipendenti, nel senso che il loro significato **non deriva dallo stesso criterio** di classificazione delle istanze della superclasse

Specializzazione

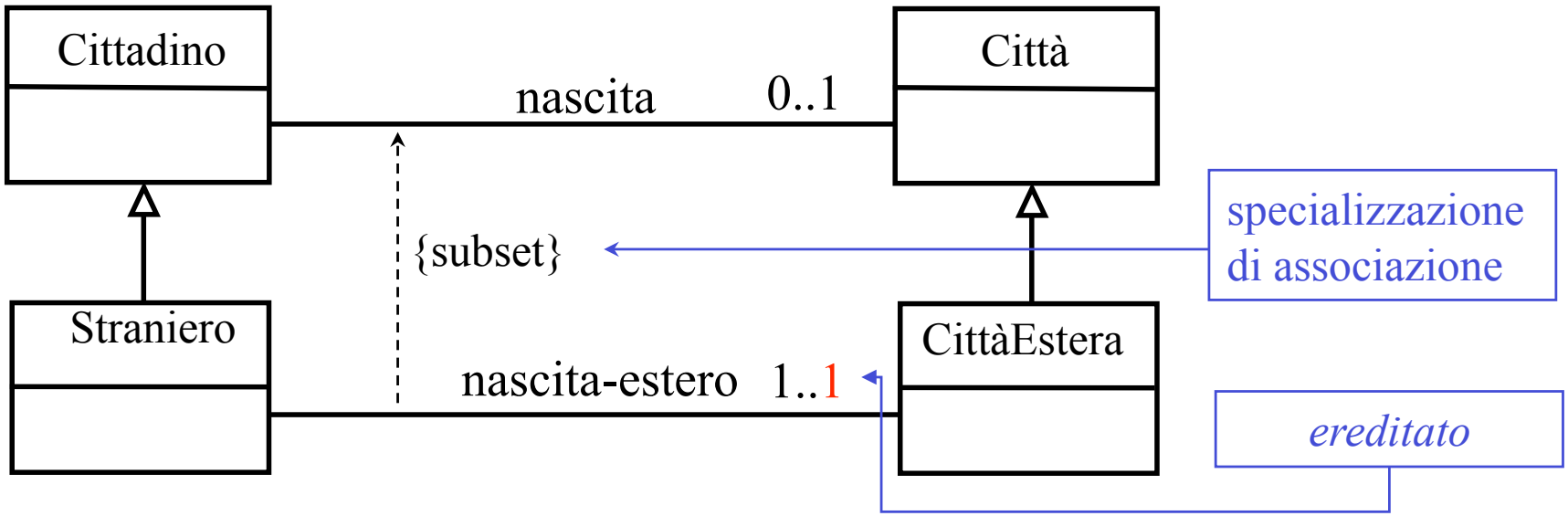
In una generalizzazione la sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche specializzare le proprietà ereditate dalla superclasse.

- 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_1.



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_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)

