

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1

Analisi dei Requisiti
Unified Modeling
Language

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1
Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi
e degli oggetti

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1.1

Analisi dei Requisiti

Unified Modeling Language

Diagrammi UML delle classi e degli
oggetti

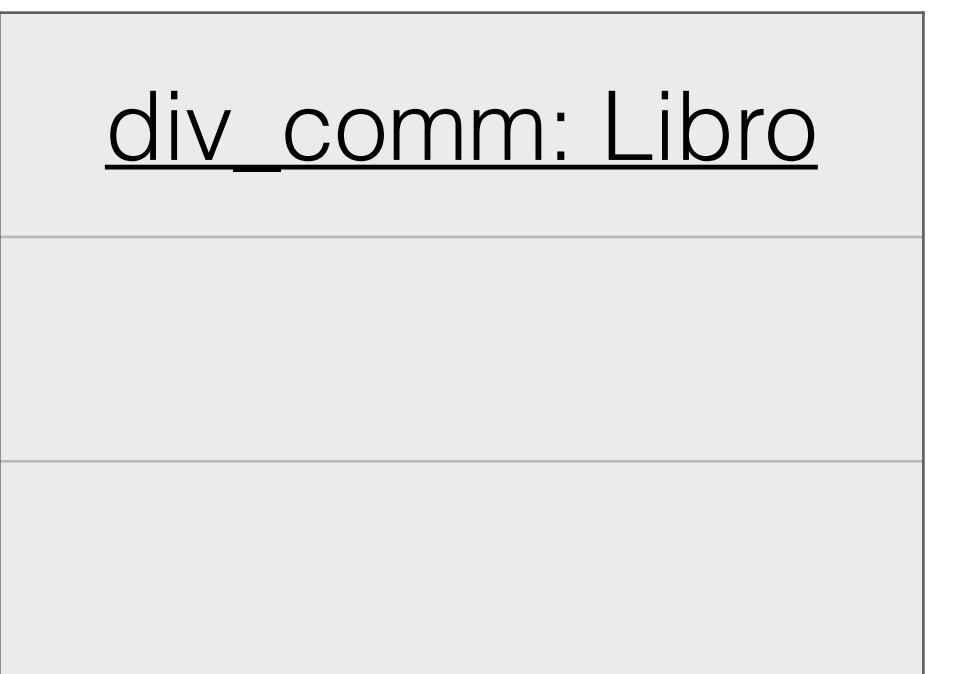
Oggetti e classi

Diagramma delle classi e degli oggetti per l'analisi

- Nella fase di analisi ci si concentra sulle classi più che sugli oggetti
- Gli oggetti servono essenzialmente per descrivere elementi singoli particolarmente significativi oppure per descrivere esempi
- Approccio simile al paradigma della programmazione object-oriented, ad es. in Java:
 - Un programma si scrive definendo un insieme definito di class(i), non di oggetti
 - Gli oggetti vengono creati, modificati e distrutti durante l'esecuzione del programma
- Come detto in precedenza, faremo riferimento solo ad un sottoinsieme dei meccanismi previsti in UML per descrivere il diagramma delle classi

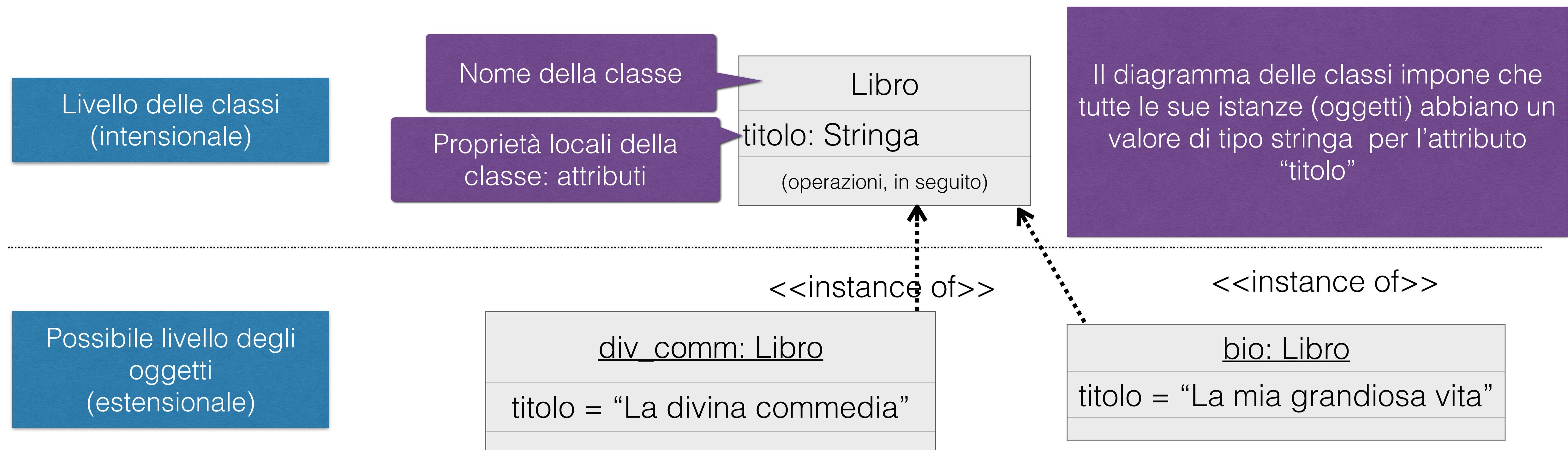
Oggetti in UML

- 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)
- div_comm è l'identificatore di oggetto (scelto dall'analista per potersi riferire all'oggetto nello schema concettuale)
- Libro è la classe più specifica di cui l'oggetto è istanza
- Si noti la sottolineatura



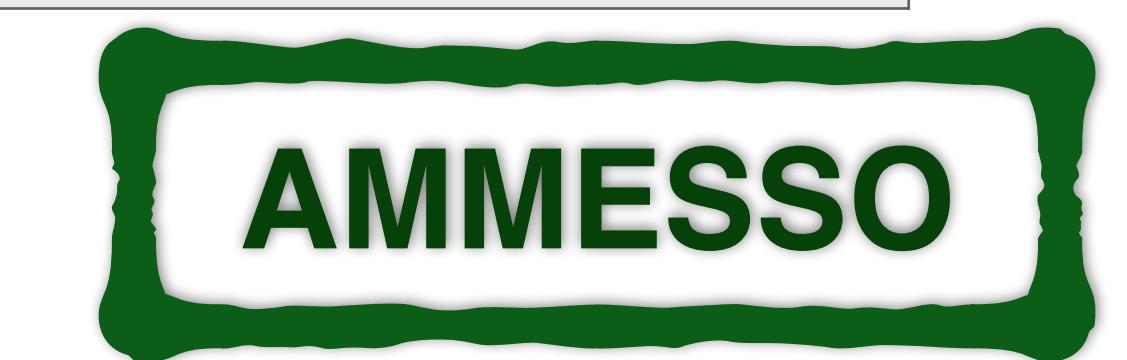
Classi

- Una classe modella un insieme di oggetti omogenei (le istanze della classe) ai quali sono associate proprietà statiche (attributi) e dinamiche (operazioni, le vedremo in seguito).
- Ogni classe è descritta da:
 - un nome
 - un insieme di proprietà (astrazioni delle proprietà comuni degli oggetti che sono istanze delle classi)



Oggetti e classi

- Un oggetto (istanza di una classe) è identificato da un identificatore univoco
- Un diagramma delle classi in generale permette la coesistenza di oggetti identici



Oggetti identici possono coesistere

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione
UNITÀ: Progettazione.1

Prof. Toni Mancini
Dipartimento di Informatica
Sapienza Università di Roma

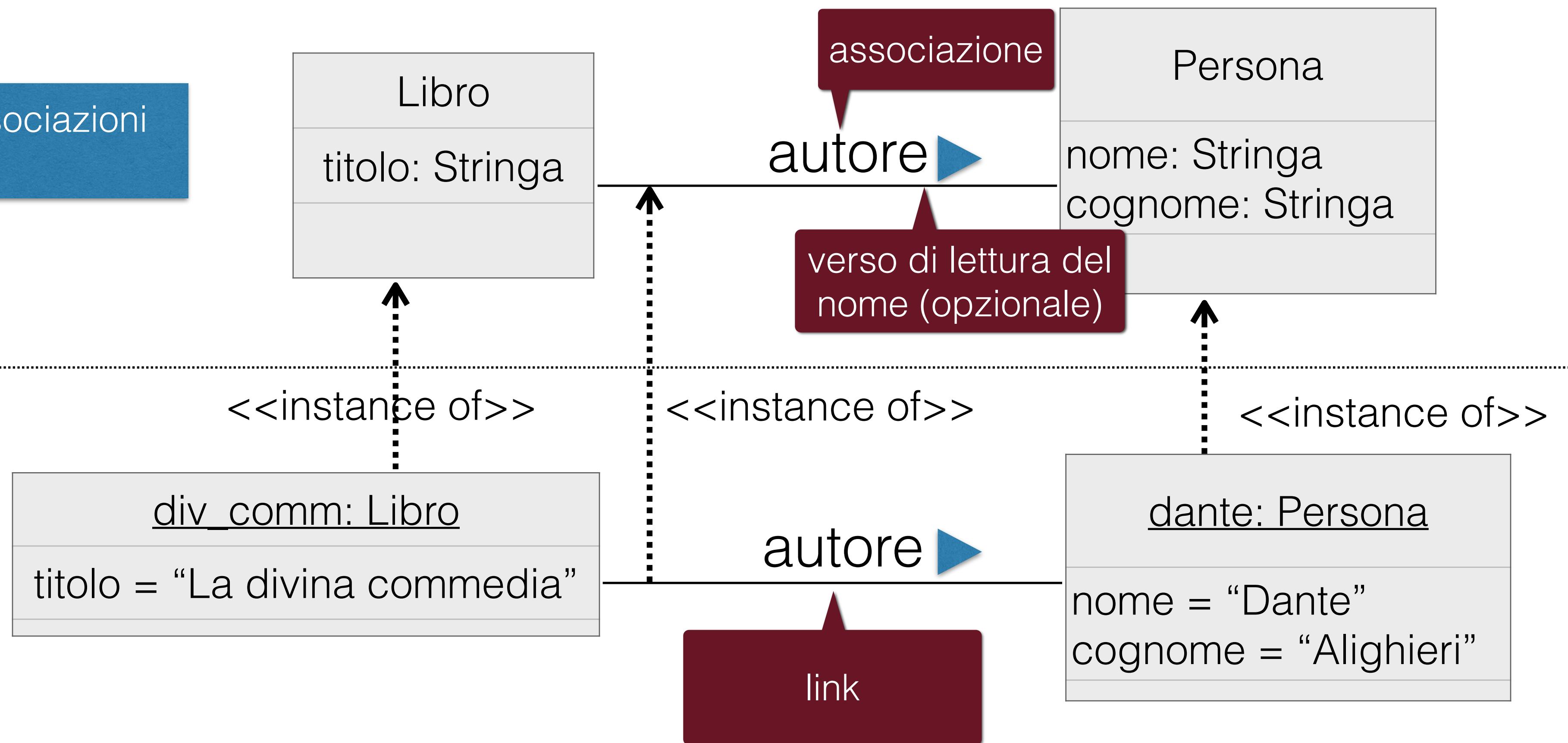


A.1.1.2
Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi e degli oggetti
Link e associazioni

Associazioni e link

- Una associazione modella la possibilità che oggetti di due (o più classi) abbiano dei legami
- Le istanze di associazioni si chiamano link: se A è una associazione tra le classi C1 e C2, una istanza di A è un link tra due oggetti (in altre parole, una coppia), uno della classe C1 e l'altro della classe C2

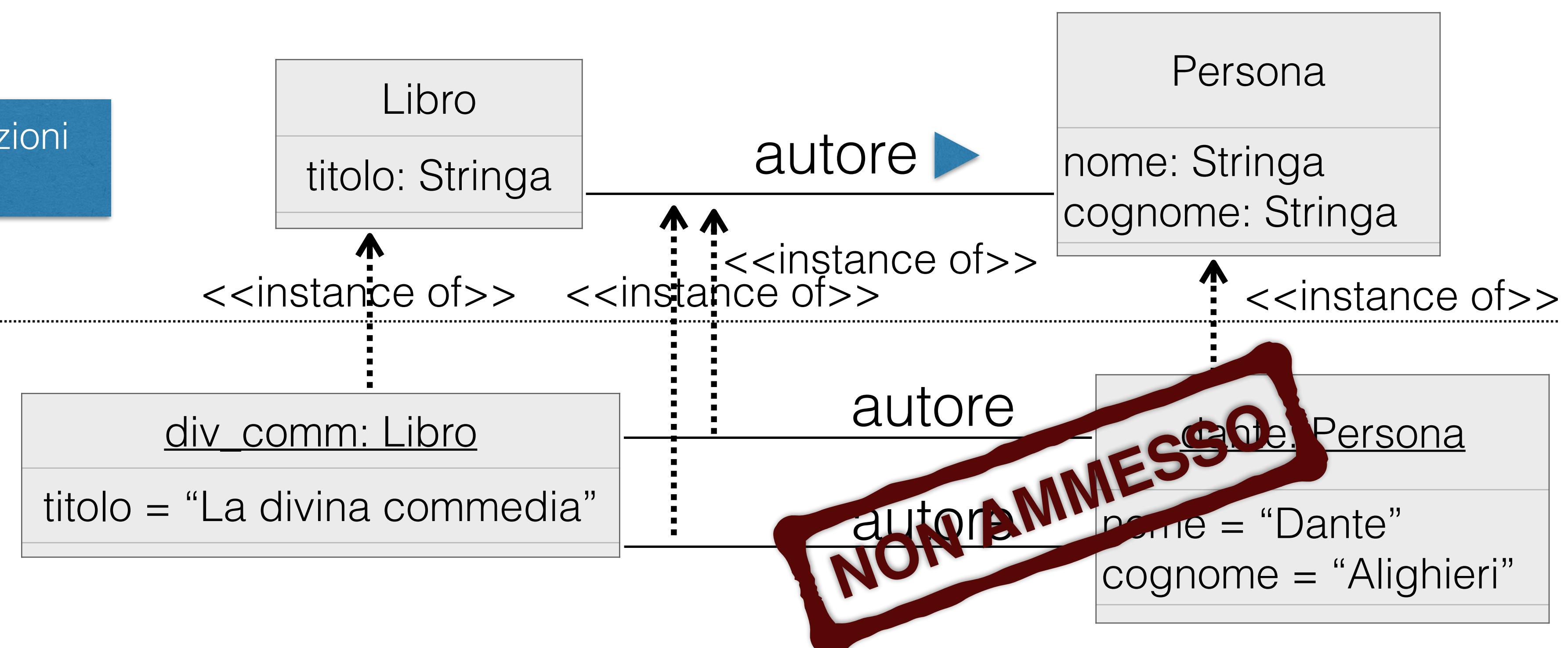
Livello delle classi e delle associazioni
(intensionale)



Associazioni e link (2)

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

Livello delle classi e delle associazioni (intensionale)

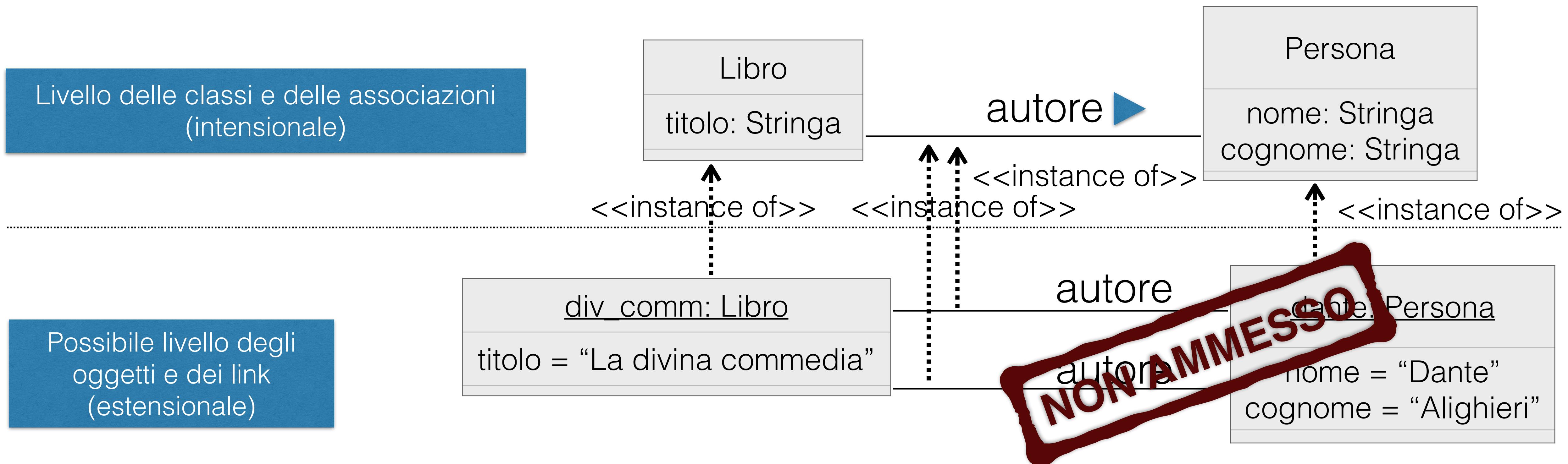


Possibile livello degli oggetti e dei link (estensionale)

Non ammesso: link uguali **non** possono coesistere

Associazioni e link (3)

- ...il seguente diagramma degli oggetti non è ammesso dal diagramma delle classi
 - Una associazione tra le classi Libro e Persona definisce la **possibilità** che le istanze (oggetti) della classe Libro siano in relazione con istanze (oggetti) di classe Persona (la relazione si chiama “autore”).
 - Aver modellato la possibilità di tali legami mediante una associazione implica che, per l’analista, **non ha concettualmente senso** il fatto che un libro abbia “più volte lo stesso autore”.



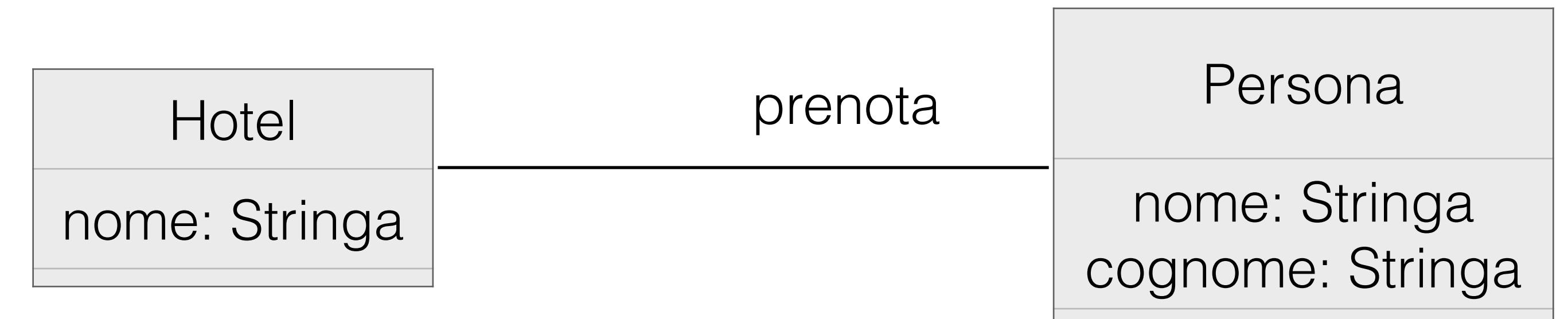
Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

Esempio

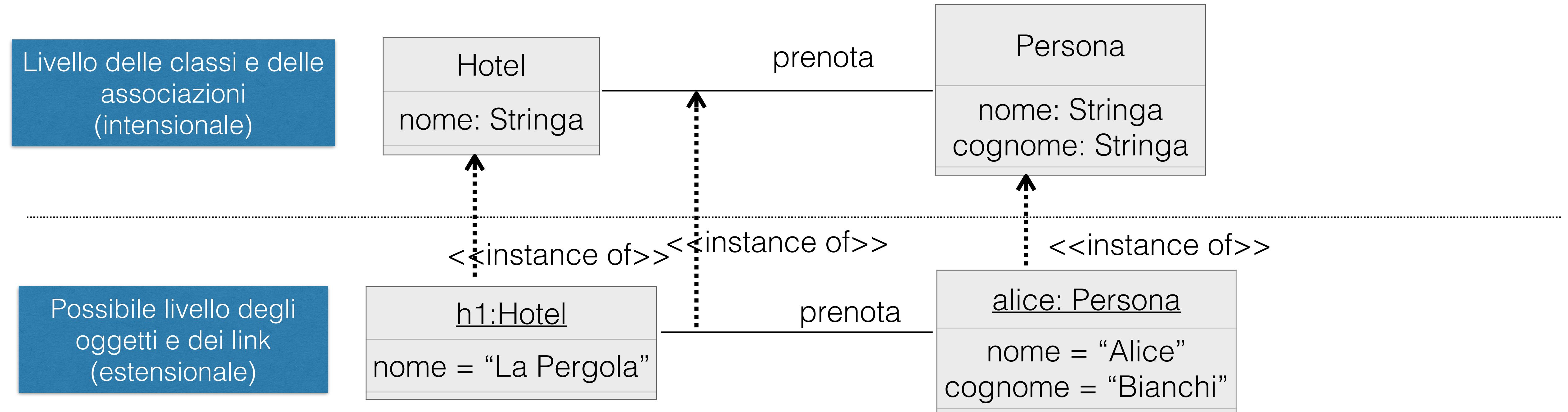
Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

Livello delle classi e delle associazioni (intensionale)



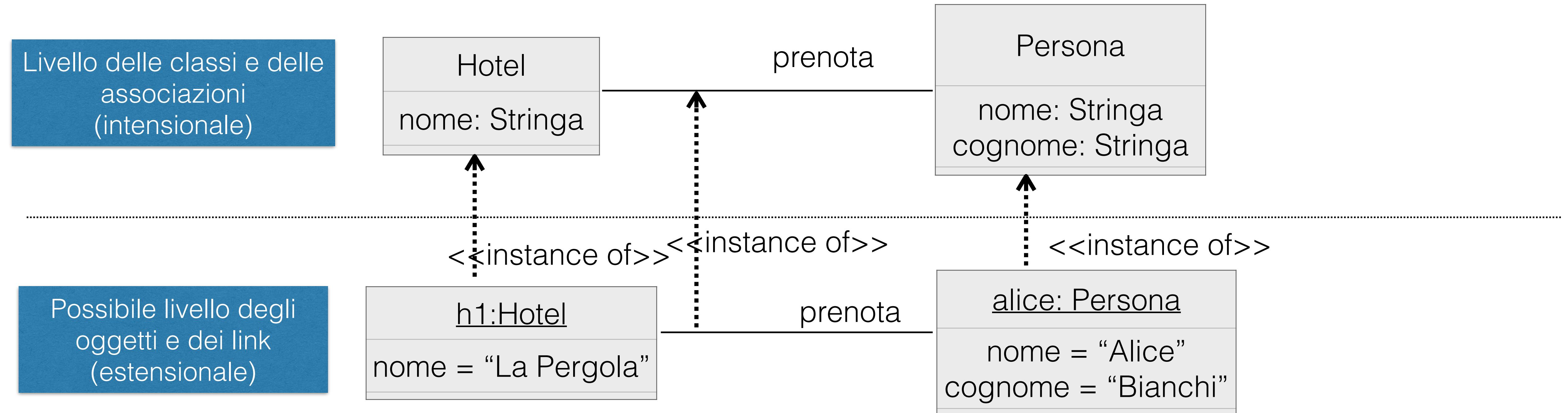
Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



Esempio

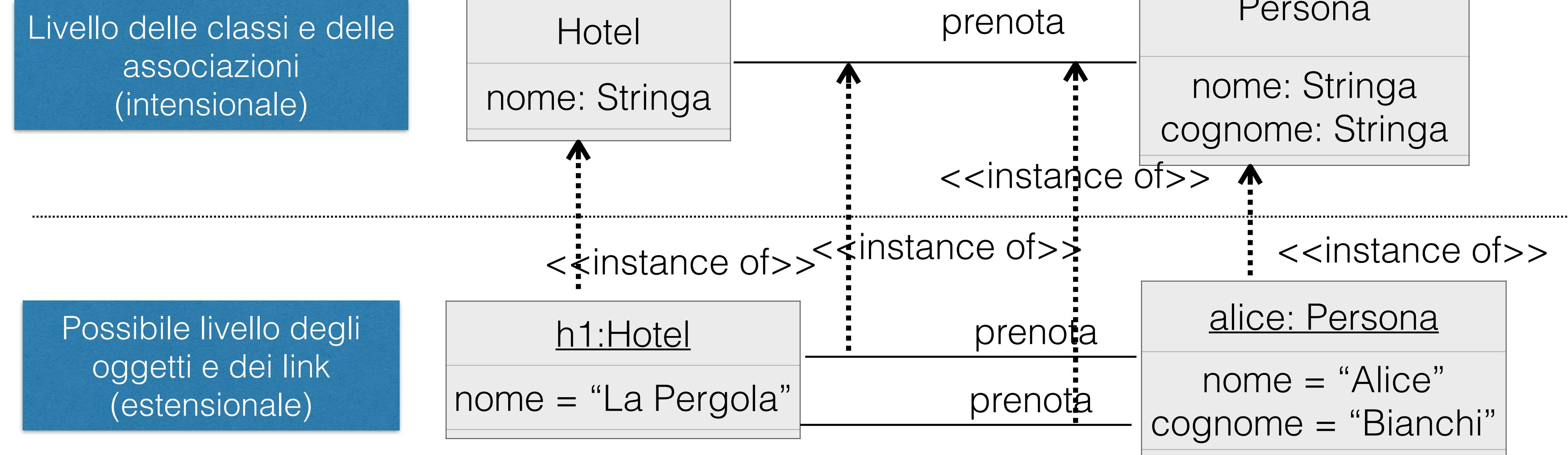
Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



- E se volessimo rappresentare una seconda prenotazione di ‘alice’ presso ‘h1’?

Esempio

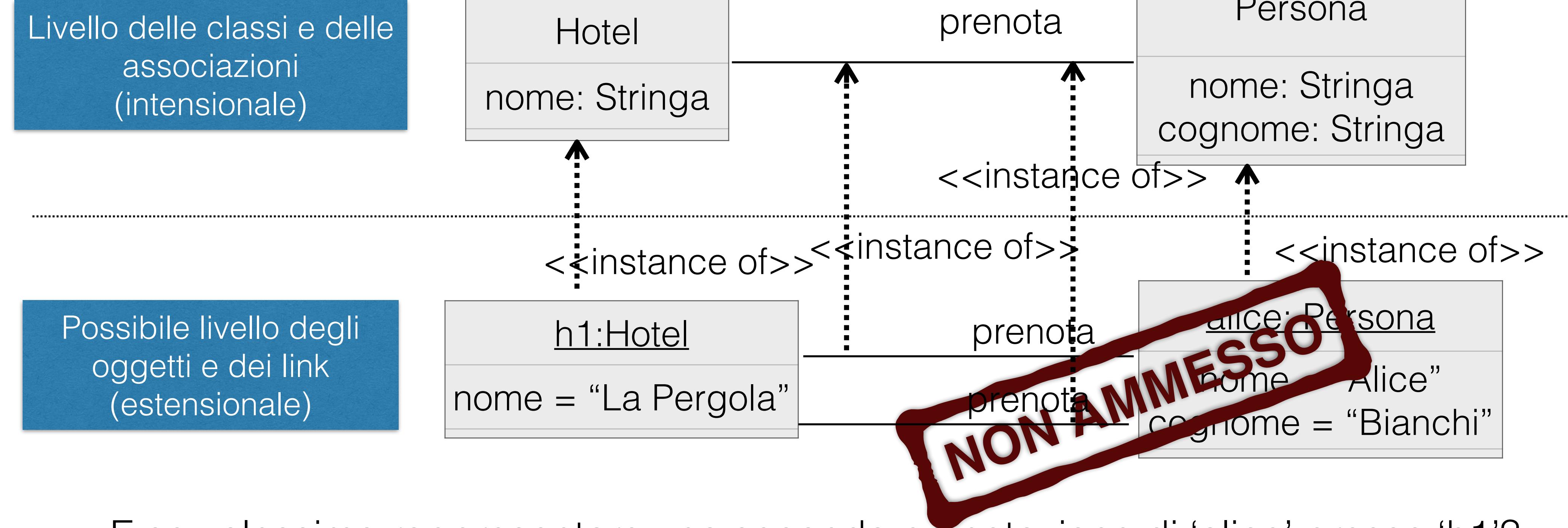
Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web



- E se volessimo rappresentare una seconda prenotazione di ‘alice’ presso ‘h1’?

Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

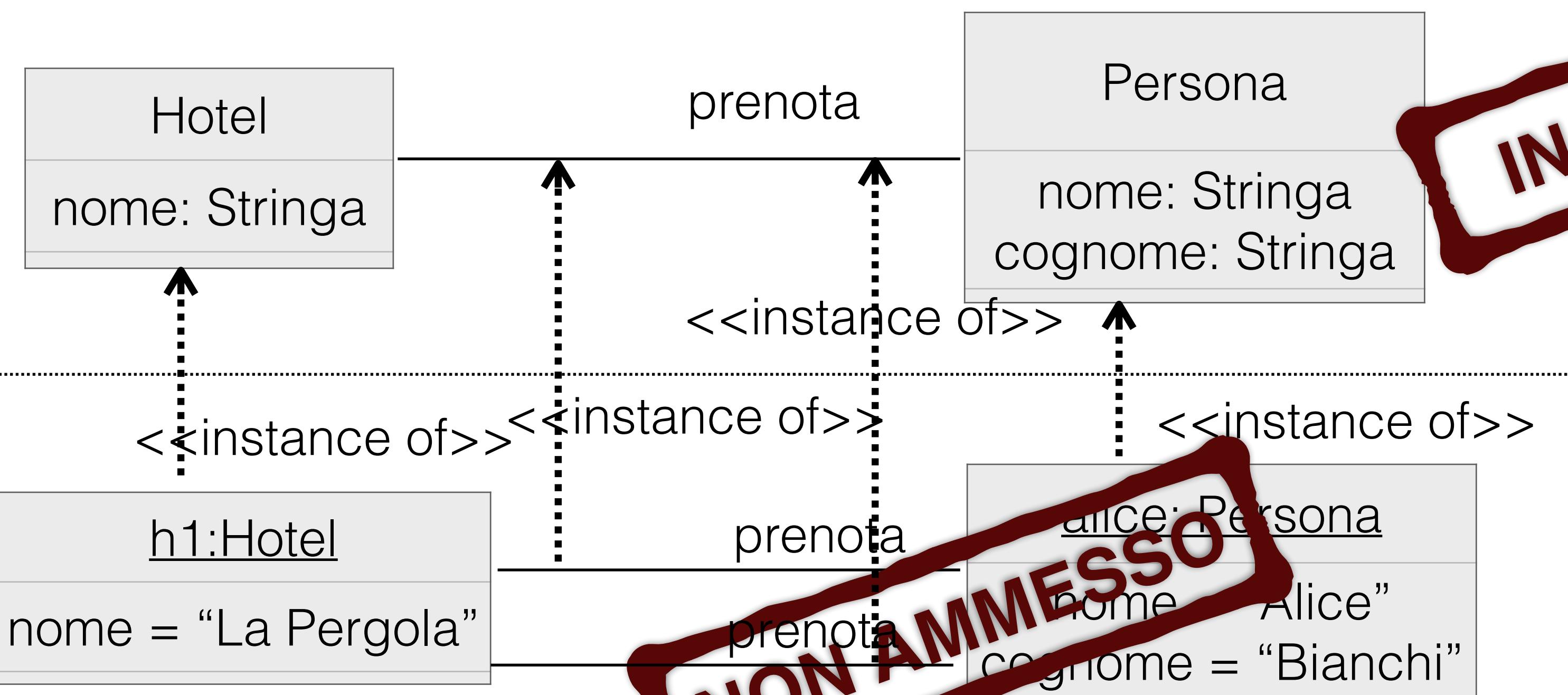


- E se volessimo rappresentare una seconda prenotazione di ‘alice’ presso ‘h1’?

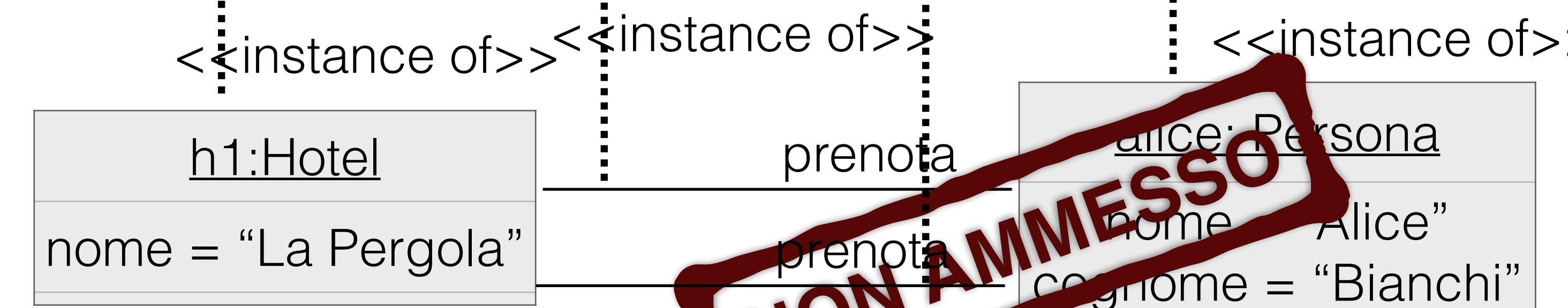
Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)

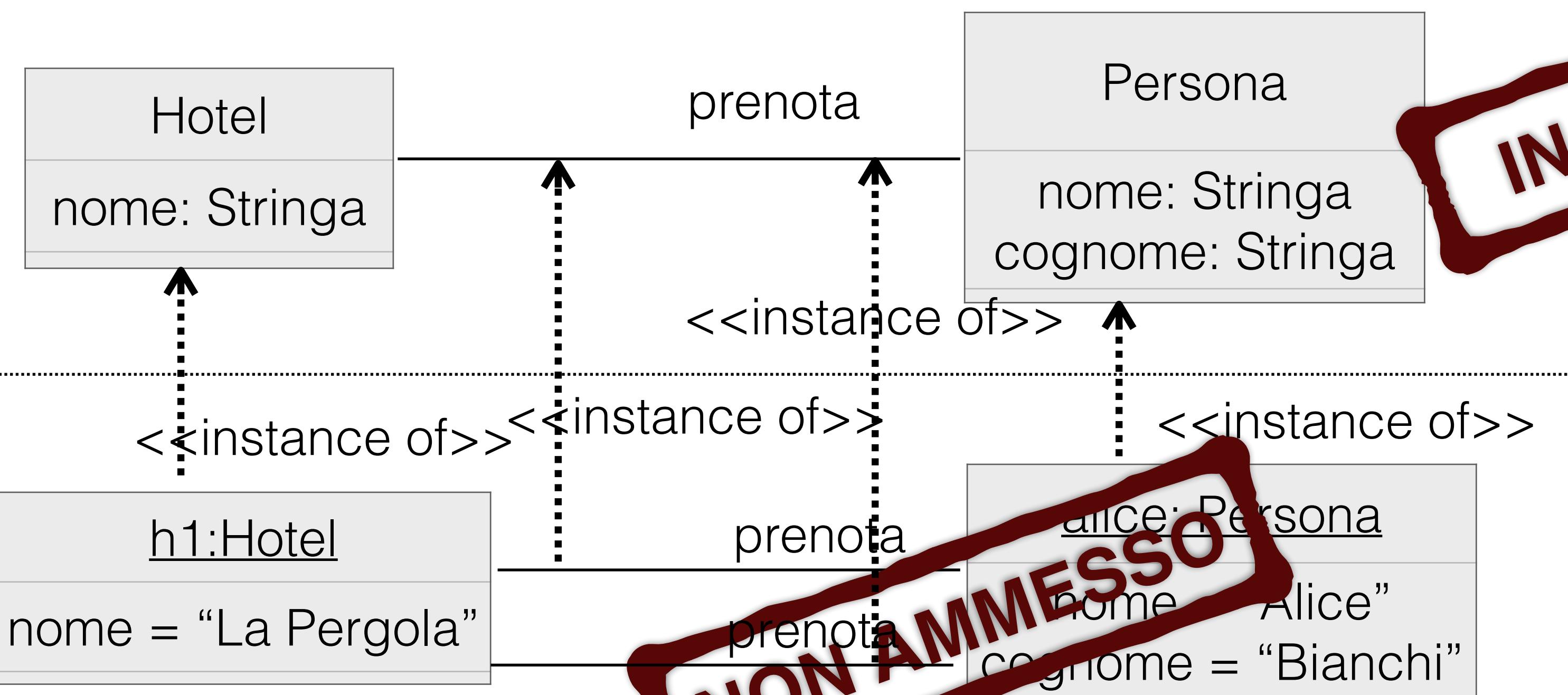


- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?
- Il diagramma qui sopra impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!

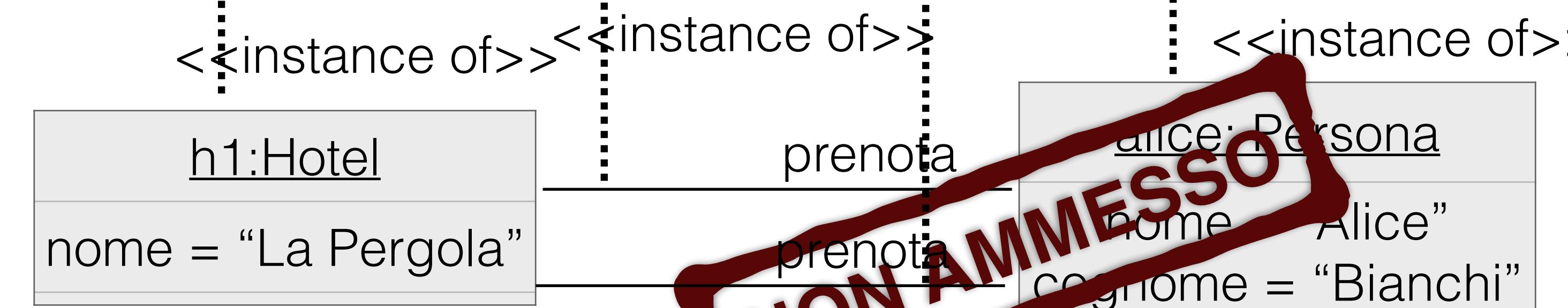
Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



- E se volessimo rappresentare una seconda prenotazione di 'alice' presso 'h1'?
- Il diagramma qui sopra impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!
- Come risolvere?

Esempio

Si vuole progettare un'applicazione che permetta ai clienti di prenotare hotel via web

- Il diagramma precedente impedirebbe ad una stessa persona di prenotare, nella sua vita, lo stesso hotel più volte!
- Come risolvere?

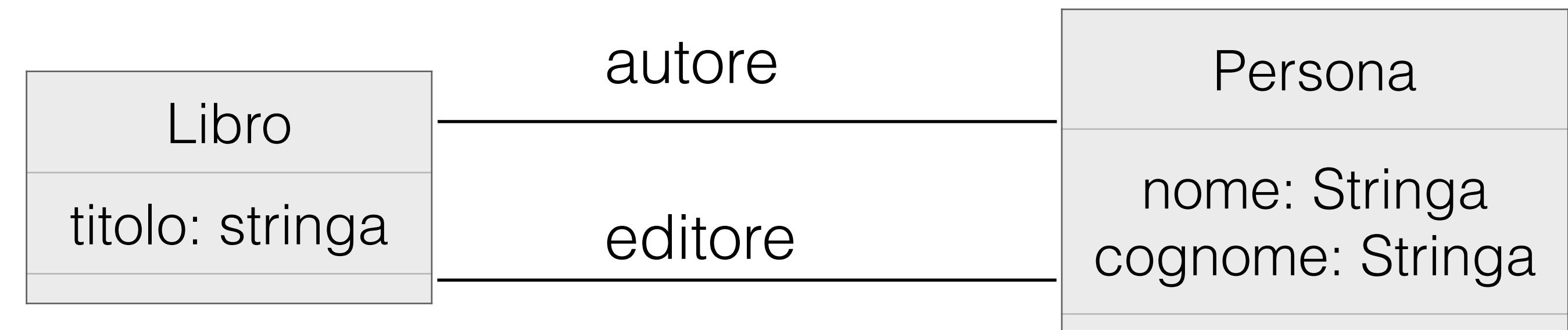


- Abbiamo fatto in modo che ogni singola prenotazione abbia “vita propria”
—> le prenotazioni sono a loro volta oggetti, istanze della nuova classe Prenotazione

Associazioni multiple tra due classi

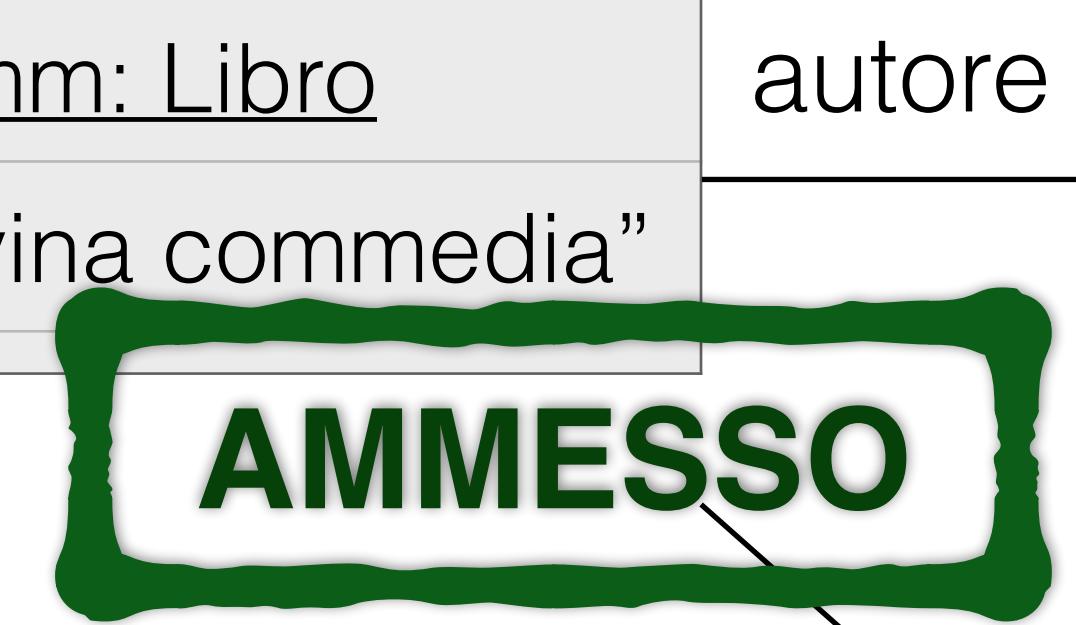
- Tra le stesse classi possono essere definite più associazioni, che modellano **legami di natura diversa**

Livello delle classi e delle associazioni
(intensionale)



.....

div_comm: Libro
titolo = “La divina commedia”



dante: Persona
nome = “Dante”
cognome = “Alighieri”

feltr: Persona
nome = “Faustino”
cognome = “Feltrinelli”

Possibile livello degli
oggetti e dei link
(estensionale)

AMMESSO

editore

bio: Libro
titolo = “La mia grandiosa vita”

autores
modestino: Persona
nome = “Modestino”
cognome = “Rossi”

editores

AMMESSO

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1.3

Analisi dei Requisiti

Unified Modeling Language

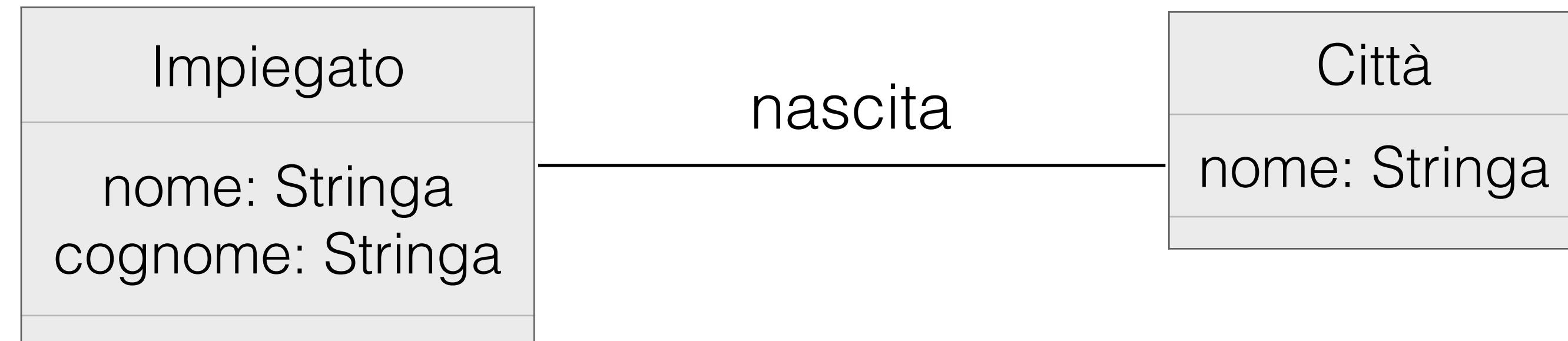
Diagrammi UML delle classi e degli oggetti