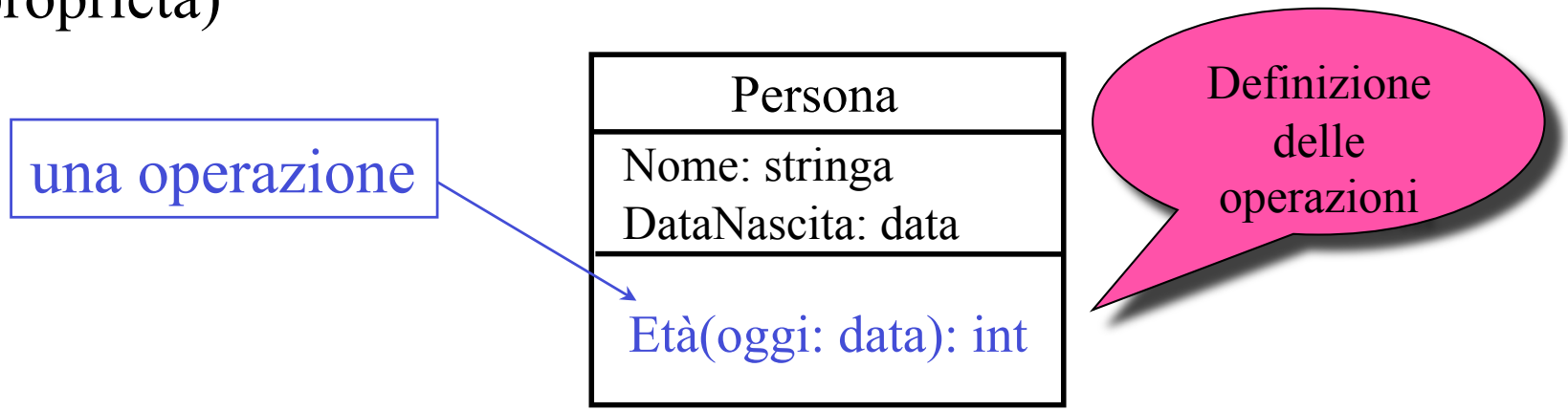


Operazioni

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),

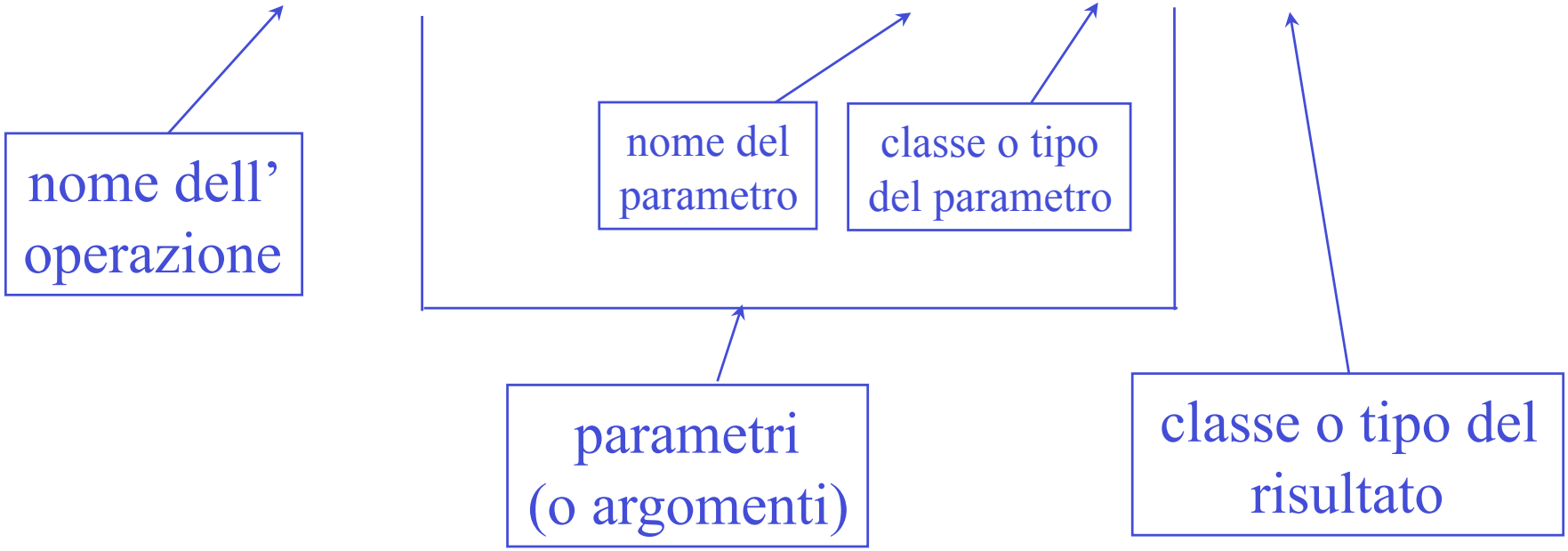
- o per calcolare le proprietà
- o per effettuare cambiamenti di stato (cioè per modificare le proprietà)



Definizione di una operazione

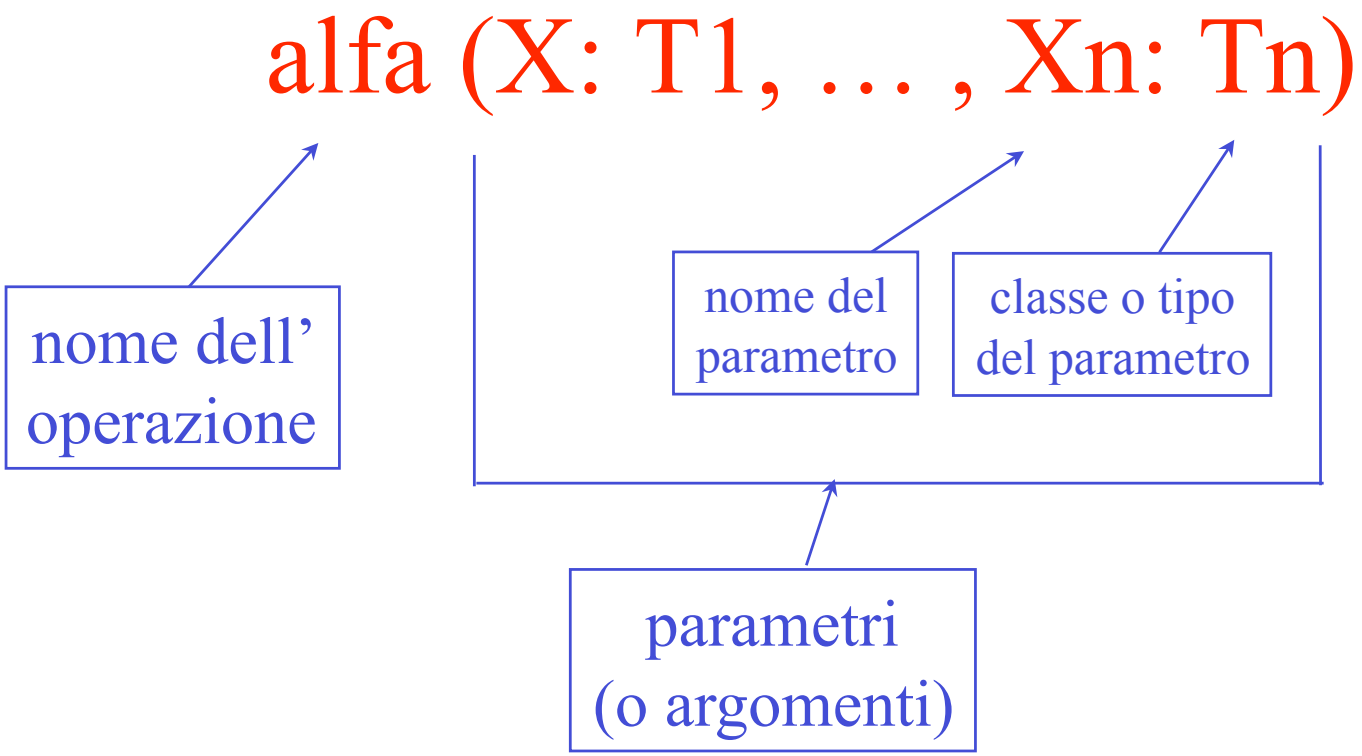
In una classe, una operazione si definisce specificando la **segnatura** (nome, parametri e il tipo del eventuale risultato) e **non il metodo** (cioè non la specifica di cosa fa l'operazione)

alfa (X: T1, … , Xn: Tn): T



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 (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 definita