Vincoli di molteplicità sulle
associazioni e sugli attributi

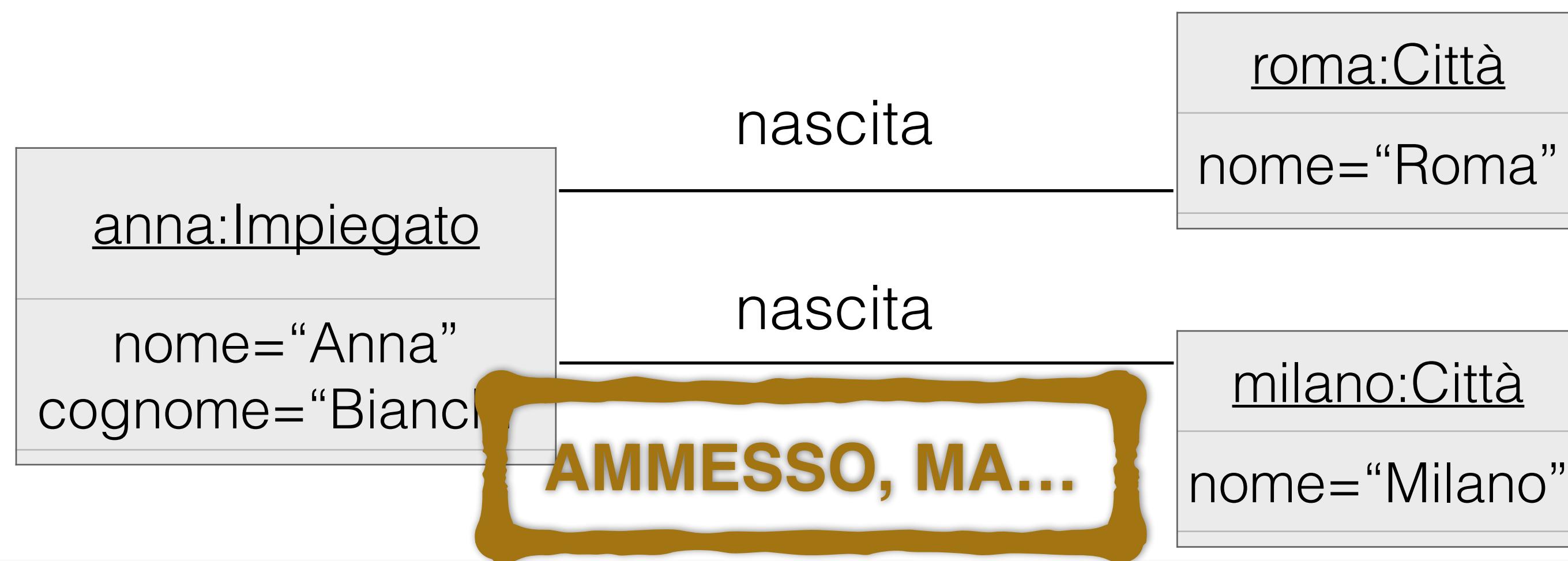
Associazioni: vincoli di molteplicità

- I diagrammi delle classi visti fino ad ora sono modelli **molto laschi** della realtà di interesse

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



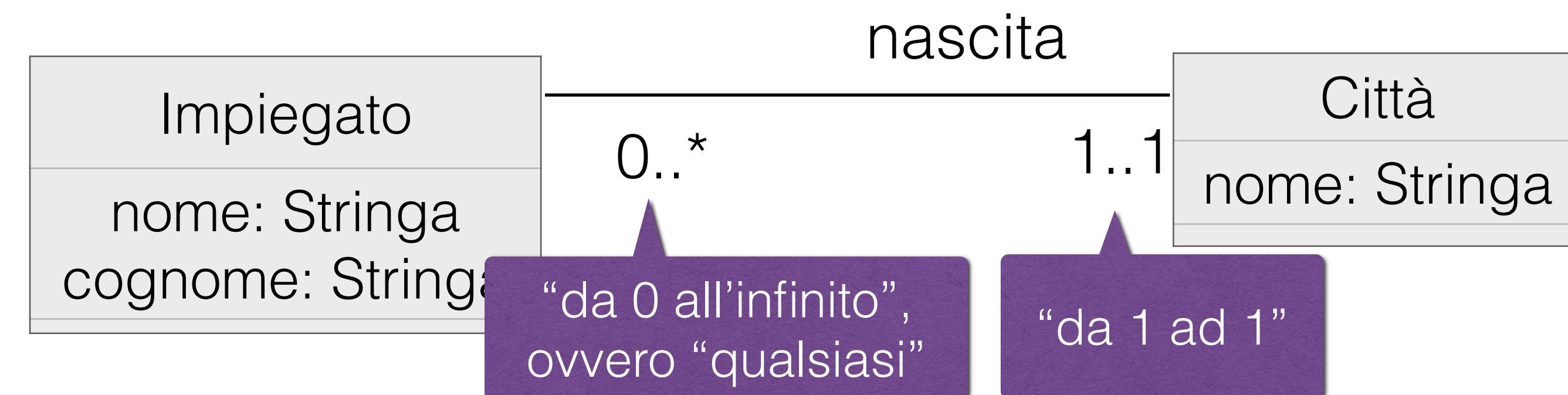
AMMESSO, MA...

Questo insieme di oggetti è ammesso dal diagramma delle classi di sopra, ma **non** soddisfa i vincoli del mondo reale

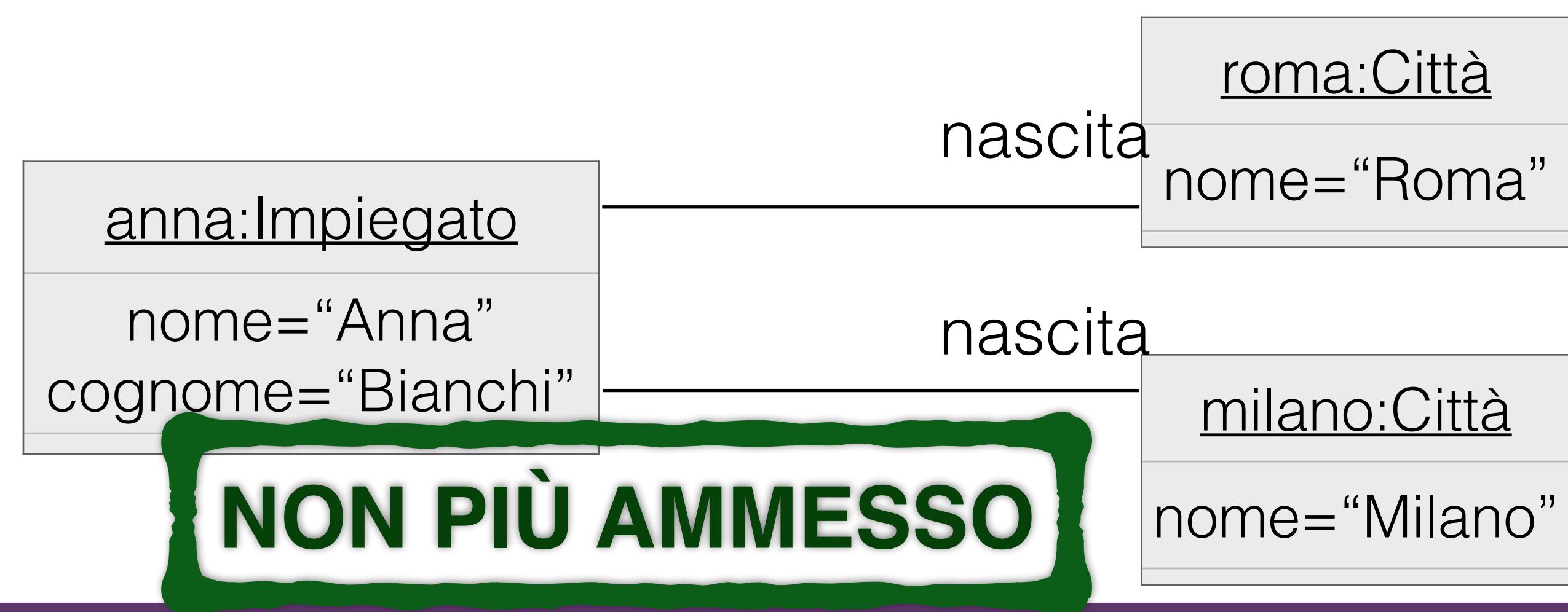
Associazioni: vincoli di molteplicità (2)

- UML permette di definire **vincoli di integrità in un diagramma delle classi**
- Un vincolo di integrità impone ulteriori restrizioni (oltre quelle strutturali imposte dal diagramma) sui livelli estensionali ammessi
- Vediamo ora i **vincoli di molteplicità sulle associazioni**

Livello delle classi
(intensionale)



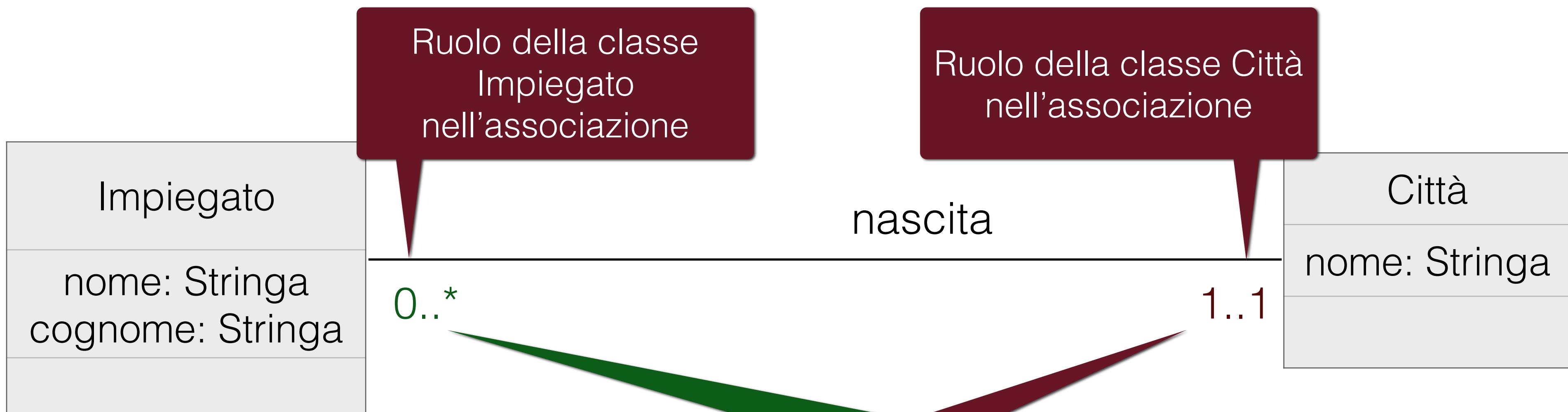
Possibile livello degli
oggetti e dei link
(estensionale)



A causa dei vincoli di
molteplicità, ora questo
insieme di oggetti e link
non è più ammesso dal
diagramma delle classi
di sopra

Associazioni: vincoli di molteplicità (3)

- Semantica dei vincoli di molteplicità sui ruoli delle associazioni

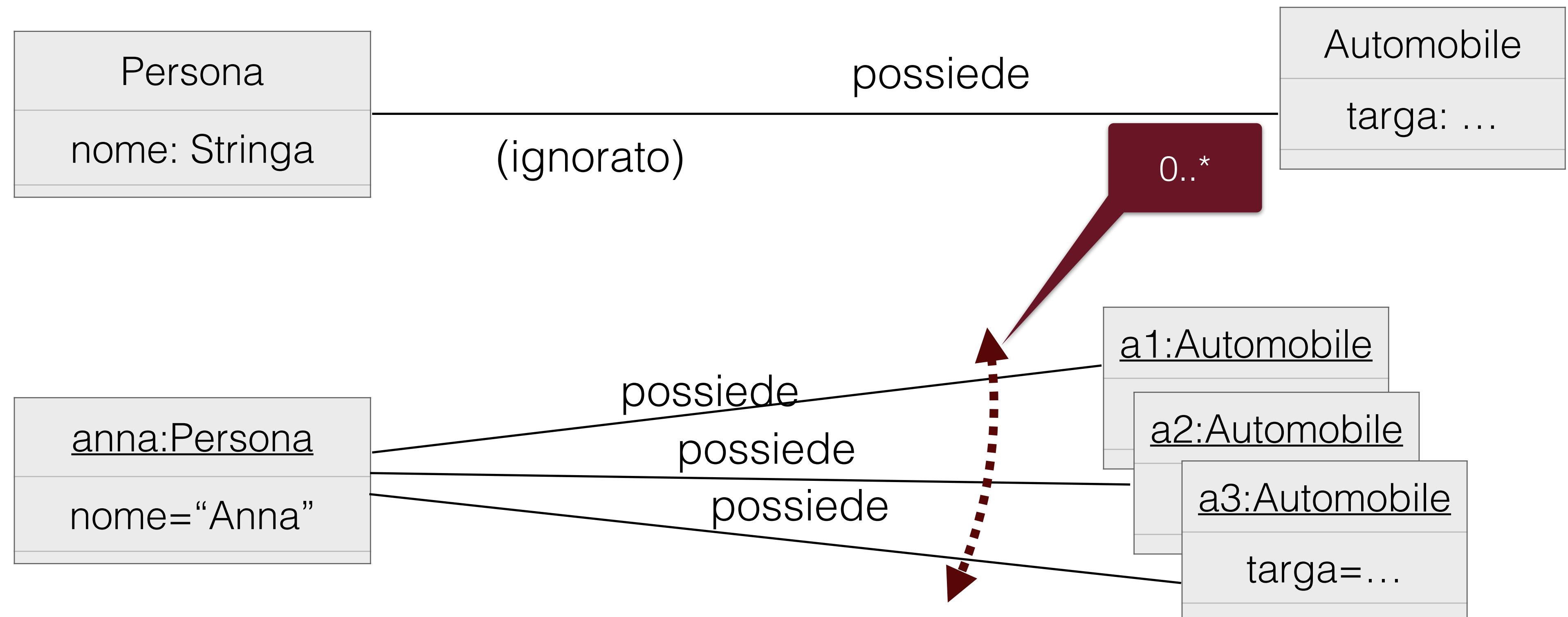


1..1: Ogni istanza di **Impiegato** deve essere coinvolta in un numero di link dell'associazione “**nascita**” che va “da 1 ad 1”
 Dato che non possiamo avere più link tra la stessa coppia di oggetti, questo è equivalente a: ogni istanza di **Impiegato** deve essere legata ad una ed una sola istanza di **Città** (tramite link dell'associazione “**nascita**”)

0..*: Ogni istanza di **Città** deve essere coinvolta in un numero di link dell'associazione “**nascita**” che va “da 0 all’infinito”
 È equivalente a dire: ogni istanza di **Città** può essere legata ad un numero qualunque (**0..***) di istanze di **Impiegato** (tramite link dell'associazione “**nascita**”)

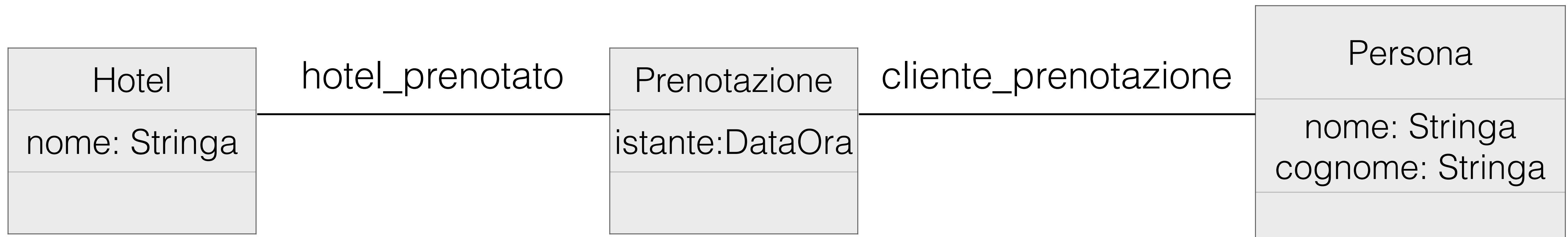
Associazioni: vincoli di molteplicità (4)

- Semantica dei vincoli di molteplicità sui ruoli delle associazioni (cont.)



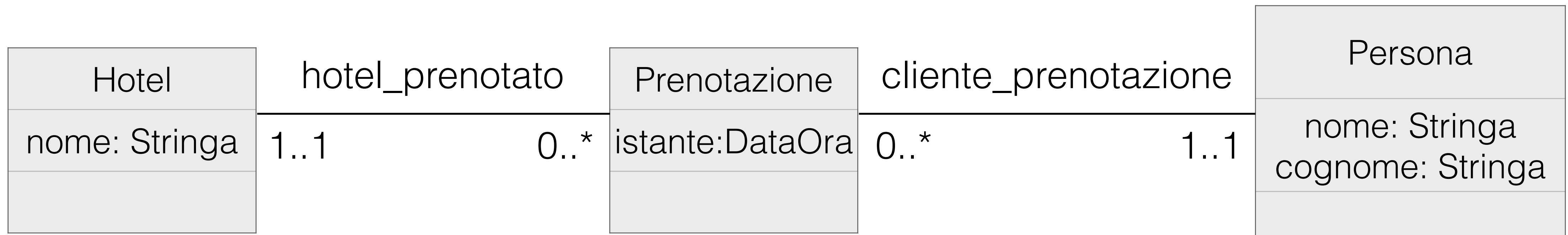
Esercizio: prenotazioni di hotel

- Definire i vincoli di molteplicità sui ruoli delle associazioni del seguente diagramma delle classi



Esercizio: prenotazioni di hotel

- Definire i vincoli di molteplicità sui ruoli delle associazioni del seguente diagramma delle classi
- Soluzione:

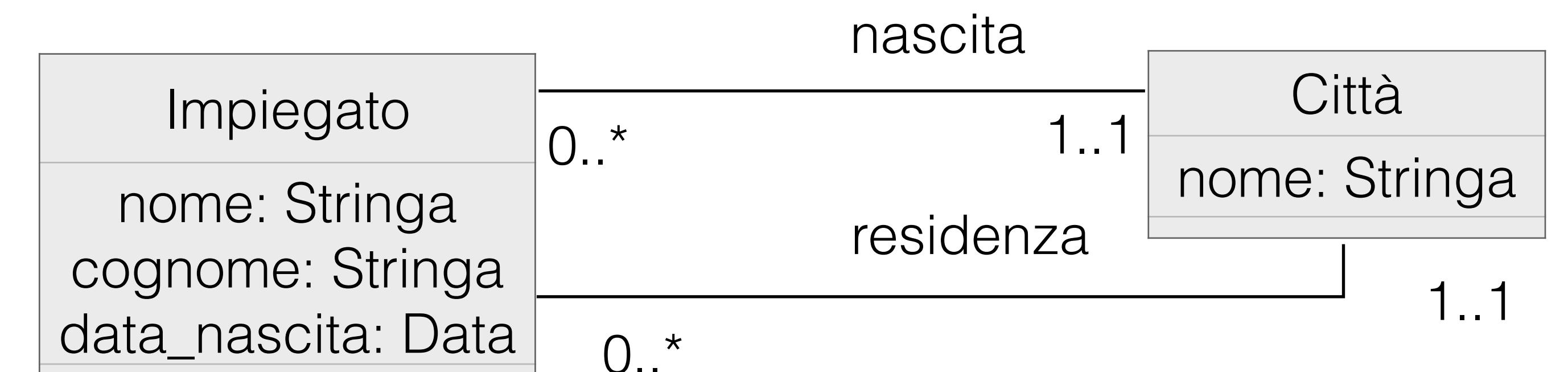


Esercizio

- Si vuole progettare un sistema che gestisca i dati anagrafici delle persone
- Di ogni persona interessa il nome, il cognome, la data di nascita, la città di nascita e quella di residenza
- Si definisca un diagramma delle classi concettuale per l'applicazione

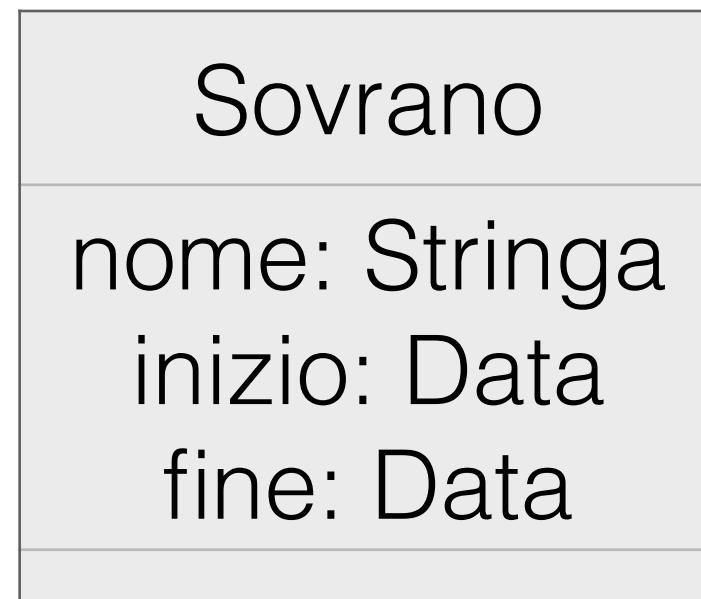
Esercizio

- Si vuole progettare un sistema che gestisca i dati anagrafici delle persone
- Di ogni persona interessa il nome, il cognome, la data di nascita, la città di nascita e quella di residenza
- Si definisca un diagramma delle classi concettuale per l'applicazione



Associazioni che insistono più volte sulla stessa classe

- Supponiamo di voler modellare i sovrani di un regno ormai scomparso
- Di ogni sovrano interessa il nome, il periodo in cui ha regnato, ed il predecessore



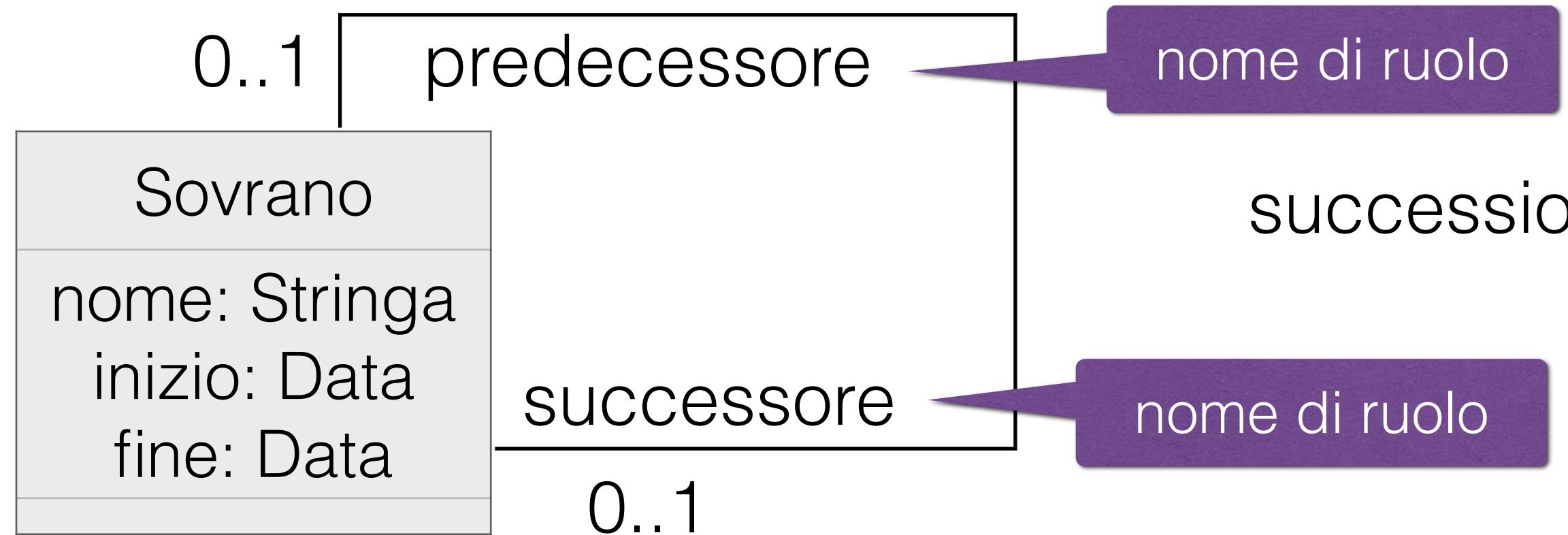
- Come possiamo rappresentare il concetto di **predecessore**?

Associazioni che insistono più volte sulla stessa classe (2)

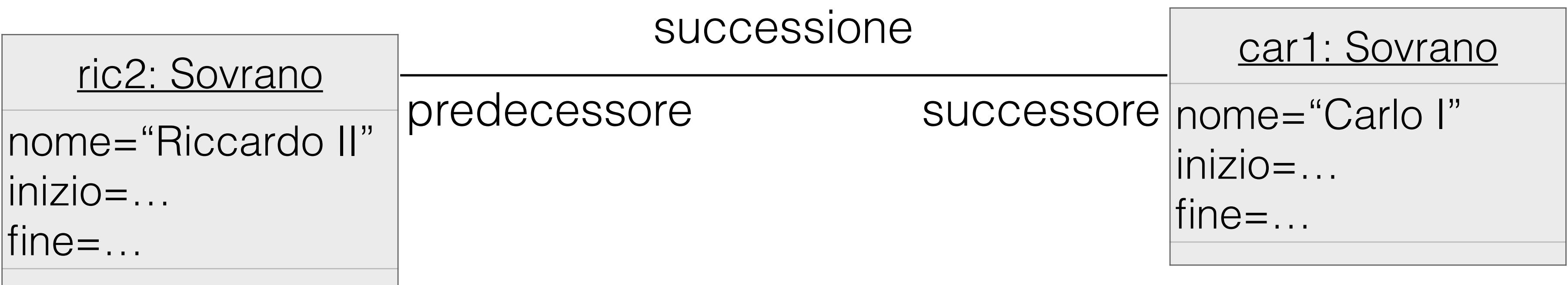
- Supponiamo di voler modellare i sovrani di un regno ormai scomparso
- Di ogni sovrano interessa il nome, il periodo in cui ha regnato, ed il predecessore

Livello delle classi e delle associazioni (intensionale)

- Sappiamo che le istanze dell'associazione sono coppie (s1, s2) di oggetti (istanze) di classe Sovrano.
- La classe Sovrano gioca due ruoli nell'associazione → UML ci obbliga a dare esplicitamente nomi distinti ai due ruoli
- Una istanza dell'associazione diventa una **coppia etichettata**: (predecessore:s1, successore:s2)
- L'ambiguità è sparita



Possibile livello degli oggetti e dei link (estensionale)



ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma

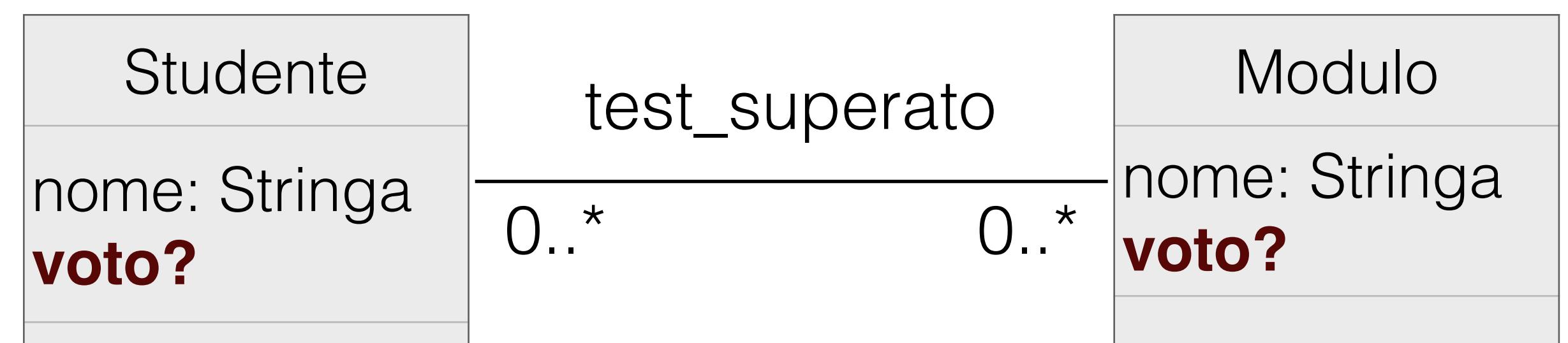


A.1.1.4

Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi e degli oggetti
Associazioni con attributi
(association class)

Associazioni con attributi

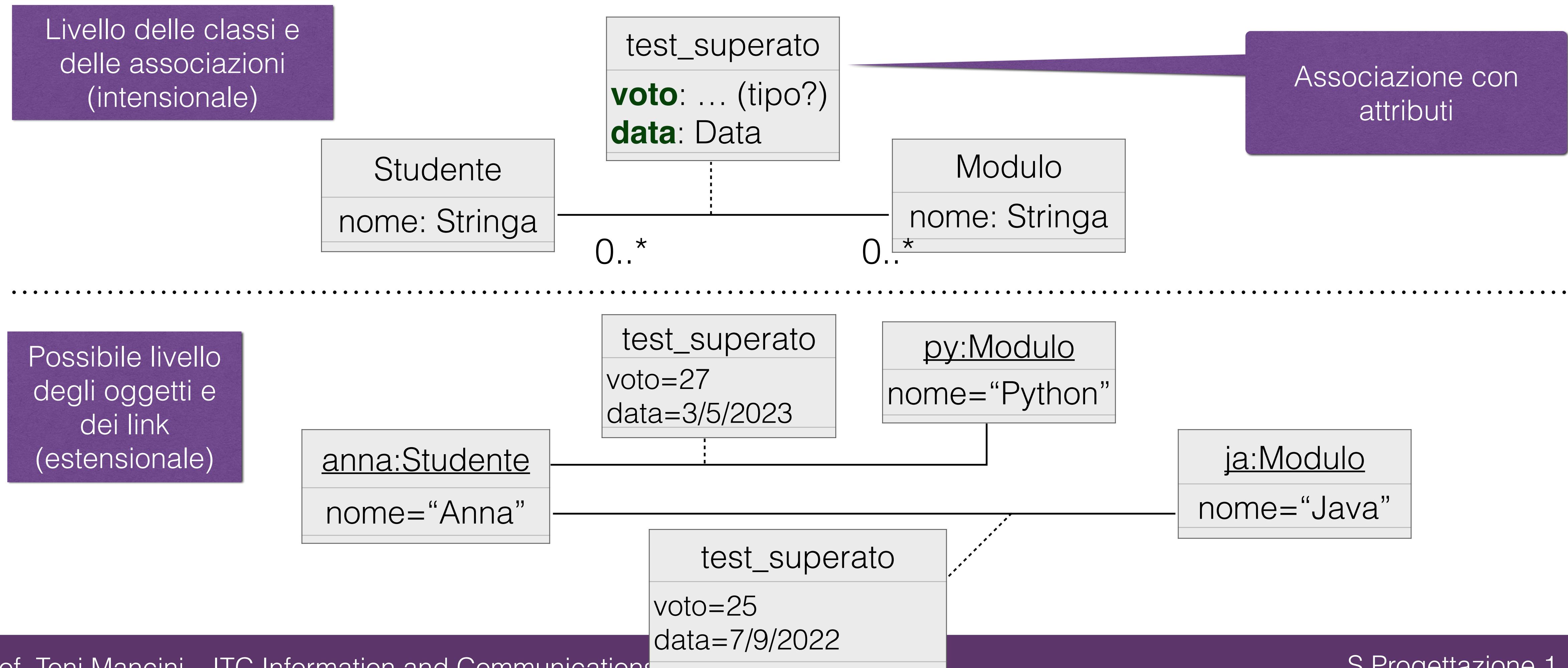
- Si vuole progettare un sistema che gestisca gli esiti (voti in trentesimi) dei test superati dagli studenti di un corso.
- Il corso è diviso in moduli e uno studente può superare il test di ogni modulo al più una volta.
- Si definisca un diagramma delle classi concettuale per l'applicazione



- Come rappresentare i voti conseguiti dagli studenti nei test dei diversi moduli?
- Non possiamo aggiungere un attributo “voto” né nella classe Studente né nella classe Modulo!
- In effetti, un voto non è una proprietà locale di uno studente, né di un modulo, ma è una proprietà del legame tra uno studente ed un modulo... ovvero è una **proprietà dell’associazione**

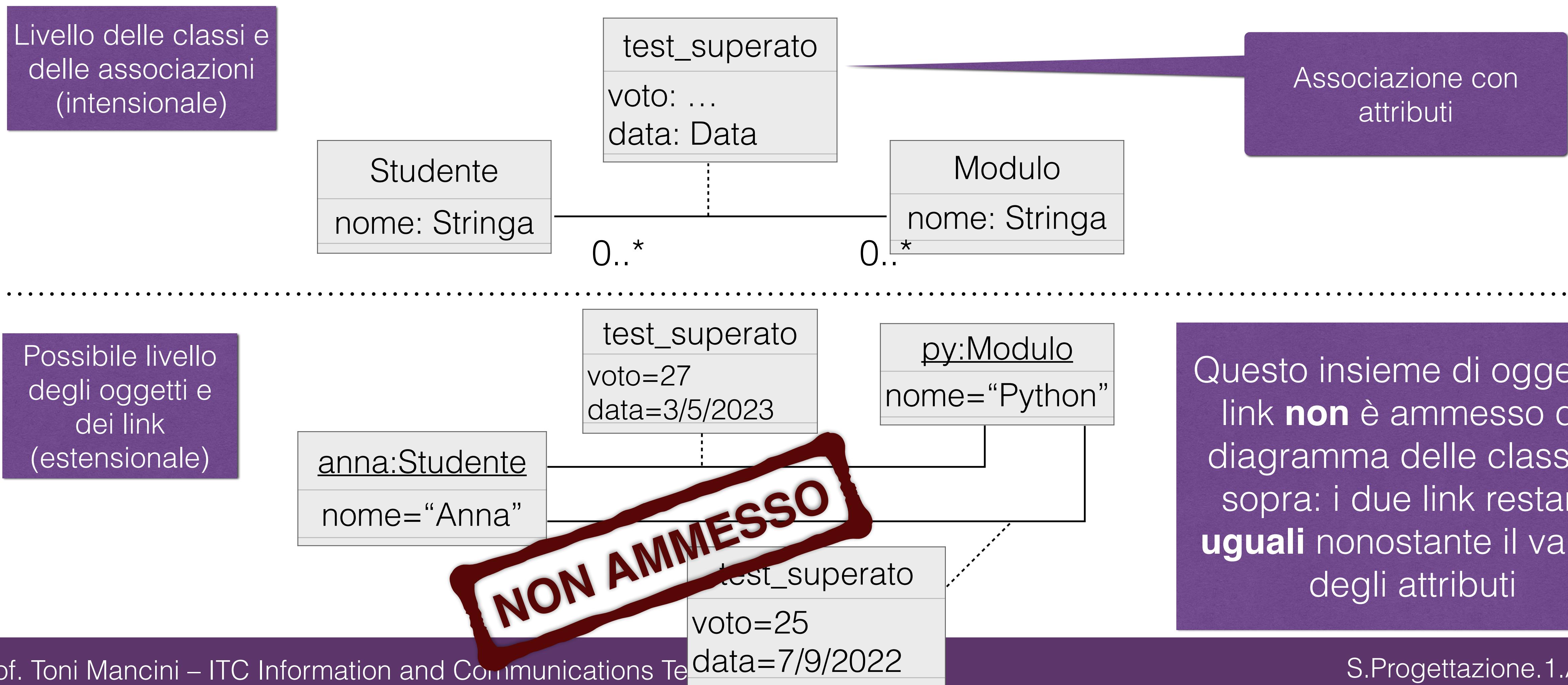
Associazioni con attributi (2)

- Si vuole progettare un sistema che gestisca gli esiti (data e voto in trentesimi) dei test svolti dagli studenti di un corso.
- Il corso è diviso in moduli e uno studente può sostenere il test di ogni modulo al più una volta.



Associazioni con attributi (3)

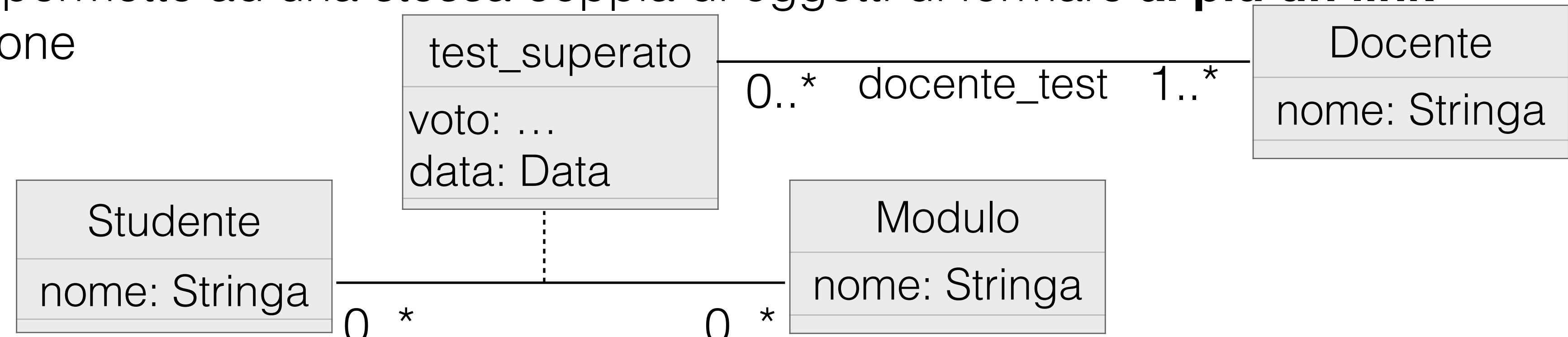
- **Attenzione:** anche in presenza di attributi, **la natura dell'associazione non cambia:**
 - il diagramma permette ad una stessa coppia di oggetti di formare **al più un link** dell'associazione



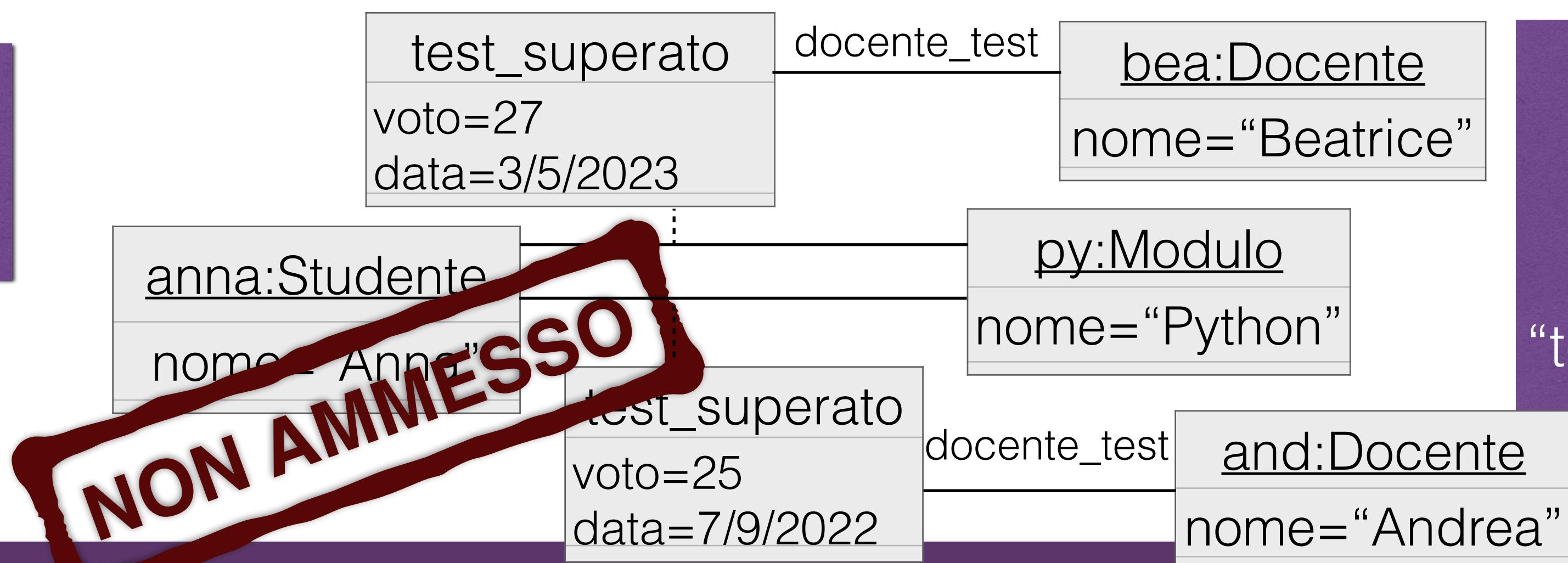
Associazioni con attributi, anche dette association class

- Un'associazione UML con attributi è anche chiamata **association class**, in quanto può essere a sua volta terminale di ulteriori associazioni
- **La natura dell'associazione non cambia:**
 - il diaogramma permette ad una stessa coppia di oggetti di formare **al più un link**

Livello delle classi e
delle associazioni
(intensionale)



Possibile livello
degli oggetti e
dei link
(estensionale)



Questo insieme di oggetti e
link **non** è ammesso dal
diagramma delle classi di
sopra: i due link
“test_superato” restano **uguali**
nonostante il valore degli
attributi e i “loro” link

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1.5

Analisi dei Requisiti

Unified Modeling Language

Diagrammi UML delle classi e degli
oggetti

Tipi di dato concettuali

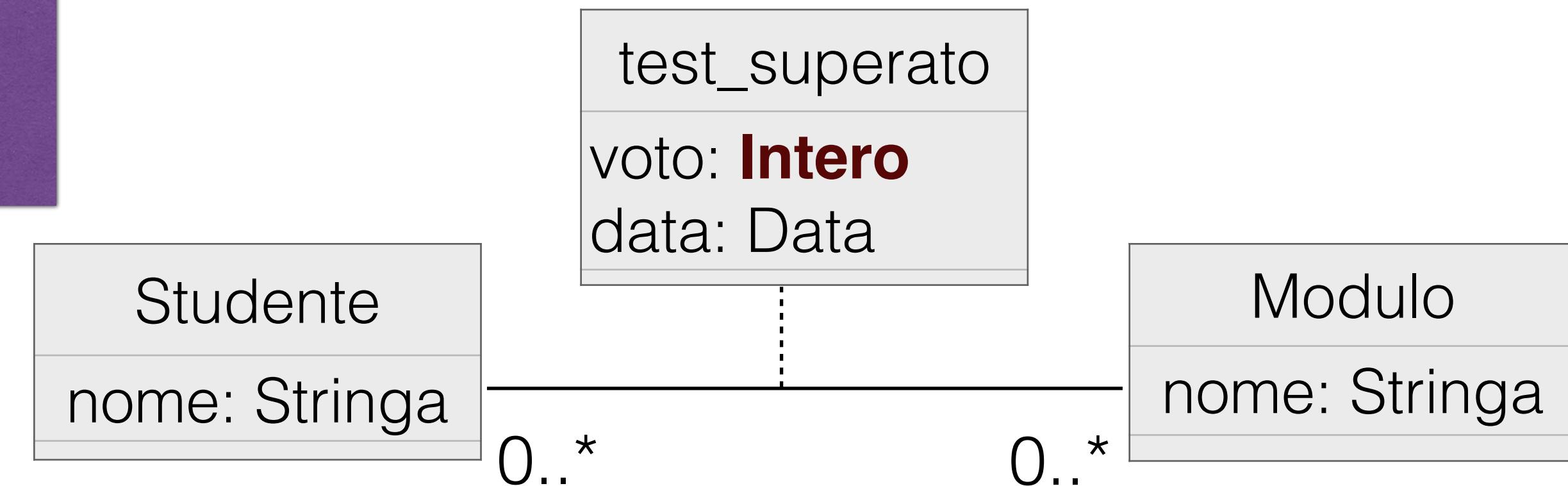
Tipi di dato concettuali: tipi base

- Durante l'analisi concettuale dobbiamo definire il tipo di ogni attributo, di classe e di associazione
- Ricordiamoci che, in analisi, non vogliamo effettuare scelte tecnologiche (ad es., sul linguaggio di programmazione)
- Dunque, vogliamo utilizzare **tipi di dato concettuali**, che siano facilmente realizzabili con qualsivoglia tecnologia informatica (Python, Java, sistemi di gestione di basi di dati, etc.)
- **Tipi base:**
 - Intero, Reale, Booleano, Data, Ora, DataOra
 - Vogliamo però anche essere certi di modellare la realtà in modo accurato!