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 6

Tracciare il diagramma delle classi corrispondenti alle seguenti specifiche:

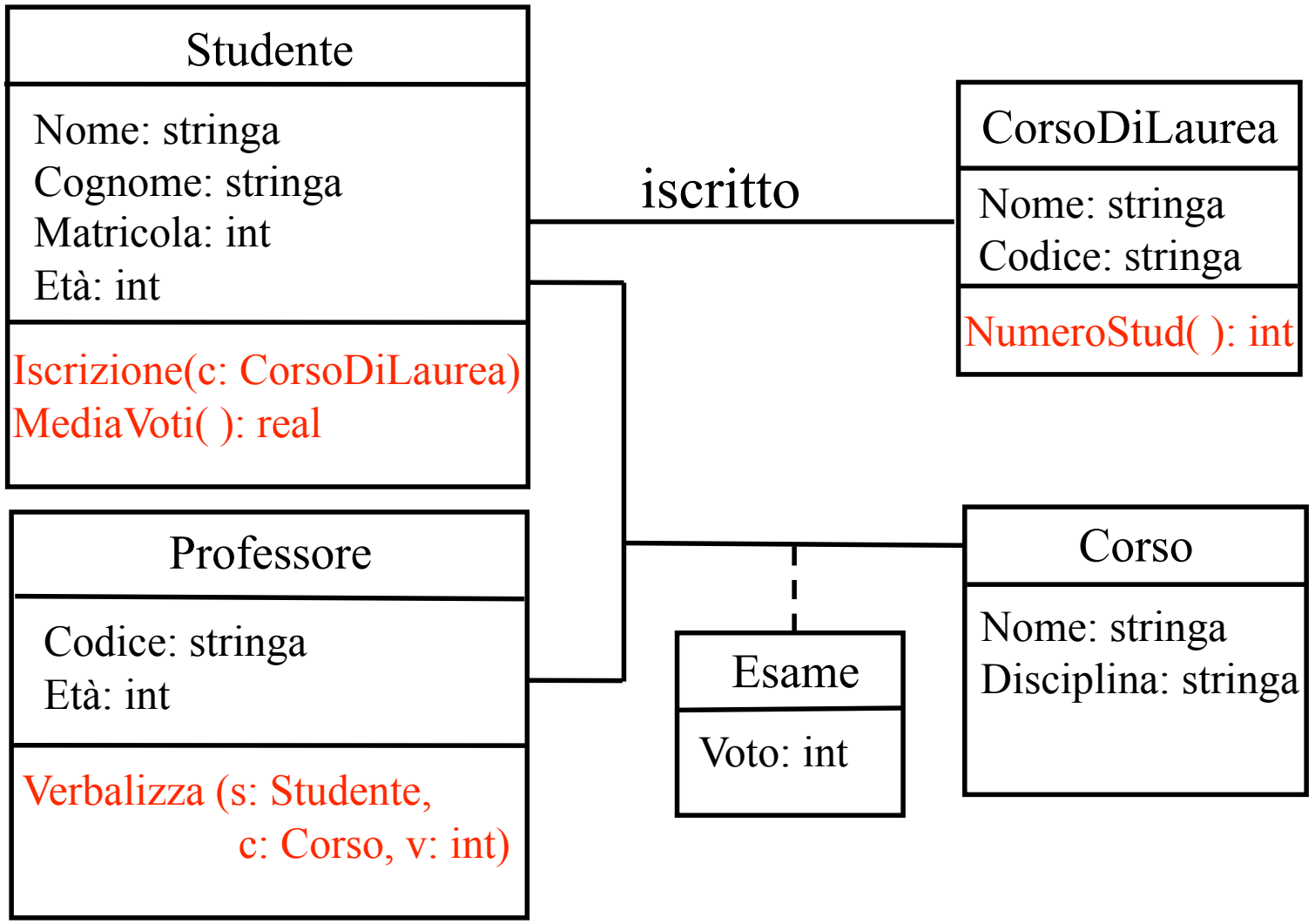
Si vogliono modellare gli studenti (con nome, cognome, numero di matricola, età), il corso di laurea in cui sono iscritti, ed i corsi di cui hanno sostenuto l'esame, con il professore che ha verbalizzato l'esame, ed il voto conseguito. Di ogni corso di laurea interessa il codice e il nome. Di ogni corso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

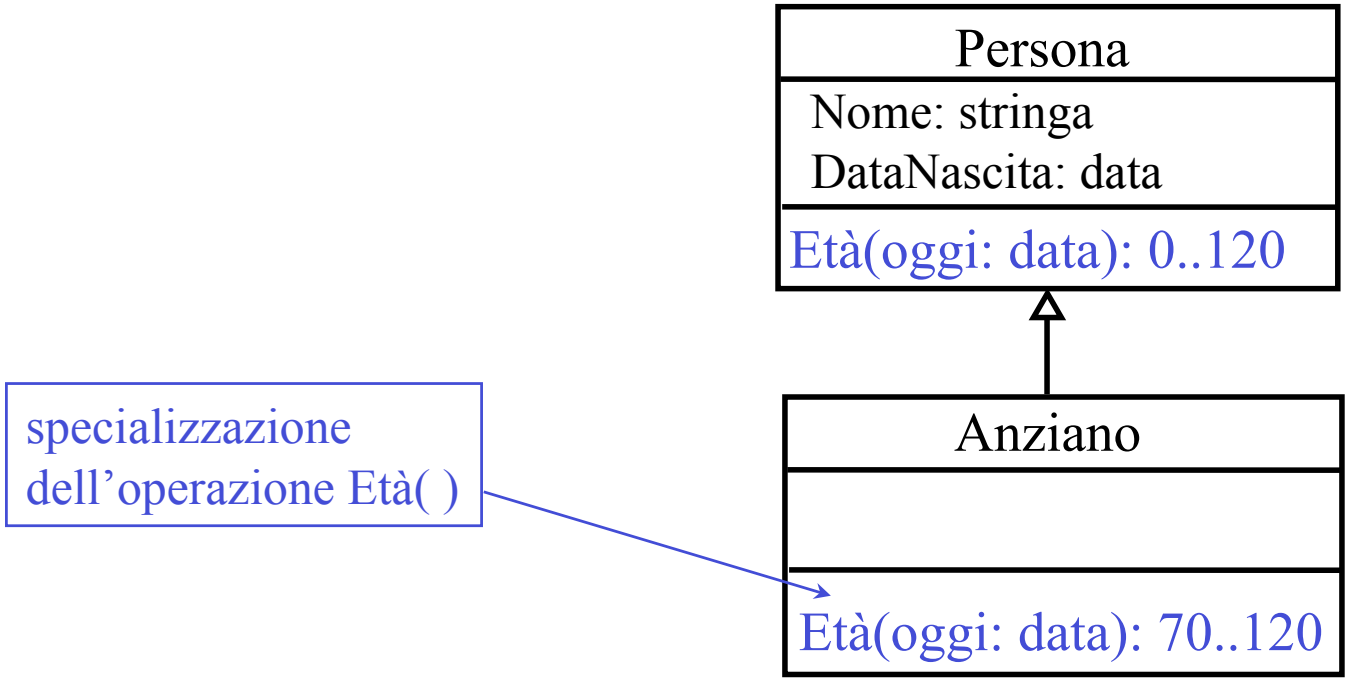
La segreteria vuole periodicamente calcolare la media dei voti di uno studente, e il numero di studenti di un corso di laurea.

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

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 si possono usare tipi definibili attraverso costruttori come
  - **Record,**
  - **Insieme,**
  - **Lista,**
  - **ecc.**

Osservazione sui tipi (2)

- Ad esempio, si può usare il tipo **indirizzo** come record con campi
  - "strada" (di tipo stringa) e
  - "numero civico" (di tipo int)
- Oppure, si può usare 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?*
- *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

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

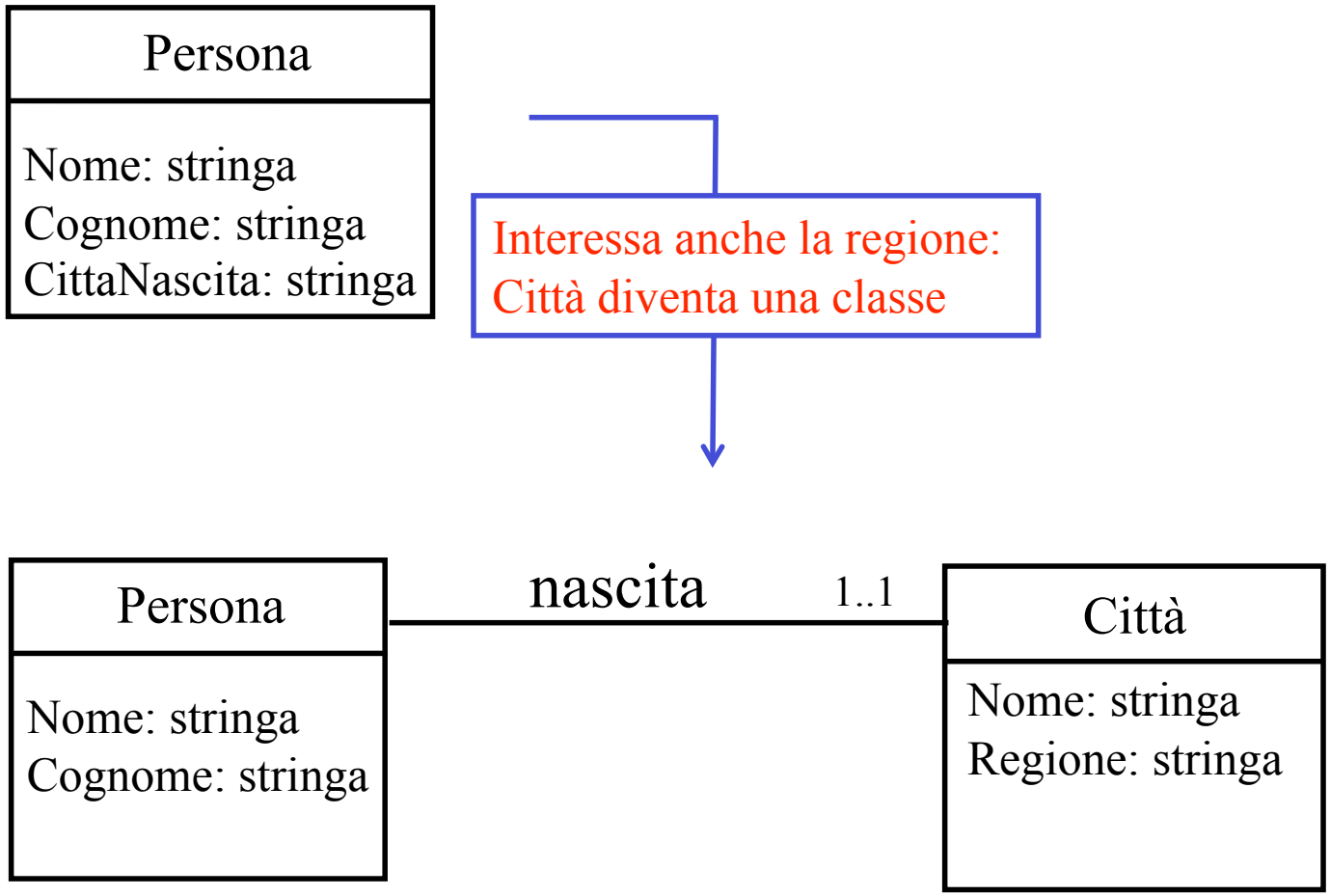
	Classe	**Tipo**
Istanze	**oggetti**	**valore**
Istanze identificate da	**identificatore di oggetto**	**valore**
Uguaglianza	**basata su identificatore**	**basata su valore**
Realizzazione	**da progettare**	**tipicamente predefinita, oppure basata su strutture di dati predefinite**