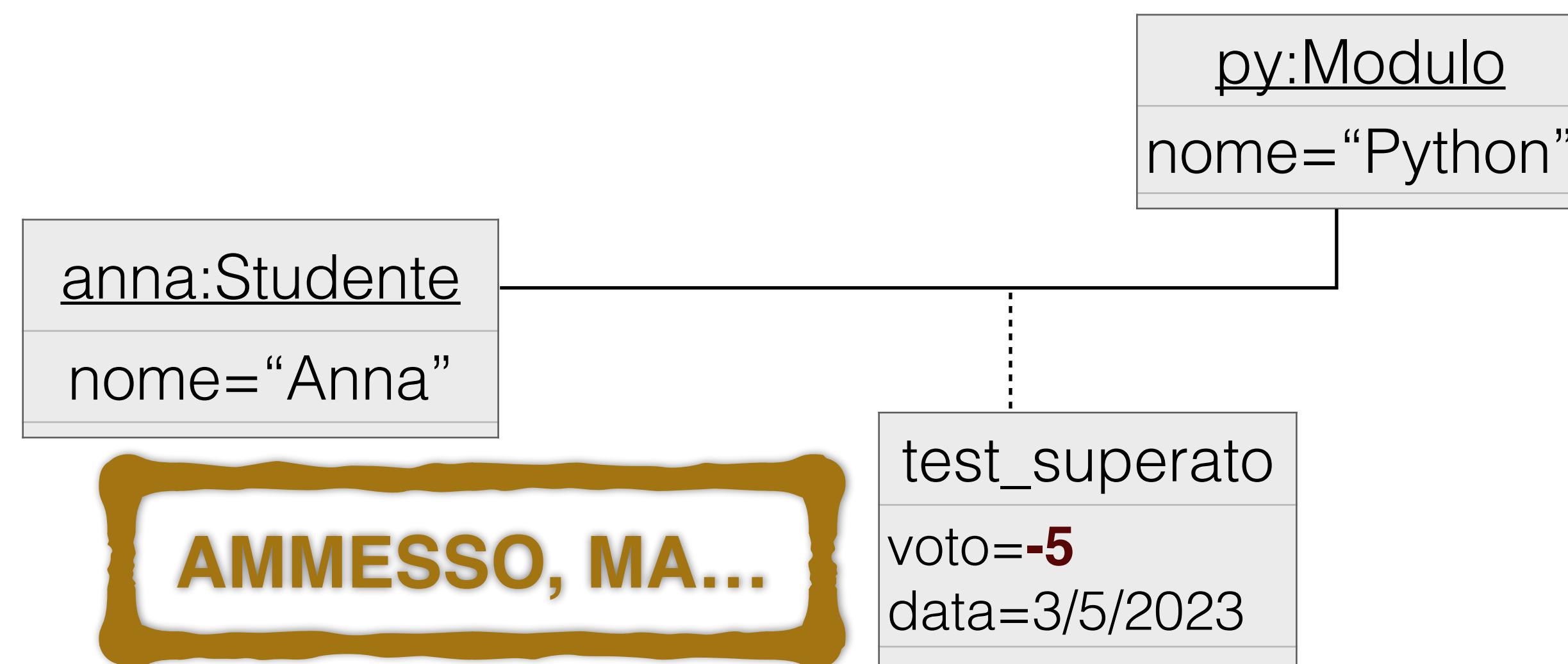
Tipi di dato concettuali: tipi specializzati

- ...Vogliamo però anche essere certi di modellare la realtà in modo accurato!

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



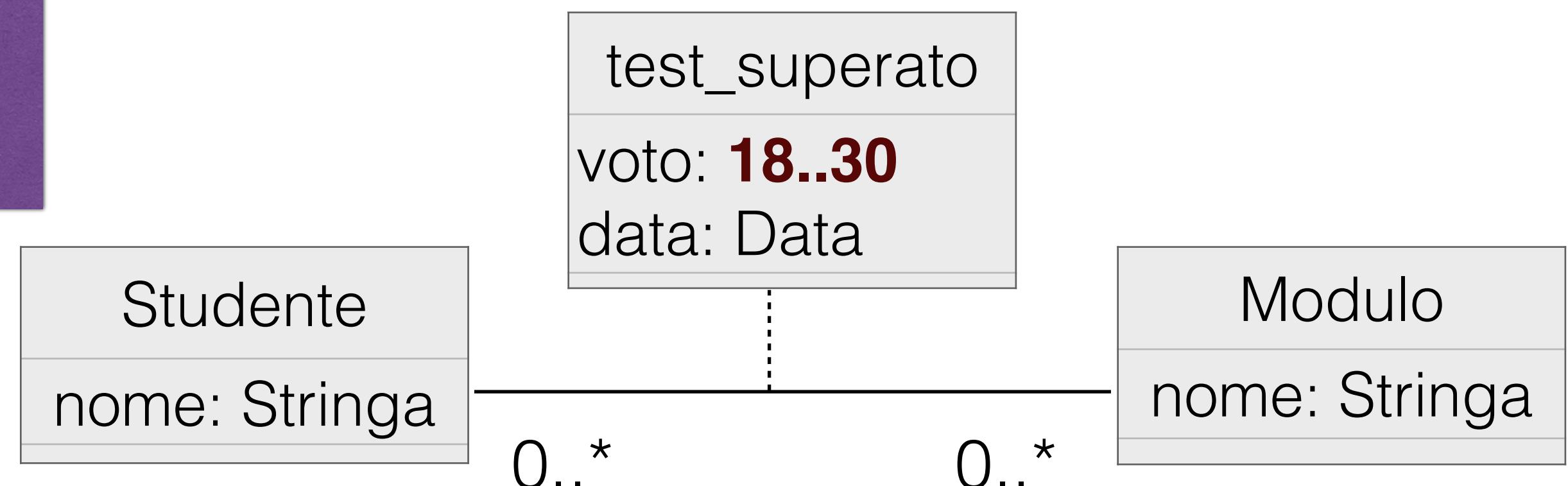
AMMESSO, MA...

Il valore -5 per il voto di 'anna' nel test del modulo 'py' sarebbe lecito per il diagramma!

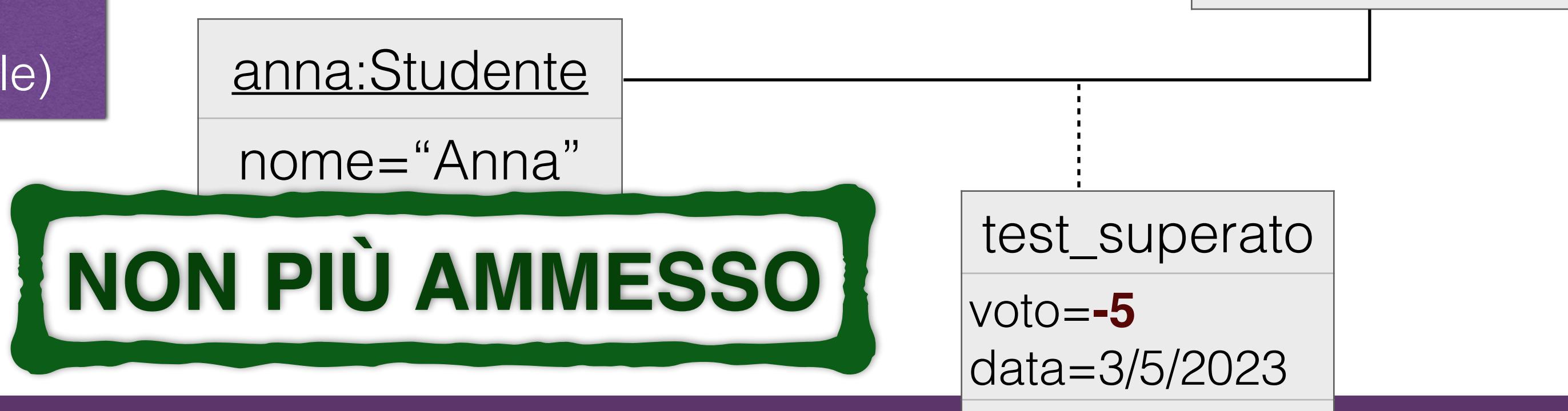
Tipi di dato concettuali: tipi specializzati (2)

- Vogliamo però anche essere certi di modellare la realtà in modo accurato!
- Tipi di dato specializzati:
 - Intero > 0, Reale <= 0, etc., tipo intervallo (di interi): 18..30, 0..100, etc.

Livello delle classi e delle associazioni (intensionale)



Possibile livello degli oggetti e dei link (estensionale)



NON PIÙ AMMESSO

Il valore -5 per il voto di anna nel test del modulo Python non è più lecito!

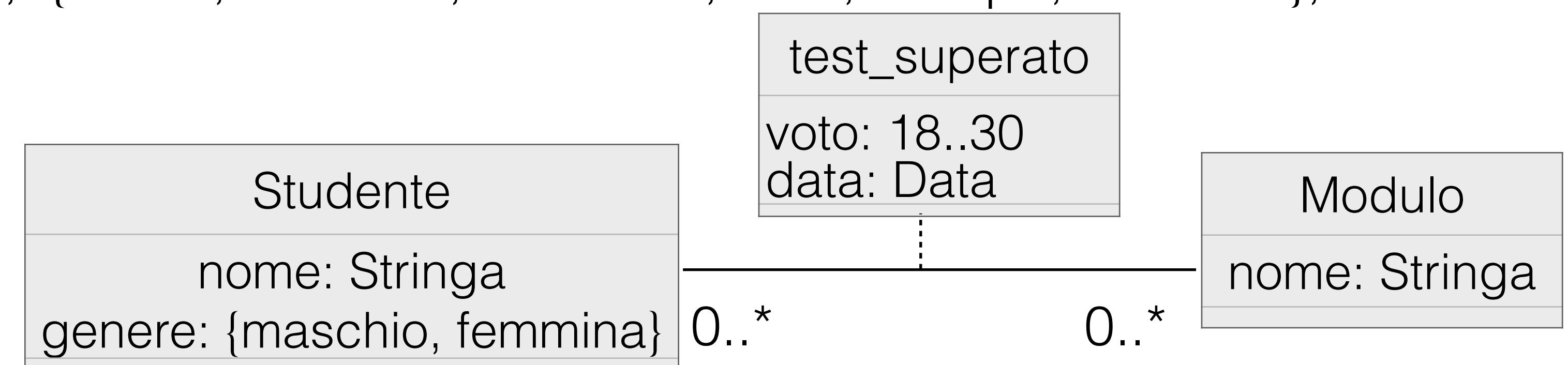
Tipi di dato concettuali: tipi enumerativi

- Quando l'insieme dei possibili valori di un attributo è finito e piccolo, possiamo usare un **tipo di dato enumerativo**, che definisce esplicitamente e completamente l'insieme dei valori possibili per l'attributo.

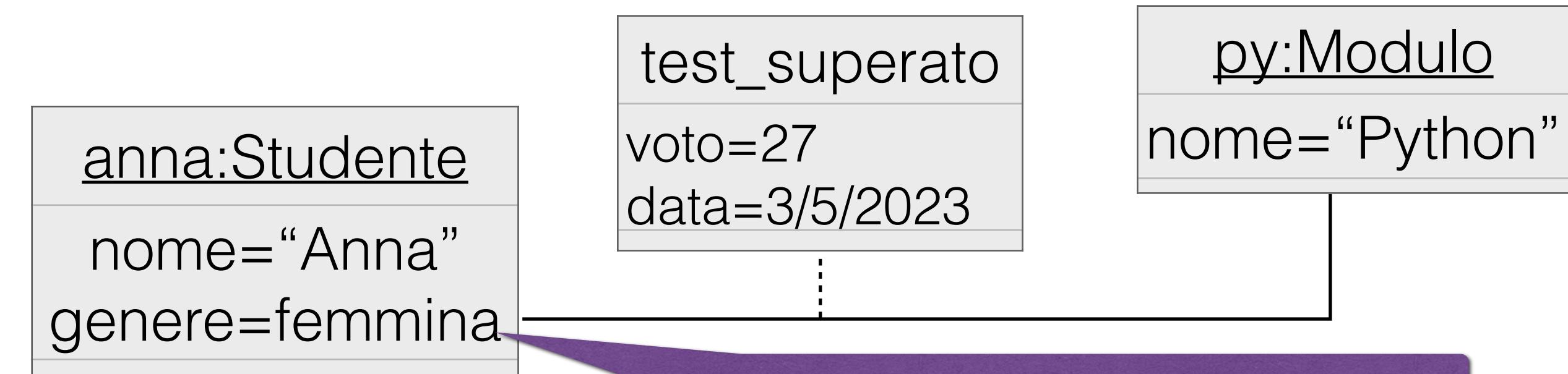
Ad esempio:

- {maschio, femmina}, {Africa, America, Antartide, Asia, Europa, Oceania}, etc.

Livello delle classi e delle associazioni (intensionale)



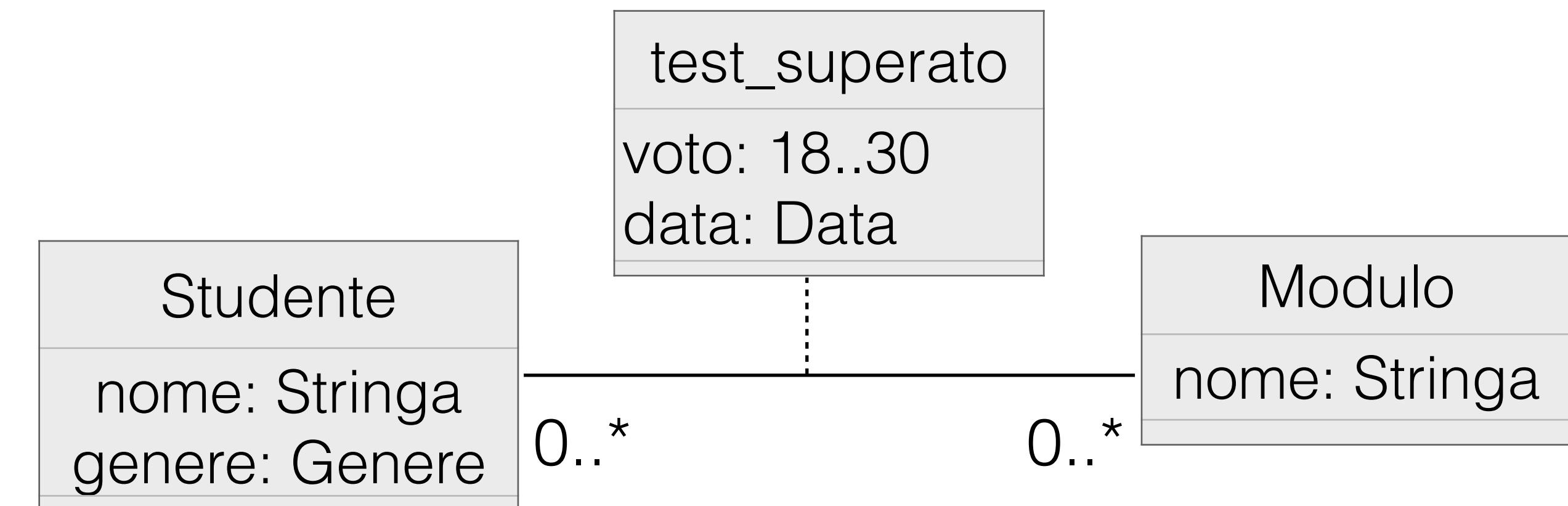
Possibile livello degli oggetti e dei link (estensionale)



Non è una stringa, ma un **simbolo** (valore del tipo enumerativo)

Tipi di dato definiti dall'utente

- UML consente all'analista di definire nuovi tipi di dato, che potranno essere usati liberamente nello schema concettuale.



- Tipo Genere = {maschio, femmina}

Tipi di dato composti

- UML consente all'analista di definire anche **tipi di dato composti** da più campi: **tipi record**

- Tipo Genere = {maschio, femmina}
- Tipo Indirizzo = (via:stringa, civico:intero>0, cap:intero>0)
(Attenzione: la scelta di usare il tipo ‘intero>0’ per i campi ‘civico’ e ‘cap’ non è affatto adeguata: è solo un semplice esempio!)

Studente
nome: Stringa
genere: Genere
indirizzo: Indirizzo

<u>anna:Studente</u>
nome=“Anna”
genere=femmina
indirizzo=(via=“Via di Casa mia”, civico=3, cap=84100)

Vincoli di molteplicità sugli attributi

- Anche gli attributi di classe e associazione possono avere vincoli di molteplicità (default: 1..1)

- Ogni studente ha associato esattamente un nome e un genere
- Ogni studente ha associato **uno o più** indirizzi email
- Ogni studente ha associato **al più un** indirizzo

Studente
nome: Stringa
genere: Genere
email: Stringa [1..*]
indirizzo: Indirizzo [0..1]

<u>anna:Studente</u>
nome=“Anna”
genere=femmina
email={"anna@kmail.com", “anna2002@yahuu.com”}
indirizzo={}

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione
UNITÀ: Progettazione.1

Prof. Toni Mancini
Dipartimento di Informatica
Sapienza Università di Roma



A.1.6
Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi e degli oggetti
Vincoli di identificazione di classe

Vincoli di identificazione di classe

- In alcuni casi, per modellare correttamente il dominio applicativo, è necessario imporre alcuni ulteriori vincoli oltre a quelli naturalmente imposti dal diagramma delle classi
- Vincolo (di integrità): una asserzione che impone restrizioni all'insieme dei livelli estensionali ammessi (ovvero agli insiemi di oggetti e link che possono coesistere) ulteriori a quelle strutturali che provengono dal diagramma delle classi
- Abbiamo già visto i vincoli di molteplicità sulle associazioni
- Ulteriore tipologia di vincolo:

vincolo di identificazione di classe

- Impone che non possono coesistere oggetti di una classe che coincidono nel valore di un insieme di attributi e/o sono collegati tramite link agli stessi oggetti di altre classi
- Esempio: Non possono esistere due persone con lo stesso codice fiscale (“{id1}”) e non possono esistere persone con, simultaneamente, lo stesso nome, cognome e data di nascita (“{id2}”)

Persona

cf: CodiceFiscale {id1}

 nome: Stringa {id2}

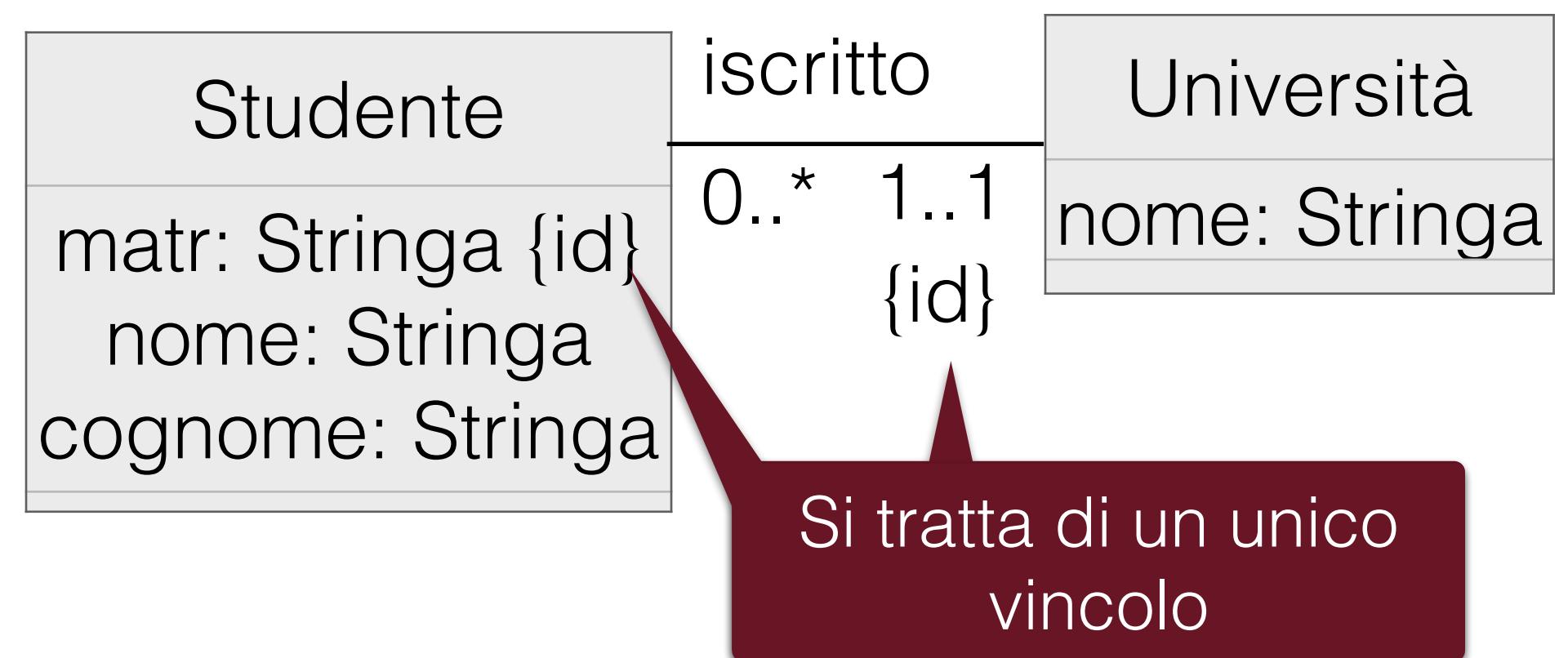
 cognome: Stringa {id2}

 nascita: Data {id2}

{id2} definisce un unico vincolo su tre attributi

Vincoli di identificazione di classe (2)

- Un vincolo di identificazione di classe può coinvolgere anche **ruoli** della classe
- Esempio: Non possono esistere due studenti con la stessa matricola nella stessa università
- **Attenzione:** un vincolo di identificazione di classe può coinvolgere solo **attributi a molteplicità [1..1]** e/o **ruoli della classe a molteplicità 1..1**



ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1.7

Analisi dei Requisiti

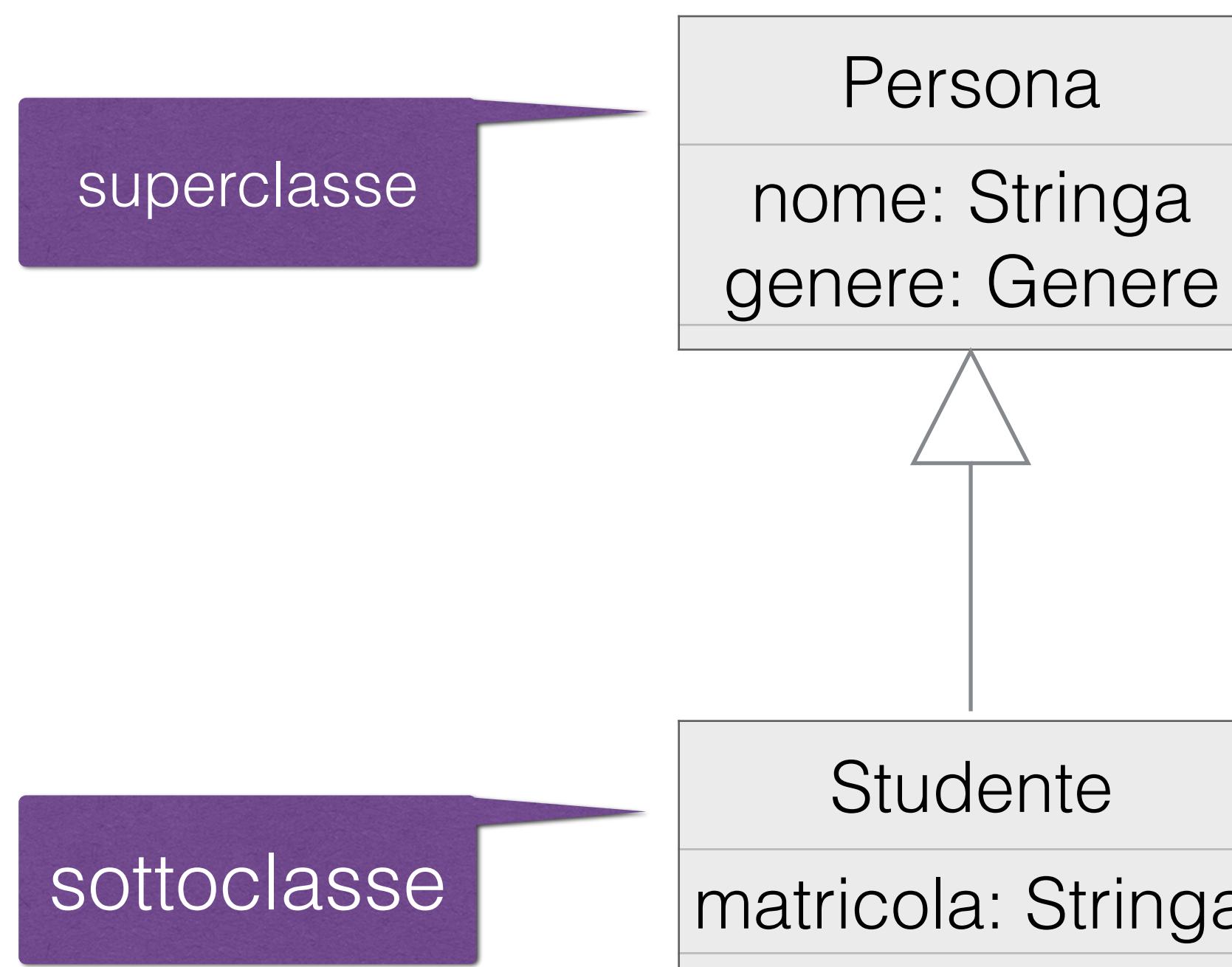
Unified Modeling Language

Diagrammi UML delle classi e degli
oggetti

Generalizzazione

Relazioni is-a tra classi

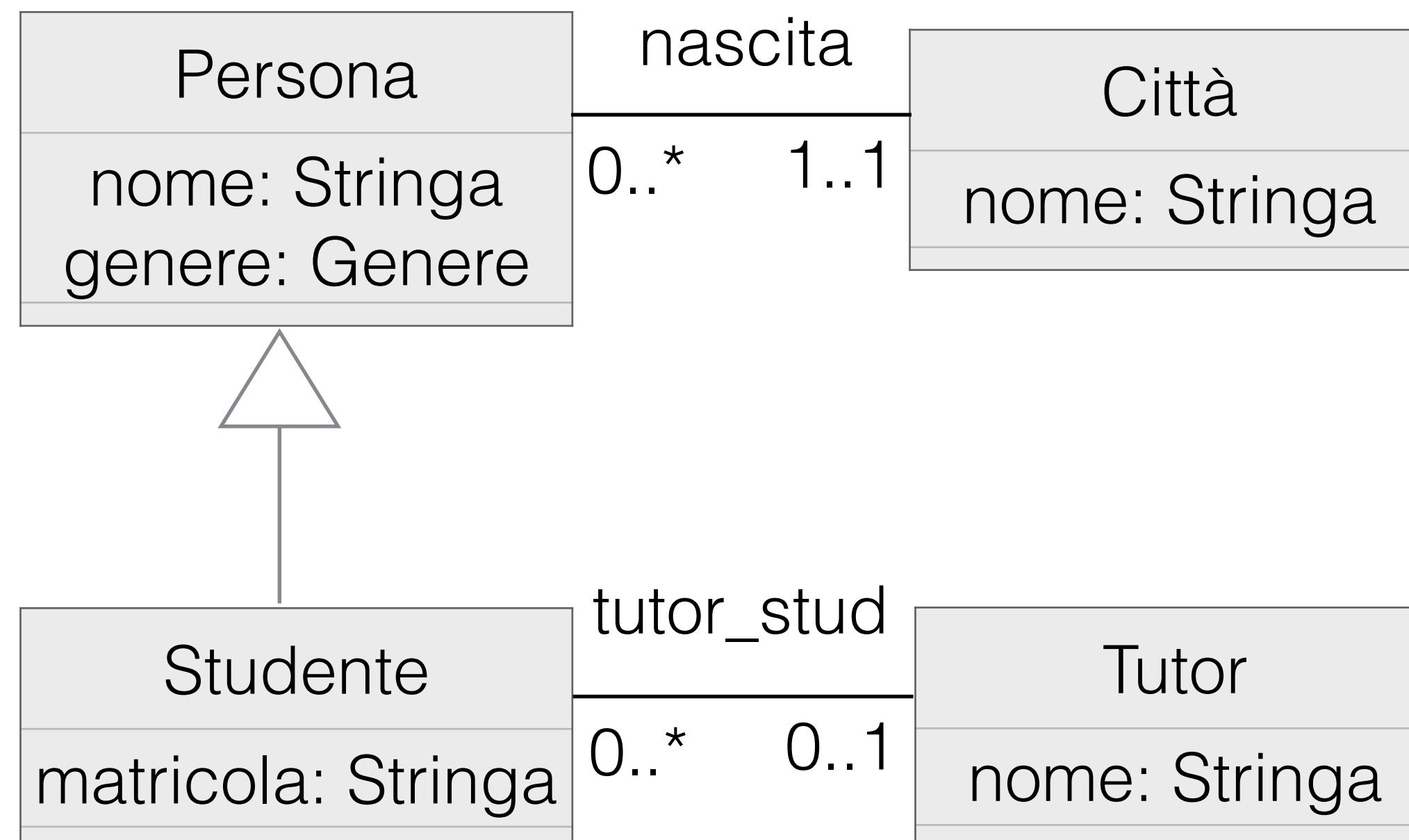
- Fino ad ora abbiamo implicitamente assunto che **classi diverse non hanno istanze in comune**
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di **sottoinsieme**
- I diagrammi delle classi permettono di definire il concetto di **relazione is-a tra classi**



- **Relazione is-a ("è anche un"):** **ogni** istanza della classe Studente (sotto-classe) è (concettualmente!) **anche** un'istanza della classe Persona (super-classe)
- Ovviamente **non vale il viceversa**: non tutte le istanze di Persona devono per forza essere anche istanze di Studente

Relazioni is-a tra classi: ereditarietà

- Fino ad ora abbiamo implicitamente assunto che classi diverse non hanno istanze in comune
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di sottoinsieme
- I diagrammi delle classi permettono di definire il concetto di **relazione is-a** tra classi

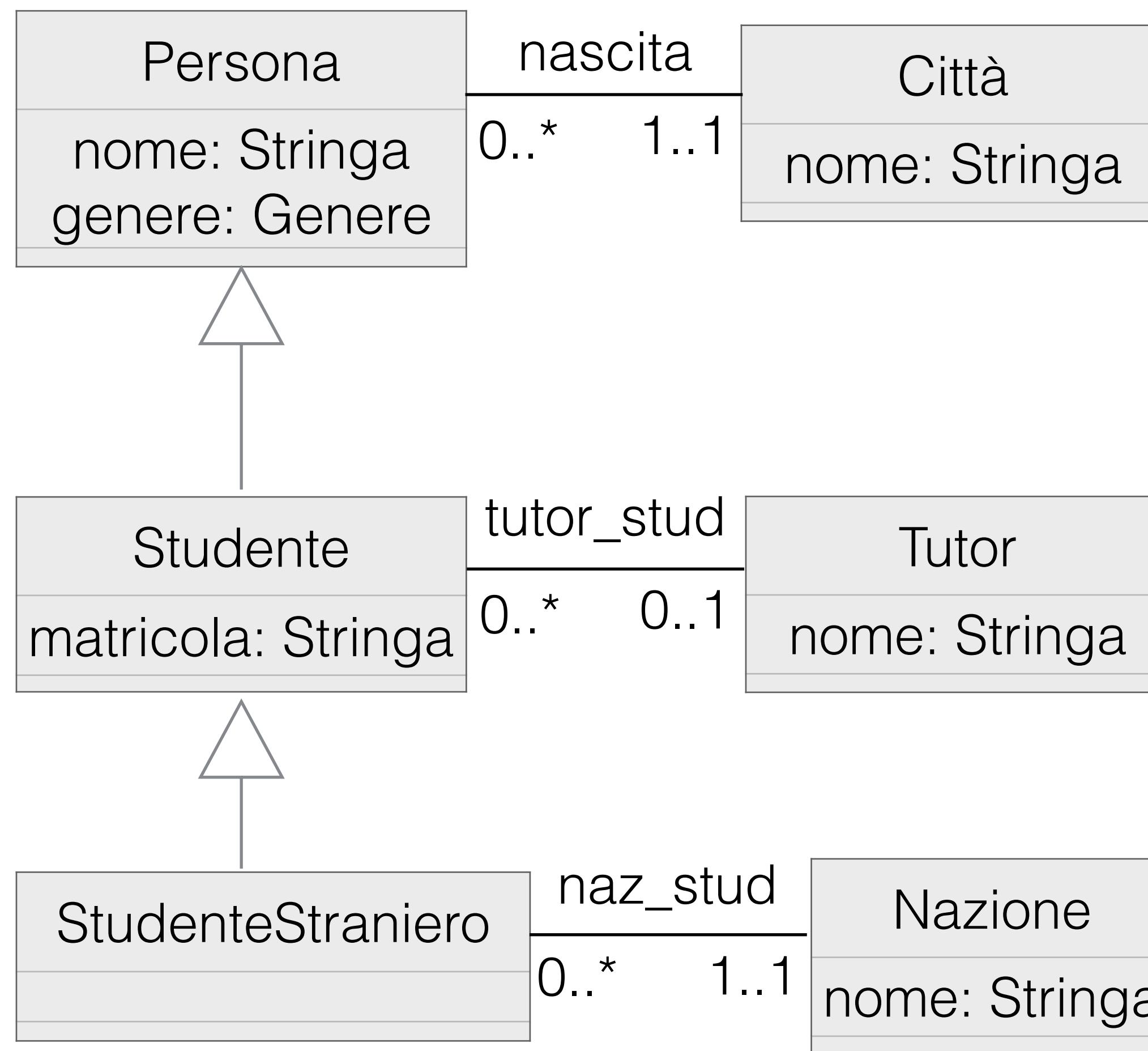


In presenza di relazioni is-a, vige il **meccanismo dell'ereditarietà** (ricorda il **sillogismo aristotelico**)

- Di tutte le persone di interesse vogliamo rappresentare nome, genere e città di nascita
- Di tutti gli studenti vogliamo rappresentare la matricola e l'eventuale tutor
- Tutti gli studenti sono persone (relazione is-a)
 - **Di conseguenza**, di tutti gli studenti **stiamo rappresentando anche** nome, genere, e città di nascita (con le loro molteplicità 1..1)

Relazioni is-a tra classi: ereditarietà (2)

- Fino ad ora abbiamo implicitamente assunto che classi diverse non hanno istanze in comune
- In molte situazioni, vogliamo rappresentare il fatto che tra due classi sussista una relazione di sottoinsieme
- I diagrammi delle classi permettono di definire il concetto di relazione is-a tra classi

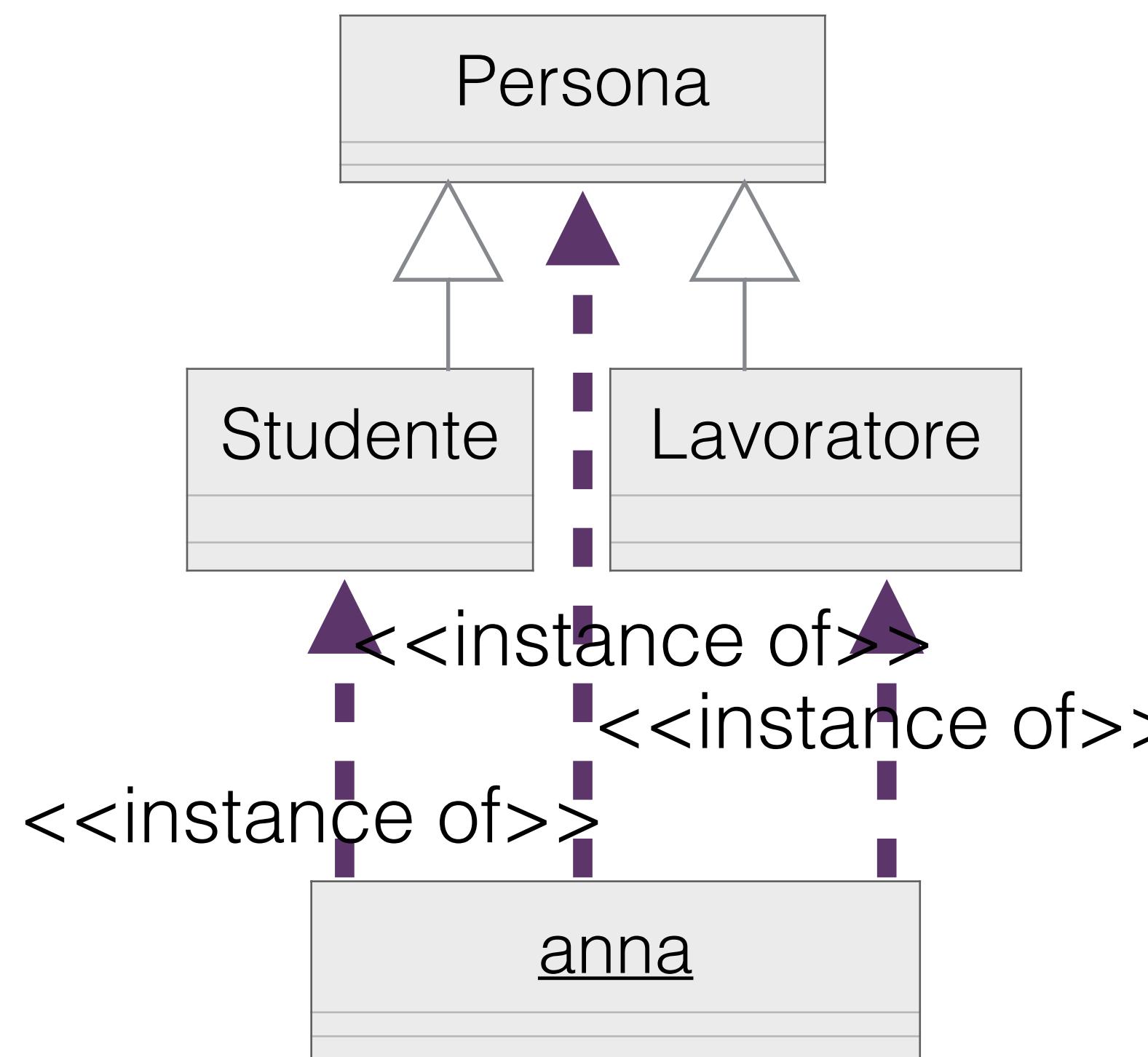


Ovviamente possiamo avere **relazioni is-a a più livelli: transitività!**

- Di tutte le persone di interesse vogliamo rappresentare nome, genere e città di nascita
- Di tutti gli studenti vogliamo rappresentare la matricola e l'eventuale tutor
- Di tutti gli studenti stranieri vogliamo rappresentare la nazione di provenienza
- Tutti gli studenti sono persone (relazione is-a)
 - **Di conseguenza**, di tutti gli studenti **stiamo rappresentando anche** nome, genere, ed città di nascita (con le loro molteplicità 1..1)
- Tutti gli studenti stranieri sono studenti (relazione is-a)
 - **Di conseguenza**, degli studenti stranieri **stiamo rappresentando anche** nome, genere, matricola, città di nascita (con le loro molteplicità 1..1)

Classi più specifiche di un oggetto

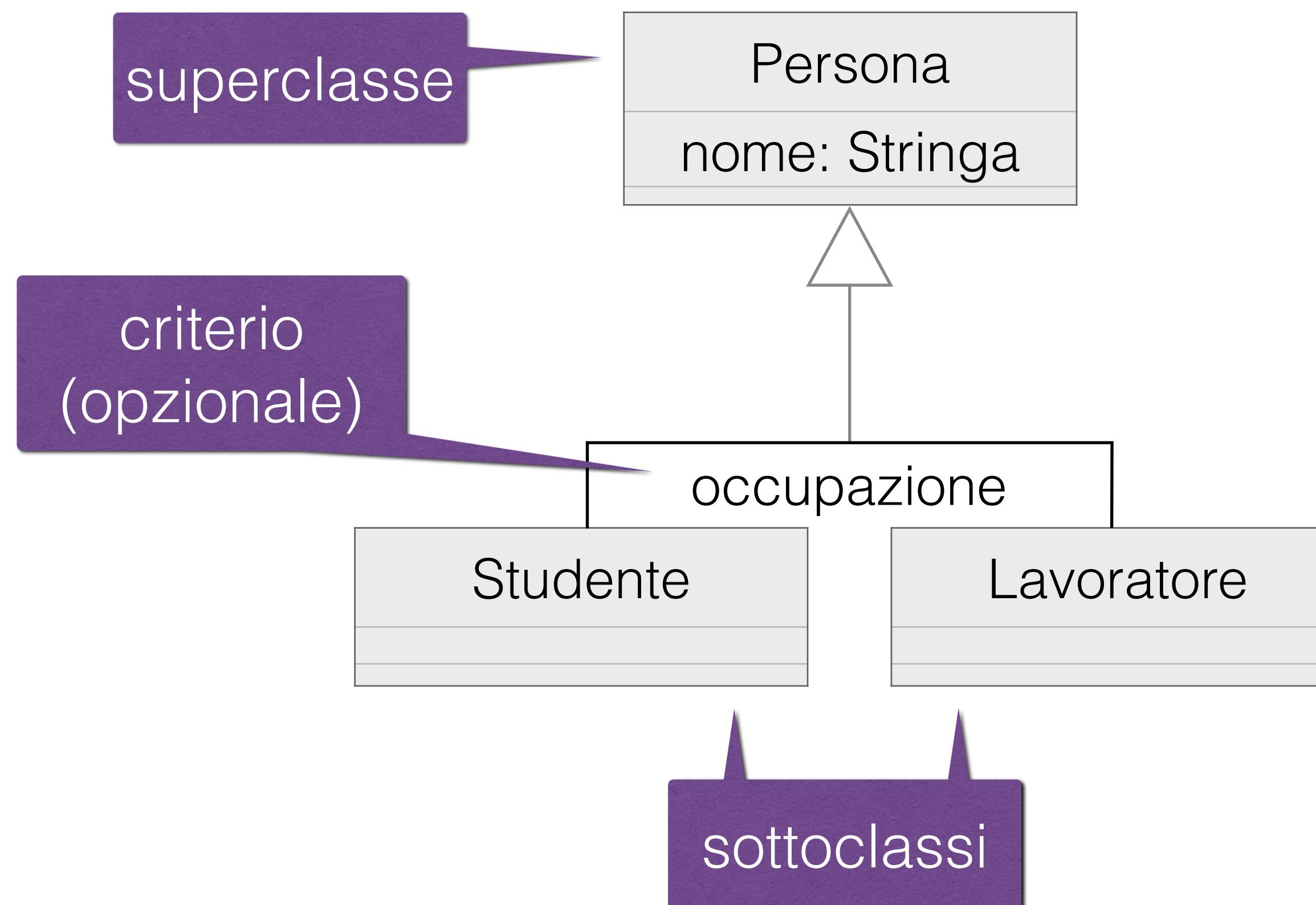
- **Classi più specifiche** di un oggetto O:
 - L'insieme delle classi di cui O è istanza che non sono a loro volta superclassi di altre classi di O
 - Esempio:



- L'oggetto ‘anna’ è istanza di Persona, Studente, Lavoratore
 - **L’insieme delle classi più specifiche** di ‘anna’ è {Studente, Lavoratore}
—> ...ma non Persona

Generalizzazioni

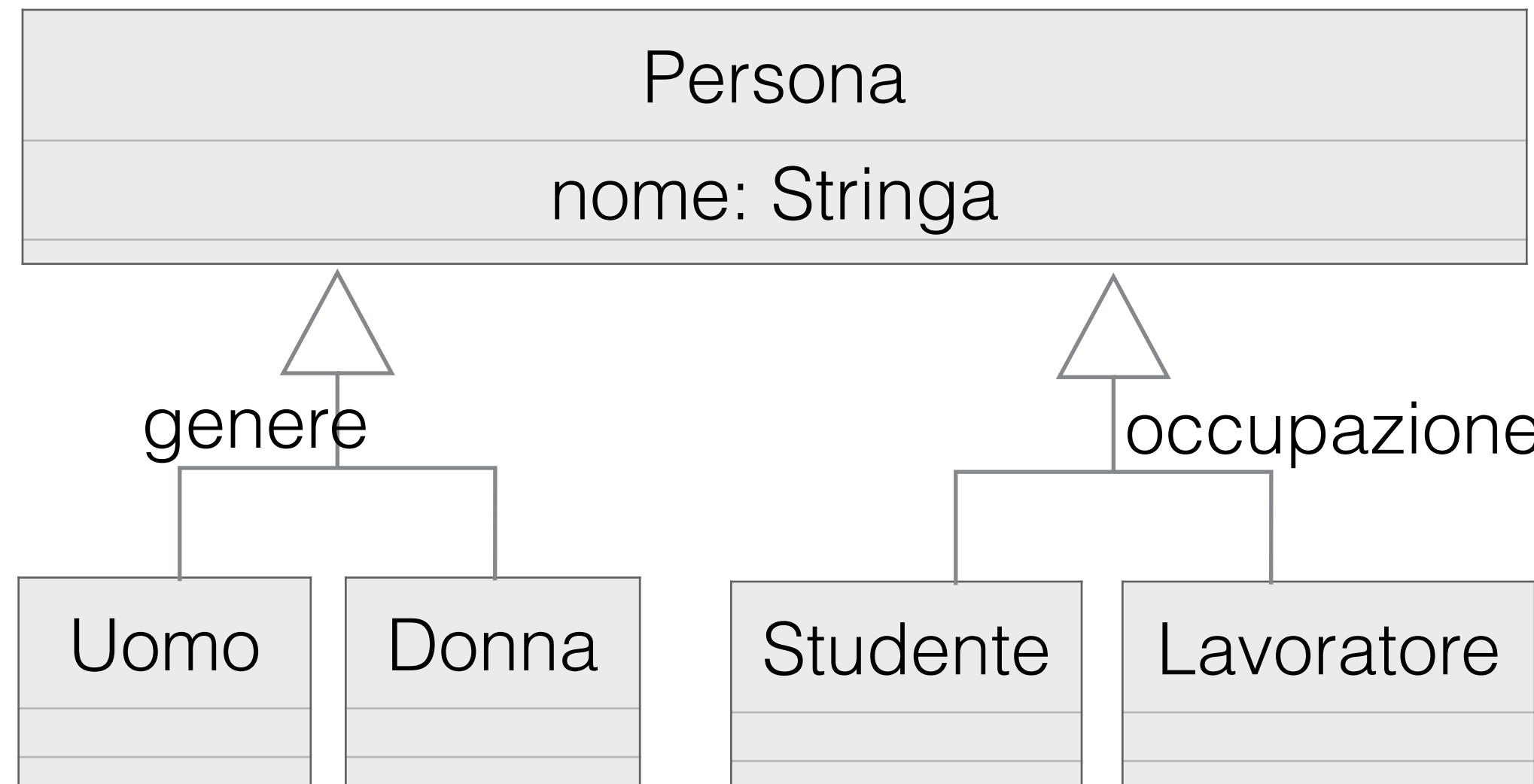
- I diagrammi delle classi UML offrono un costrutto più complesso della relazione is-a: il costrutto della **generalizzazione**
- Permette di definire che le istanze di una classe possono essere istanze di più classi figlie **secondo uno stesso criterio concettuale**