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
    - 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

Scelta tra attributi e classi

Le scelte possono cambiare durante l'analisi. Esempio:

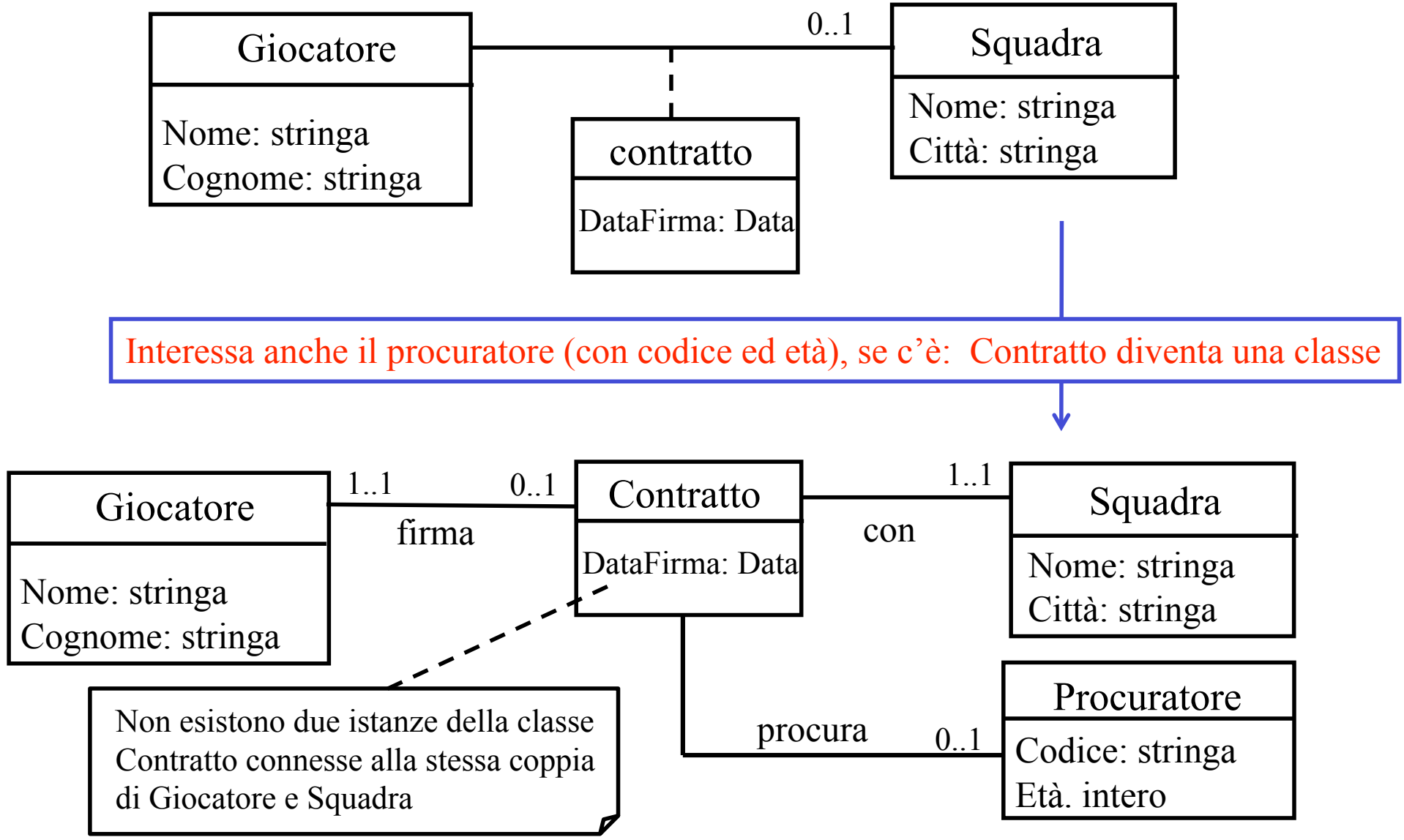


Scelta tra classi e associazione

- 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**

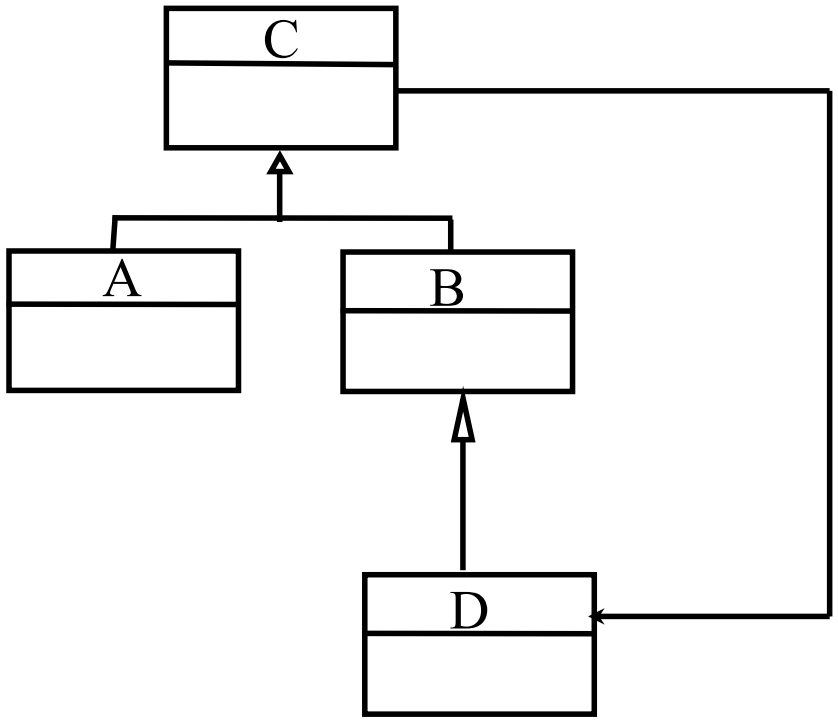
Scelta tra classi e associazioni

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!

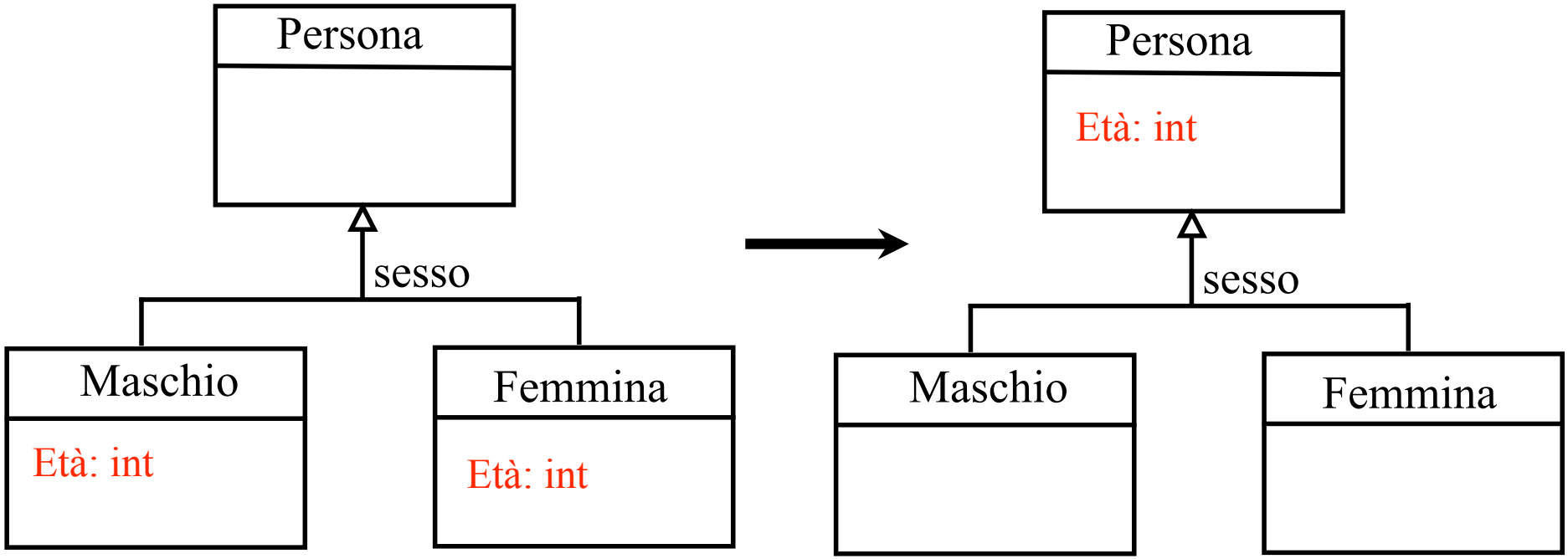
Verifiche sulle specializzazioni

- Specializzazione di un attributo: se una classe C_1 ha un attributo A di tipo T_1, 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)

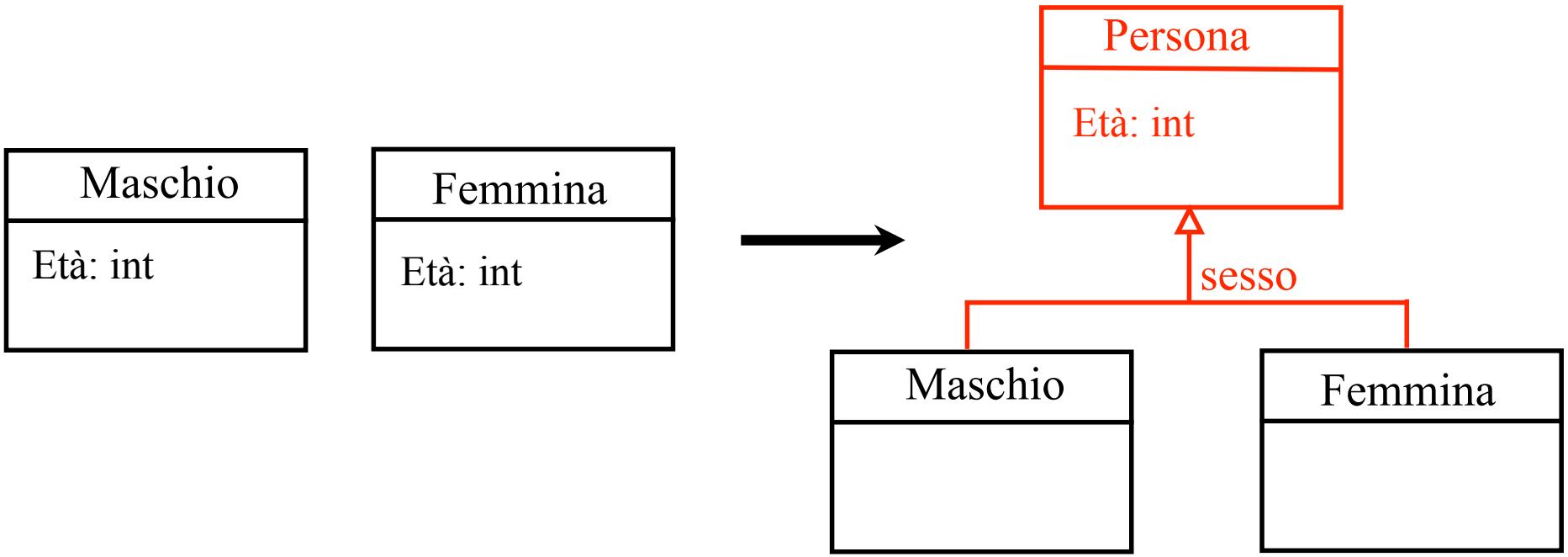
Si possono applicare ulteriori generalizzazioni?

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



Si possono applicare ulteriori generalizzazioni?

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



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 un 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 un Use Case

La specifica di un 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 un 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 un Use Case, la specifica di una operazione ha la seguente forma:

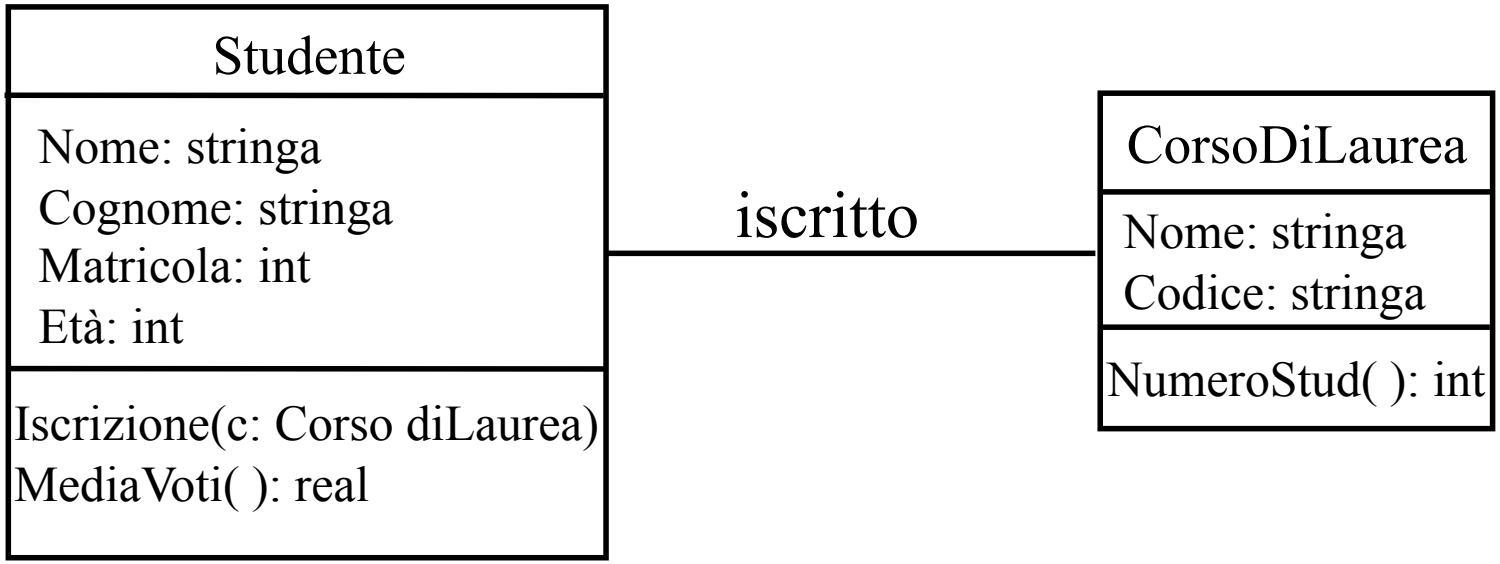
alfa (X: T1, … , Xn: Tn): T

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

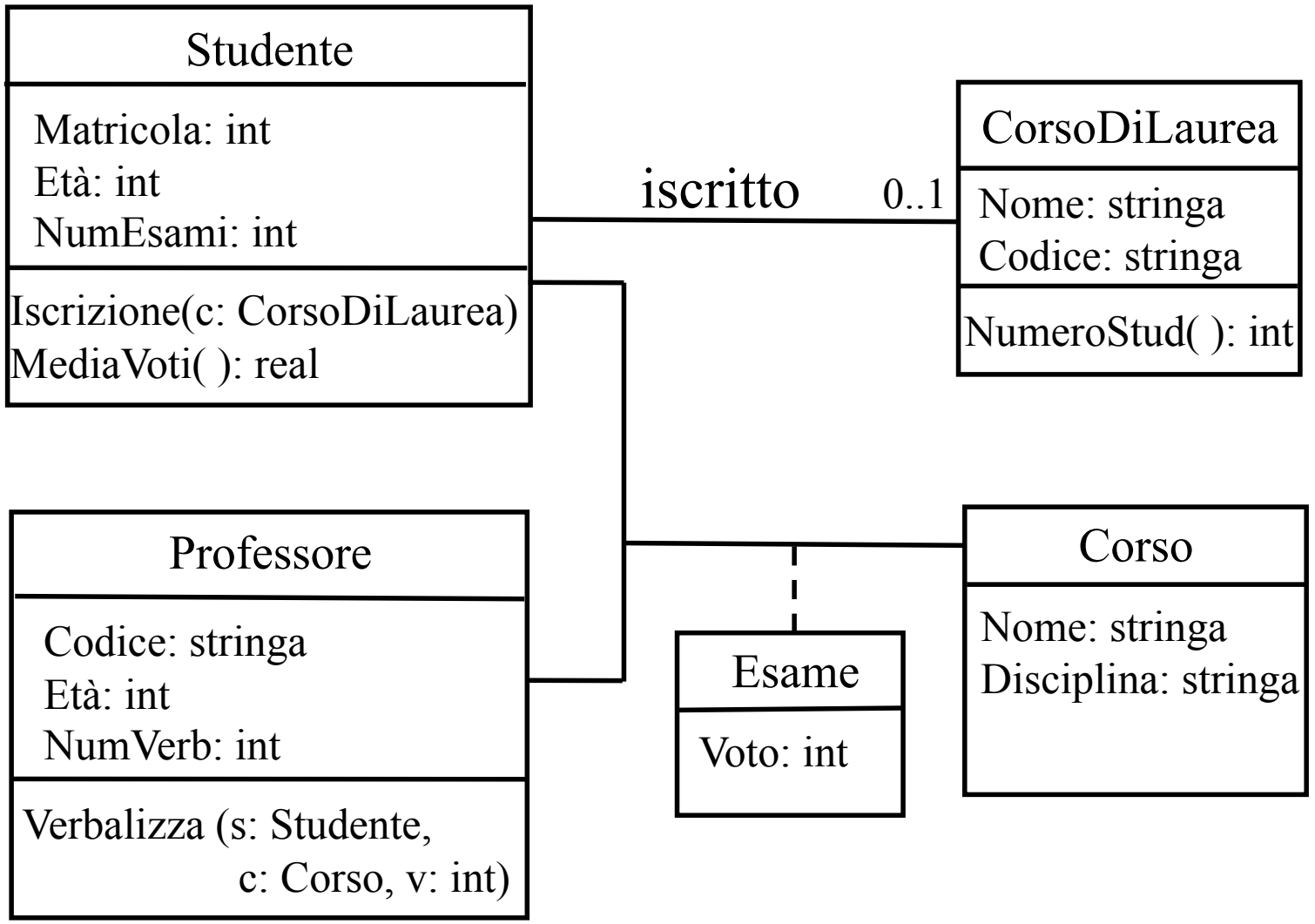
post : **result** è uguale al numero di studenti iscritti nel corso di laurea **this**

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 ≤ v ≤ 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.NumEsami > 0
        post : result è la media dei voti degli esami sostenuti da this
FineSpecifica

Specifica mediante una notazione formale

InizioSpecificaClasse Professore

Verbalizza(s: Studente, c: Corso, v: int)

pre : $\neg(\exists\ p \mid p \in \text{Professore} \wedge \langle s,p,c\rangle \in \text{Esame}) \wedge 18 \leq v \leq 31$

post : $\text{Esame} = \text{pre}(\text{Esame}) \cup \{\langle s,\text{this},c\rangle\} \wedge$
$\text{Esame.voto}(s,\text{this},c) = v \wedge$
$s.\text{NumEsami} = \text{pre}(s.\text{NumEsami}) + 1 \wedge$
$\text{this.NumVerb} = \text{pre}(\text{this.NumVerb}) + 1$

FineSpecifica

Specifica mediante una notazione formale (2)

InizioSpecificaClasse Studente

Iscrizione(c: CorsoDiLaurea)

pre : $\neg(\exists\ c2 \mid \langle this,c2\rangle \in iscritto)$

post : $iscritto = pre(iscritto) \cup \{\langle this,c\rangle\}$

MediaVoti() : real

pre : $(\exists\ c \mid c \in CorsoDiLaurea \wedge \langle this,c\rangle \in iscritto) \wedge$
$this.NumEsami > 0$

post : definiamo Voti come l'insieme
$\{\langle c,v\rangle \in int \mid \exists\ p \mid p \in Professore \wedge c \in Corso \wedge$
$\langle this,p,c\rangle \in Esame \wedge Esame.voto\ (this,p,c) = v\}$

$$result = \frac{\sum_{\langle c,v\rangle \in Voti} v}{this.NumEsami}$$

FineSpecifica

Esercizio

Scrivere la specifica delle seguenti operazioni
usando una notazione formale:

- Operazione NumeroStud() della classe CorsoDiLaurea

Soluzione

InizioSpecificaClasse CorsoDiLaurea

NumeroStud():int

pre : true

post : result = card({ s | s ∈ Studente ∧ <s,this> ∈ iscritto})

FineSpecifica