- Significato:
 - Studente **is-a** Persona
 - Lavoratore **is-a** Persona
 - Il **criterio** secondo il quale le Persone sono considerate studenti e/o lavoratori è **lo stesso**: quello dell'“occupazione”
- È ancora possibile per un oggetto essere istanza sia di Studente che di Lavoratore

Generalizzazioni (2)

- La stessa classe può essere superclasse di generalizzazioni distinte (criteri diversi)

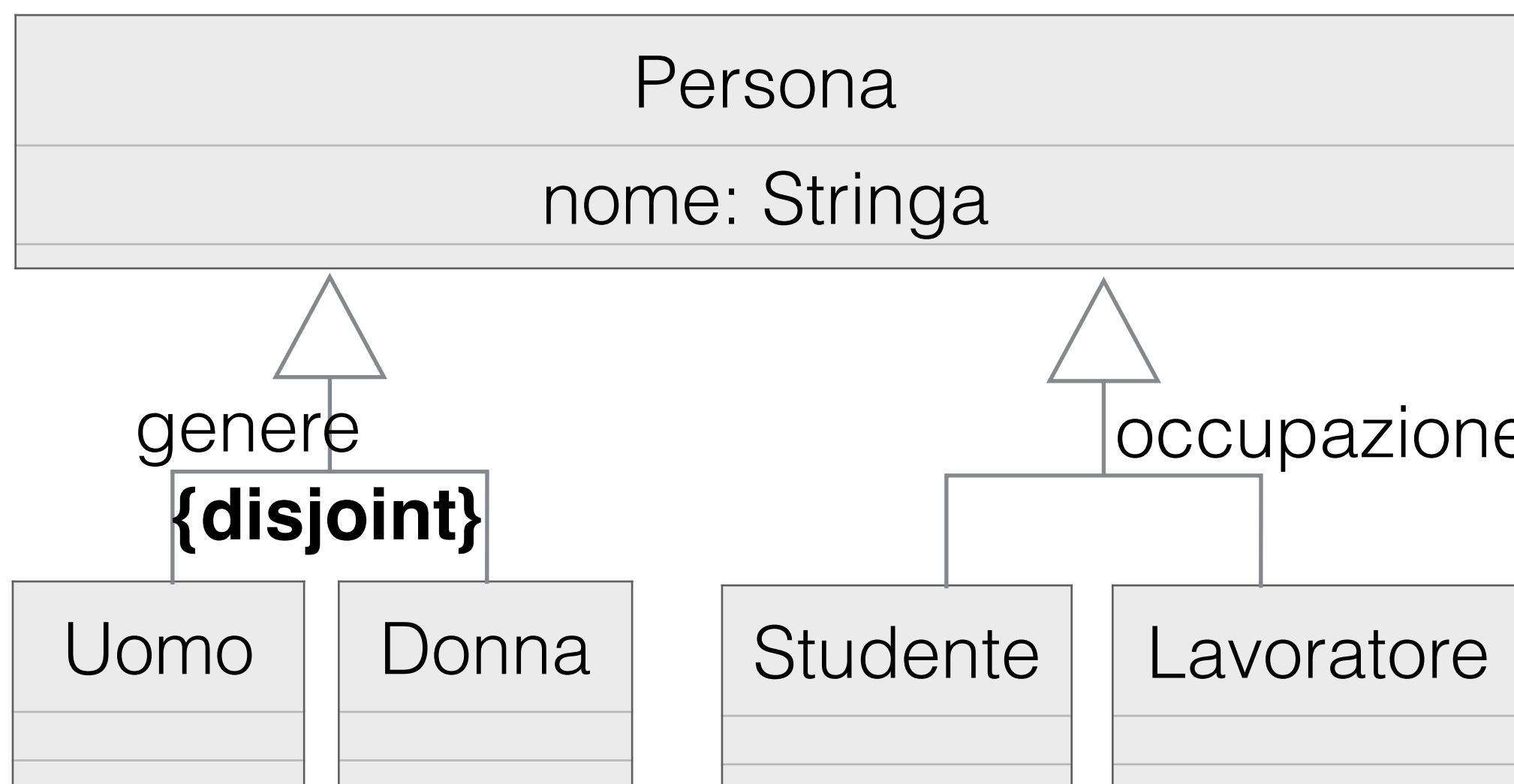


- Significato:
 - Secondo il **criterio** del genere, le Persone sono considerate uomini e/o donne
 - Secondo il **criterio (indipendente!)** dell'occupazione, le persone sono considerate studenti e/o lavoratori

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? Sì
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? Sì

Vincoli sulle generalizzazioni: {disjoint}

- In linea di principio, una istanza della classe base può essere istanza di più di una sottoclasse di una stessa generalizzazione
- Spesso, nell'ottica di modellare accuratamente i requisiti, vorremmo evitarlo: **generalizzazioni disgiunte**.

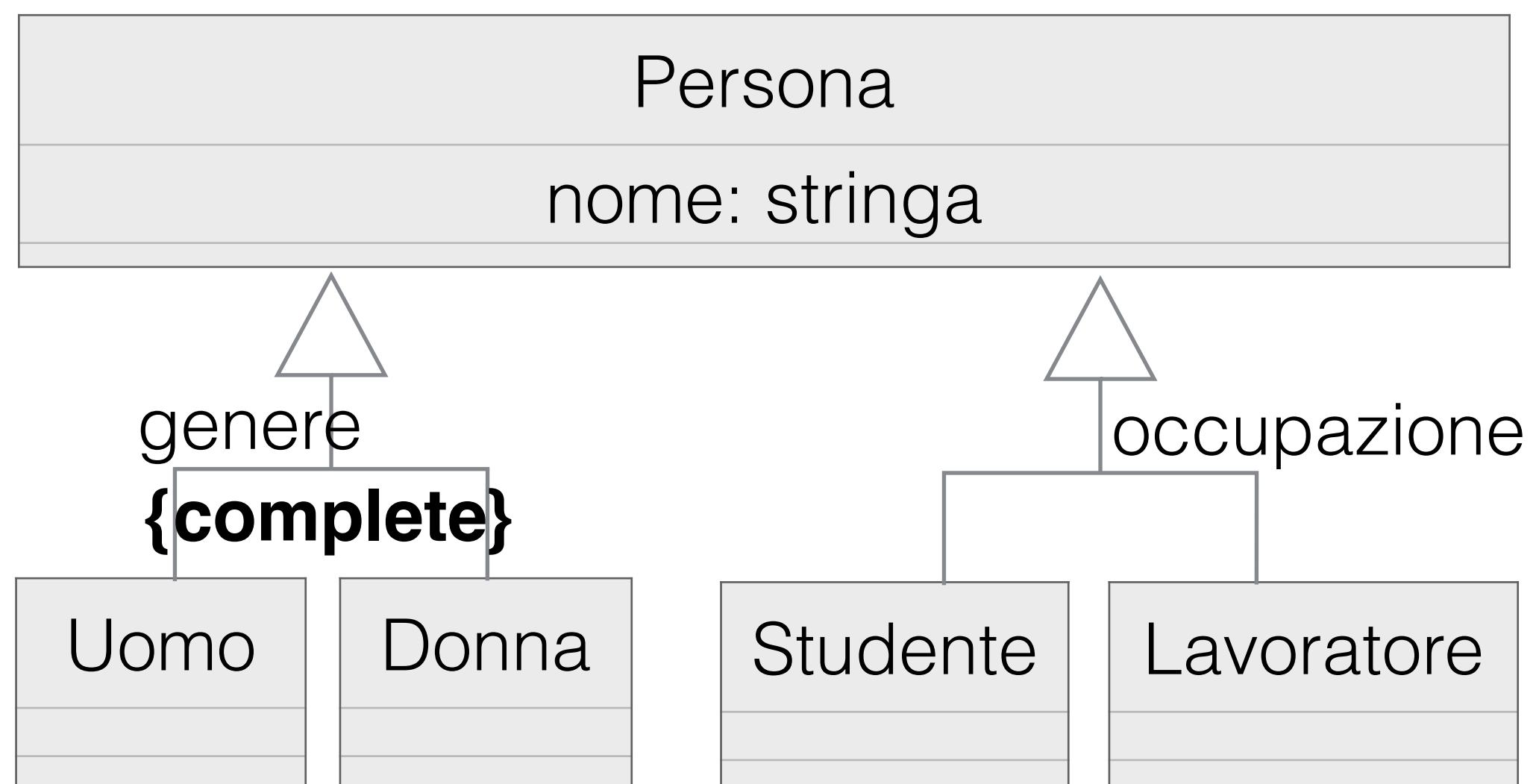


- Significato:
 - La generalizzazione sul genere è disgiunta: una istanza di Uomo non può essere anche istanza di Donna
 - La generalizzazione sull'occupazione non è disgiunta: una istanza di Studente può essere anche istanza di Lavoratore (ma...)

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? Sì
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? **Non più**

Vincoli sulle generalizzazioni: {complete}

- In linea di principio, una istanza della classe base può essere istanza di nessuna delle sottoclassi di una stessa generalizzazione
- Spesso, nell'ottica di modellare accuratamente i requisiti, vorremmo evitarlo: **generalizzazioni complete**.

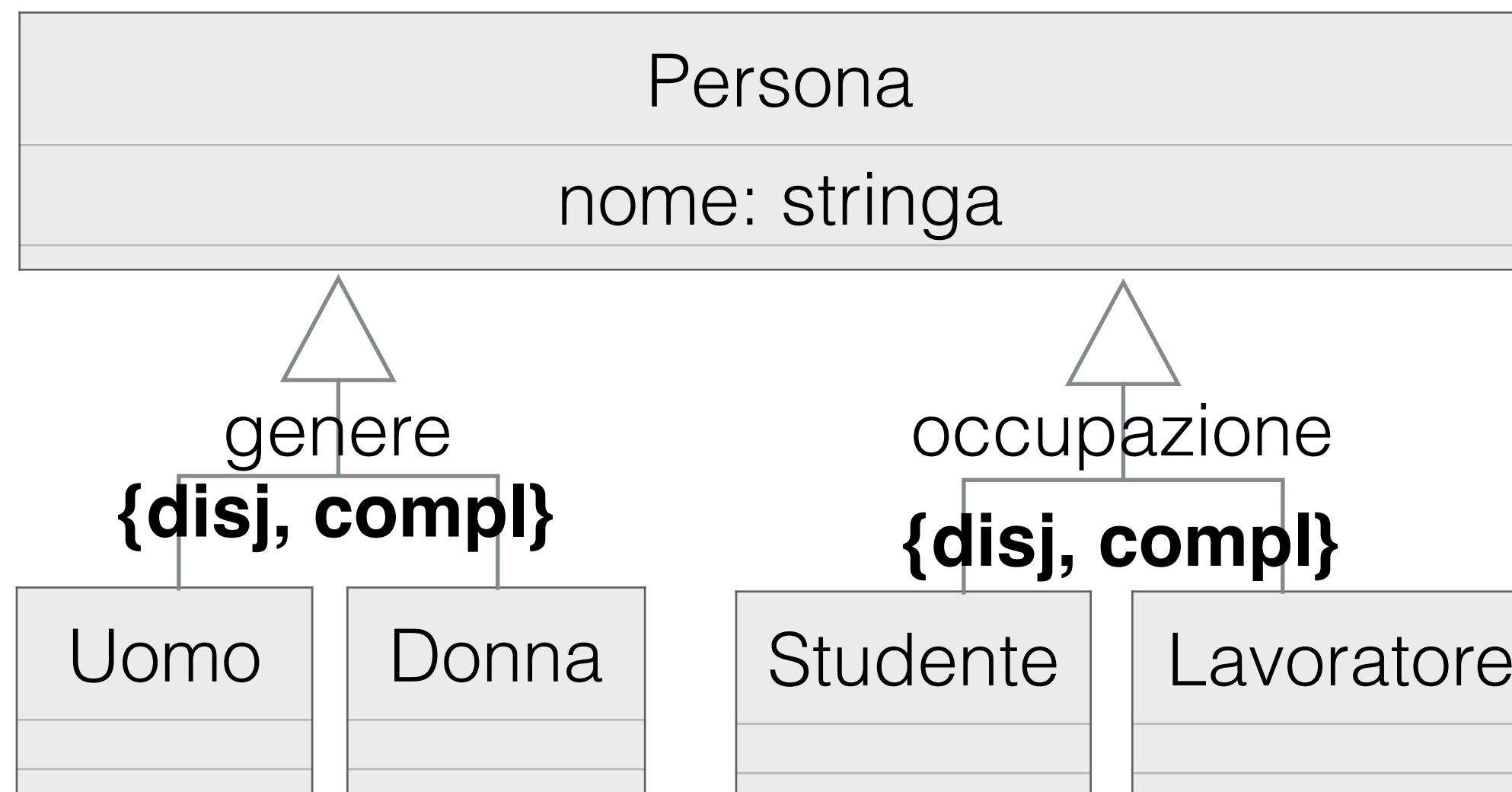


- Significato:
 - La generalizzazione sul genere è completa: ogni istanza di Persona deve essere anche istanza di almeno una sottoclasse tra Uomo e Donna
 - La generalizzazione sull'occupazione non è completa: possono esistere istanze di Persona che non sono né istanze di Studente né istanze di Lavoratore

- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? **Non più**
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? Sì

Vincoli sulle generalizzazioni: {disjoint, complete}

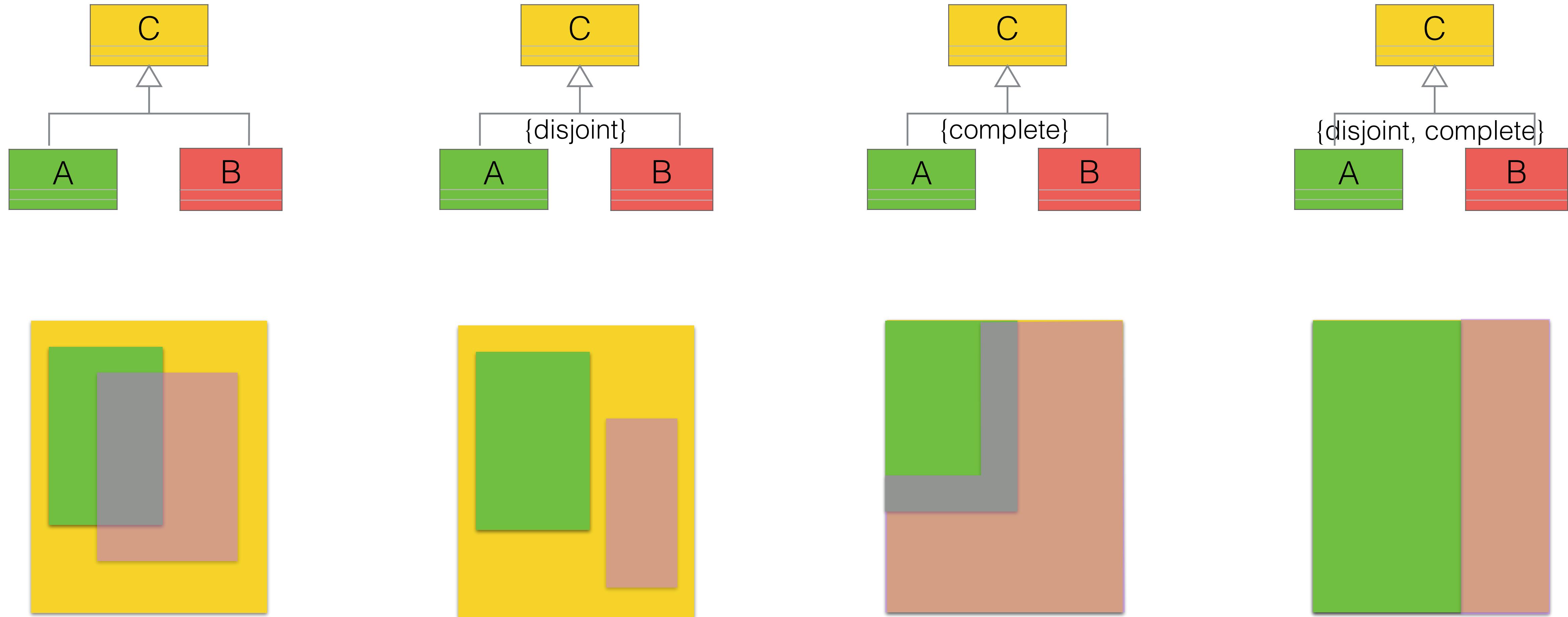
- Una generalizzazione può essere **sia disgiunta che completa**
- Ogni istanza della superclasse è anche istanza di **esattamente una** sottoclasse della generalizzazione



- Significato:
 - ogni istanza di Persona deve essere anche istanza di esattamente una sottoclasse tra Uomo e Donna
 - ogni istanza di Persona deve essere anche istanza di esattamente una sottoclasse tra Studente e Lavoratore

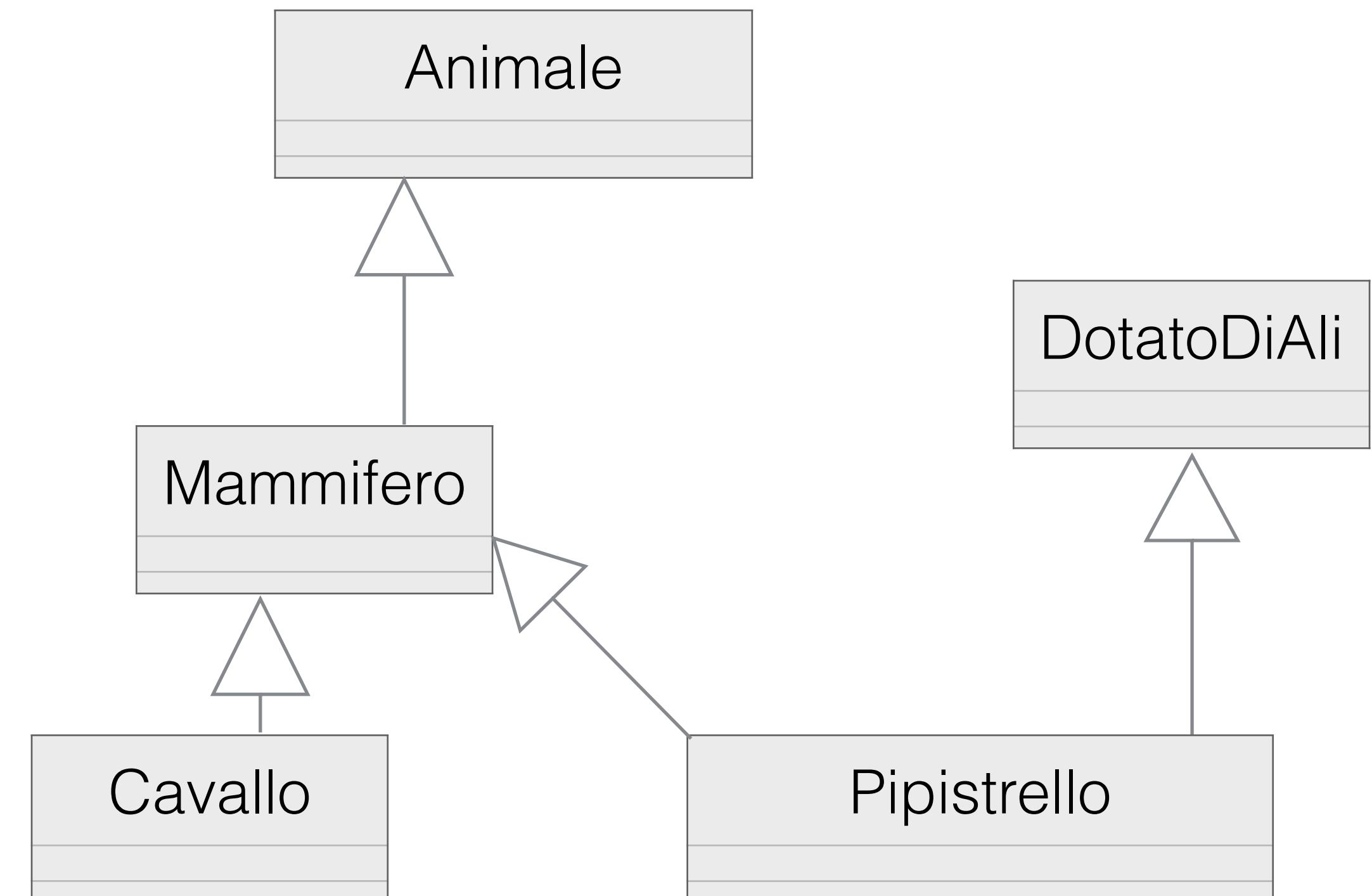
- Una istanza di Persona può essere sia istanza di Uomo che di Studente? Sì
- Una istanza di Persona può essere istanza né di Uomo né di Donna? **Non più**
- Una istanza di Persona può essere istanza sia di Uomo che di Donna? **Non più**

Vincoli sulle generalizzazioni: riepilogo



Ereditarietà multipla

- Una classe può essere sottoclasse di più classi. Il meccanismo dell'ereditarietà vale come sempre.



ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione

UNITÀ: Progettazione.1

Prof. Toni Mancini

Dipartimento di Informatica
Sapienza Università di Roma



A.1.1.8

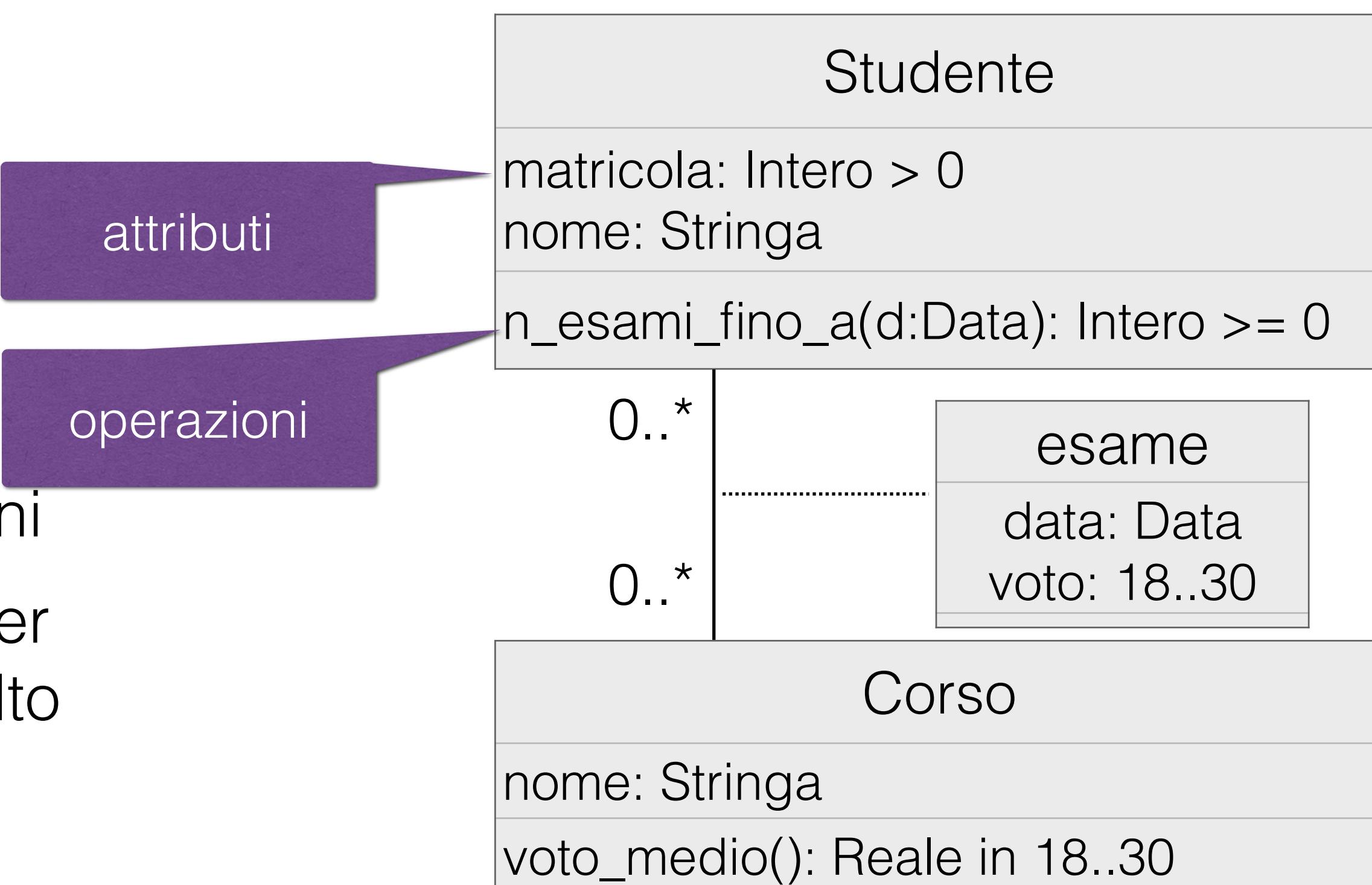
Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi e degli oggetti
Operazioni di classe

Operazioni di classe

- Finora abbiamo fatto riferimento solo a proprietà statiche (attributi e associazioni) di classi.
 - Proprietà statiche: i valori cambiano a seguito di esplicita modifica dei dati da parte degli utenti del sistema (o da parte di altre parti del sistema)
- Una classe UML può definire anche proprietà dinamiche, che si chiamano operazioni.
 - Proprietà dinamiche: i valori vengono calcolati ogni volta che servono, a partire dai valori di altre proprietà

Operazione di classe

- Una operazione della classe C indica che su ogni oggetto (istanza) della classe C si può eseguire un calcolo per:
 - calcolare un valore a partire da altri dati e operazioni
 - effettuare cambiamenti di stato dell'oggetto (cioè per modificare le sue proprietà), dei link in cui è coinvolto e/o degli oggetti a questo collegati



Operazioni: sintassi

Sintassi per la segnatura dell'operazione:

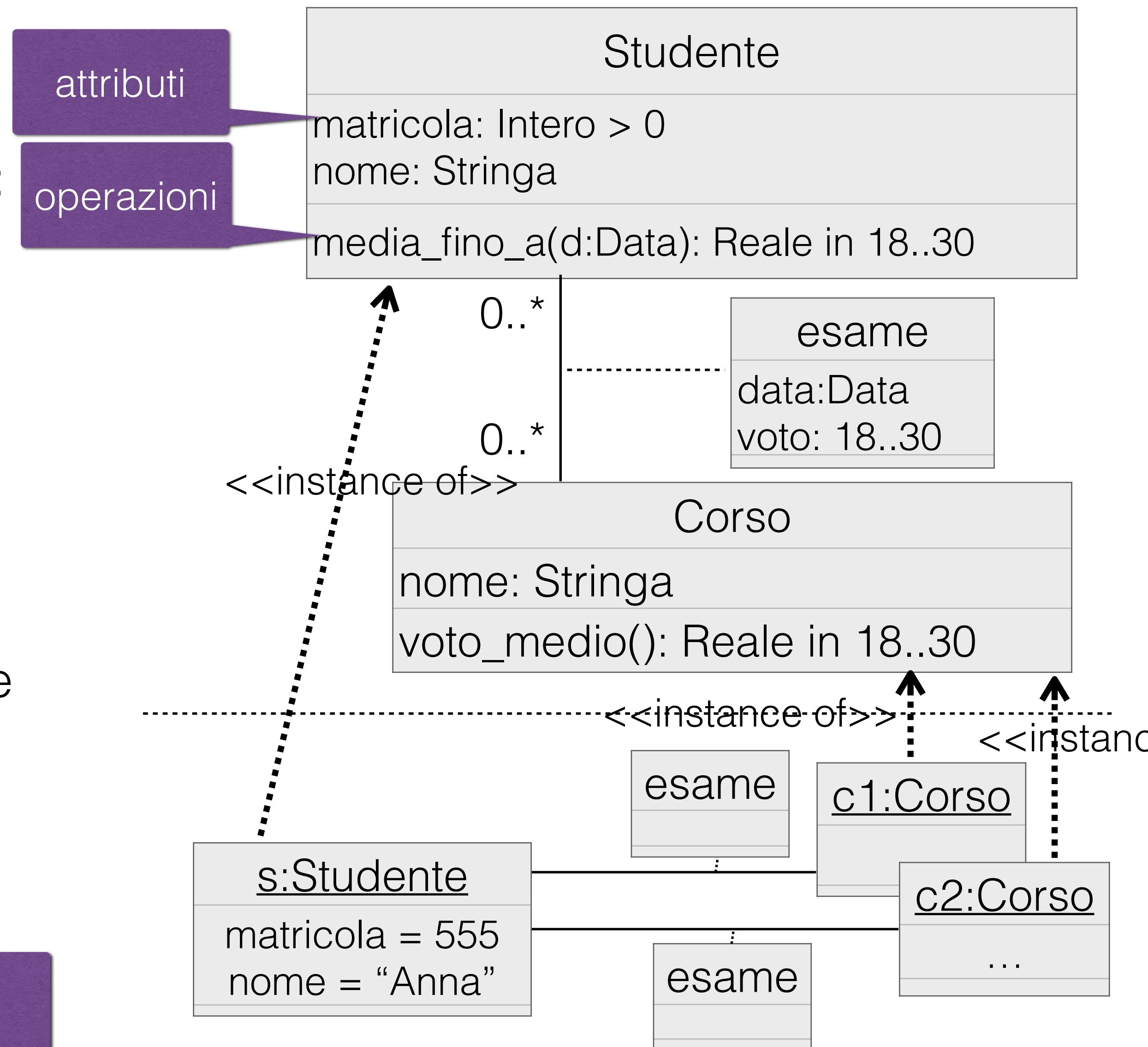
- **nome_operazione(argomenti) : tipo Ritorno**
dove:
- “**argomenti**” è una lista di elementi della forma:
nome_argomento : tipo_argomento
- “**tipo Ritorno**” è il tipo del valore restituito dall'operazione
- I tipi degli argomenti e del valore di ritorno possono essere tipi di dato concettuali oppure classi del diagramma
- Una operazione di classe può essere invocata **solo** su un oggetto della classe, che è un ulteriore argomento (non indicato nella segnatura)

s.media_fino_a(2/6/2023) → 22.5

oggetto di invocazione

valore degli argomenti

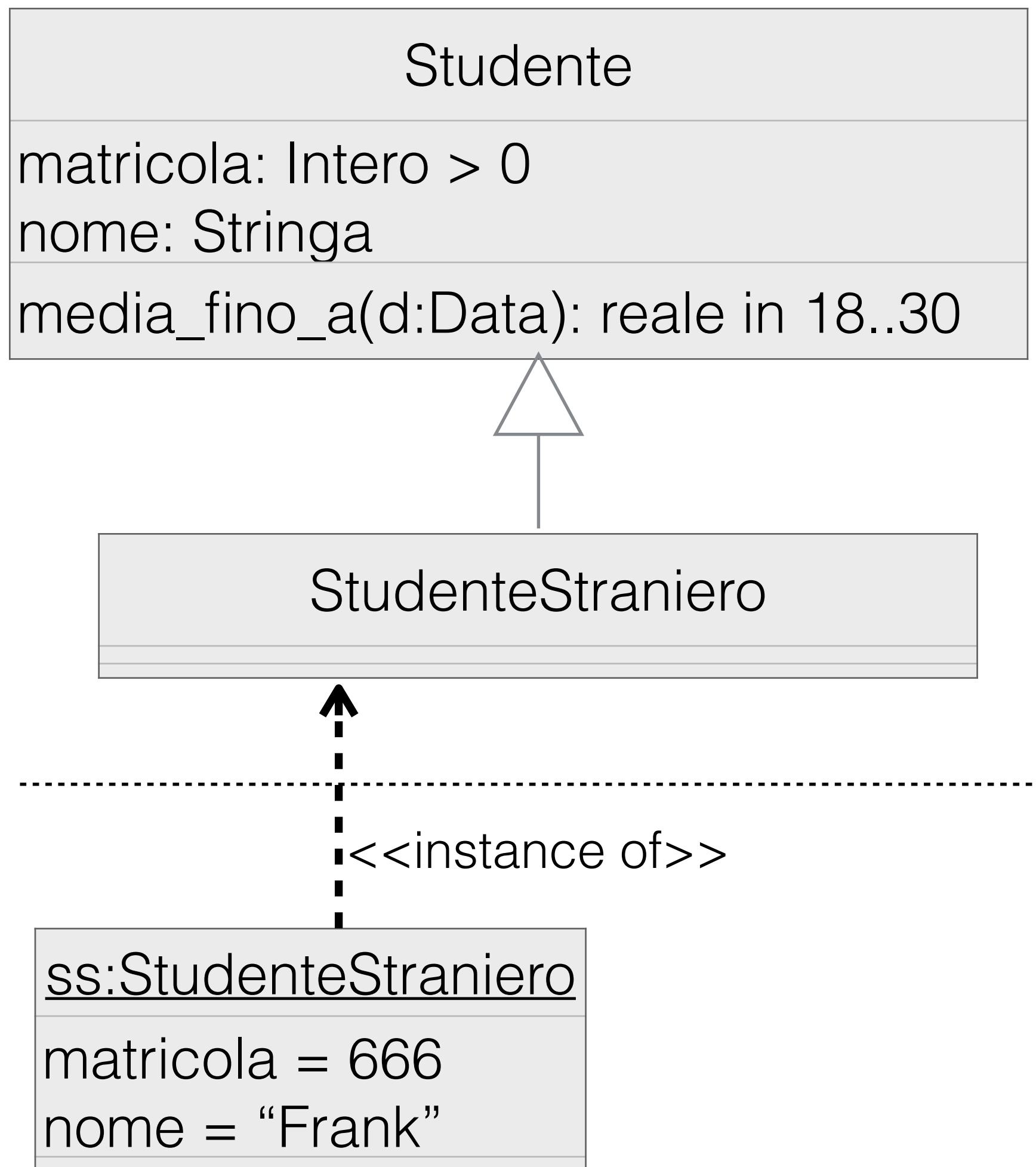
valore di ritorno



Operazioni: ereditarietà

- Il meccanismo dell'ereditarietà si applica anche alle operazioni di classe
- Dunque, ad es., possiamo invocare:

`ss.media_fino_a(2/6/2023) → 28.3`



Operazioni: specifiche

- Il diagramma delle classi non definisce cosa calcolano le operazioni, né se e come modificano i dati
- Ogni classe del diagramma con operazioni andrà affiancata da un documento di specifica che entra nel dettaglio (v. seguito)

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione
UNITÀ: Progettazione.1

Prof. Toni Mancini
Dipartimento di Informatica
Sapienza Università di Roma

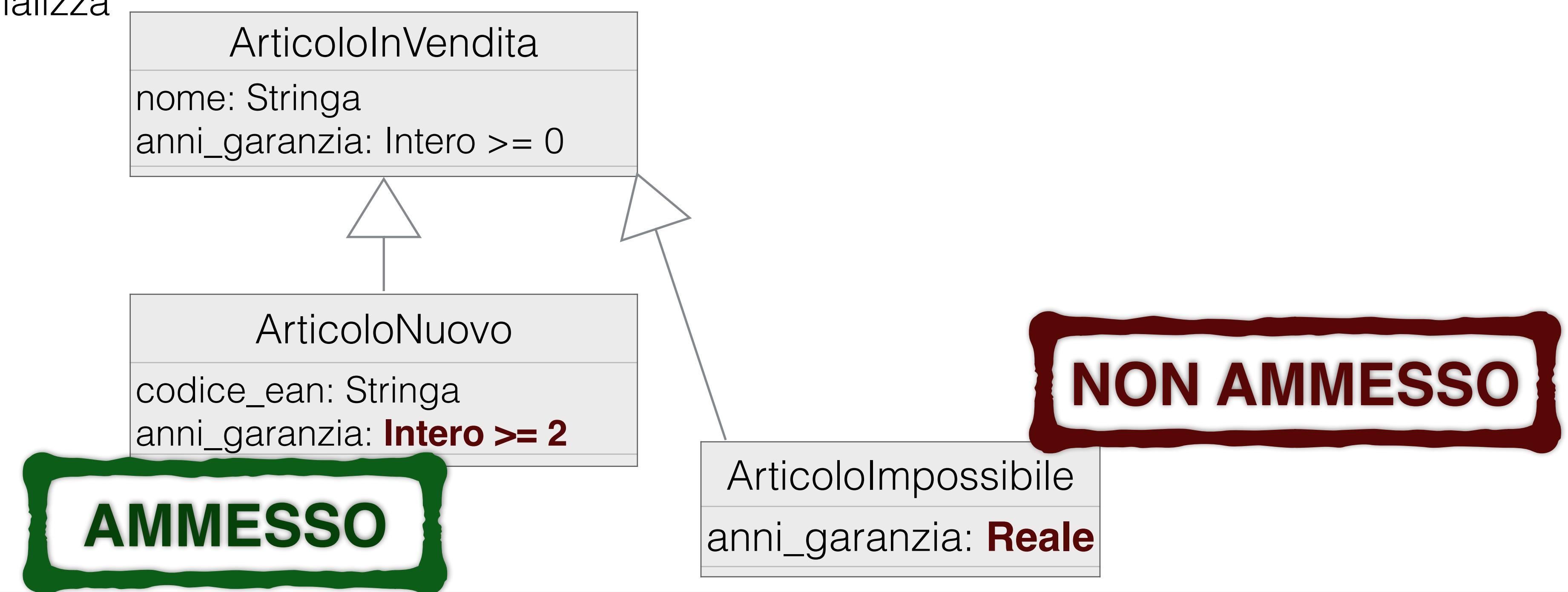


A.1.1.9
Analisi dei Requisiti
Unified Modeling Language
Diagrammi UML delle classi e degli oggetti
Specializzazioni di attributi,
associazioni ed operazioni

Specializzazione di attributi

- In una generalizzazione, una sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche **specializzare** le proprietà ereditate dalla superclasse, **restringendone il tipo**
- Degli articoli in vendita vanno rappresentati: nome e numero di anni di garanzia (anche zero)
- Alcuni articoli sono nuovi
- Degli articoli nuovi interessa anche il codice EAN
- Inoltre, per gli articoli nuovi, la garanzia deve essere di **almeno due anni**

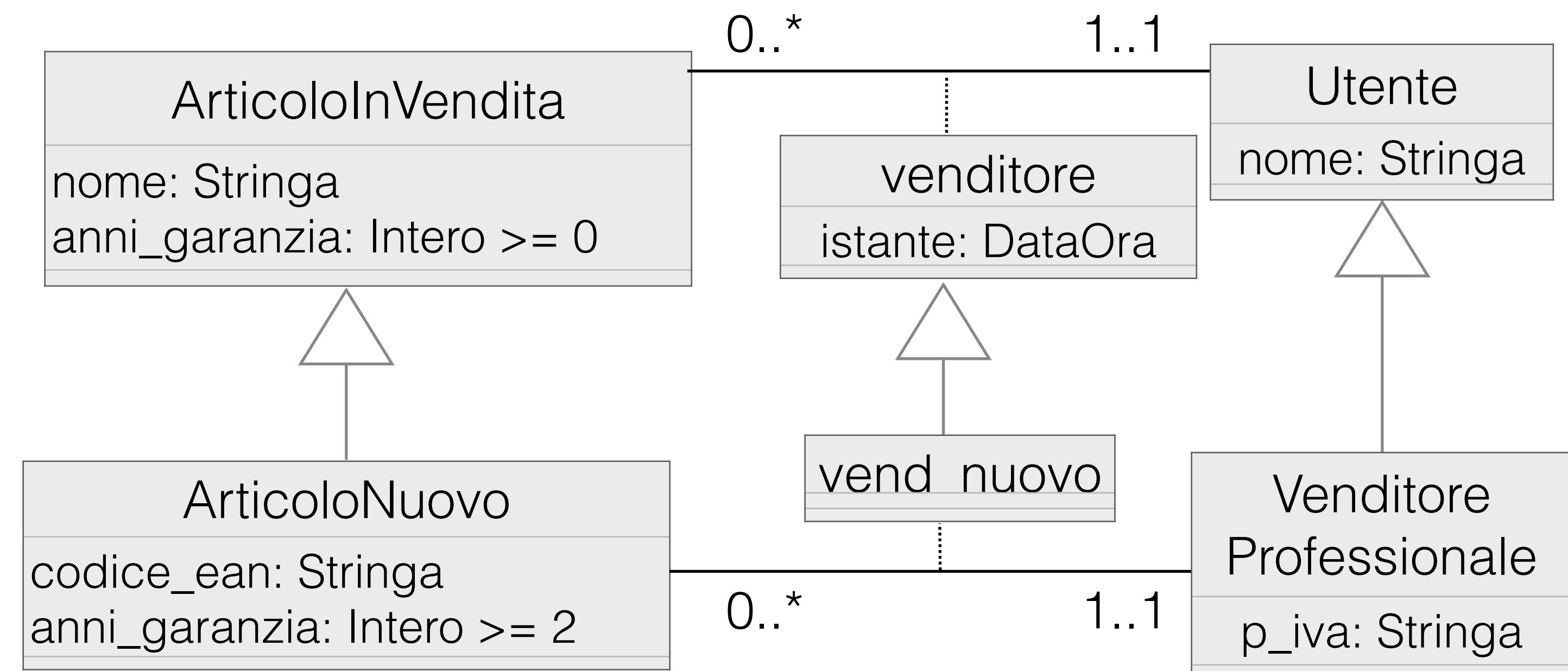
Attenzione: il tipo dell'attributo specializzato (`ArticoloNuovo.anni_garanzia`) deve essere **più ristretto** del tipo dell'attributo che specializza



Specializzazione di associazioni

- Ricordiamo che un'associazione con attributi si chiama “association class” e dunque... è anche una classe, e dunque... può a sua volta essere terminale di associazioni
 - Una association class, in quanto classe, può anche essere radice di relazioni is-a e generalizzazioni!
- Ogni articolo è venduto da **esattamente** un utente
- Gli articoli nuovi possono essere venduti **solo** da venditori professionali

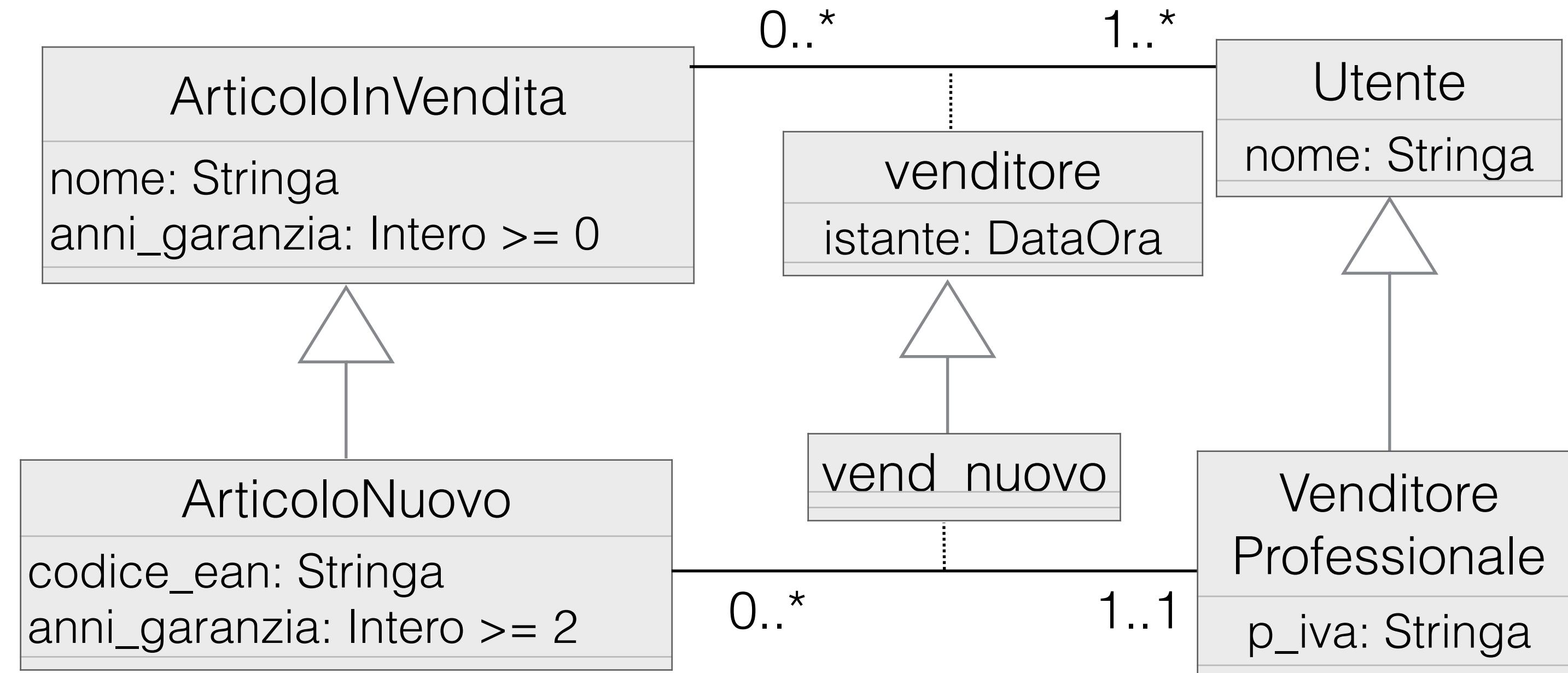
Attenzione: l'associazione *vend_nuovo* deve essere di tipo compatibile con il tipo dell'associazione *venditore*



Specializzazione di associazioni (2)

- Ricordiamo che un'associazione con attributi si chiama “association class” e dunque... è anche una classe, e dunque... può a sua volta essere terminale di associazioni
 - Una association class, in quanto classe, può anche essere radice di relazioni is-a e generalizzazioni!
- Ogni articolo è venduto da **almeno un** utente
- Gli articoli nuovi devono essere venduti da **almeno un** venditore professionale

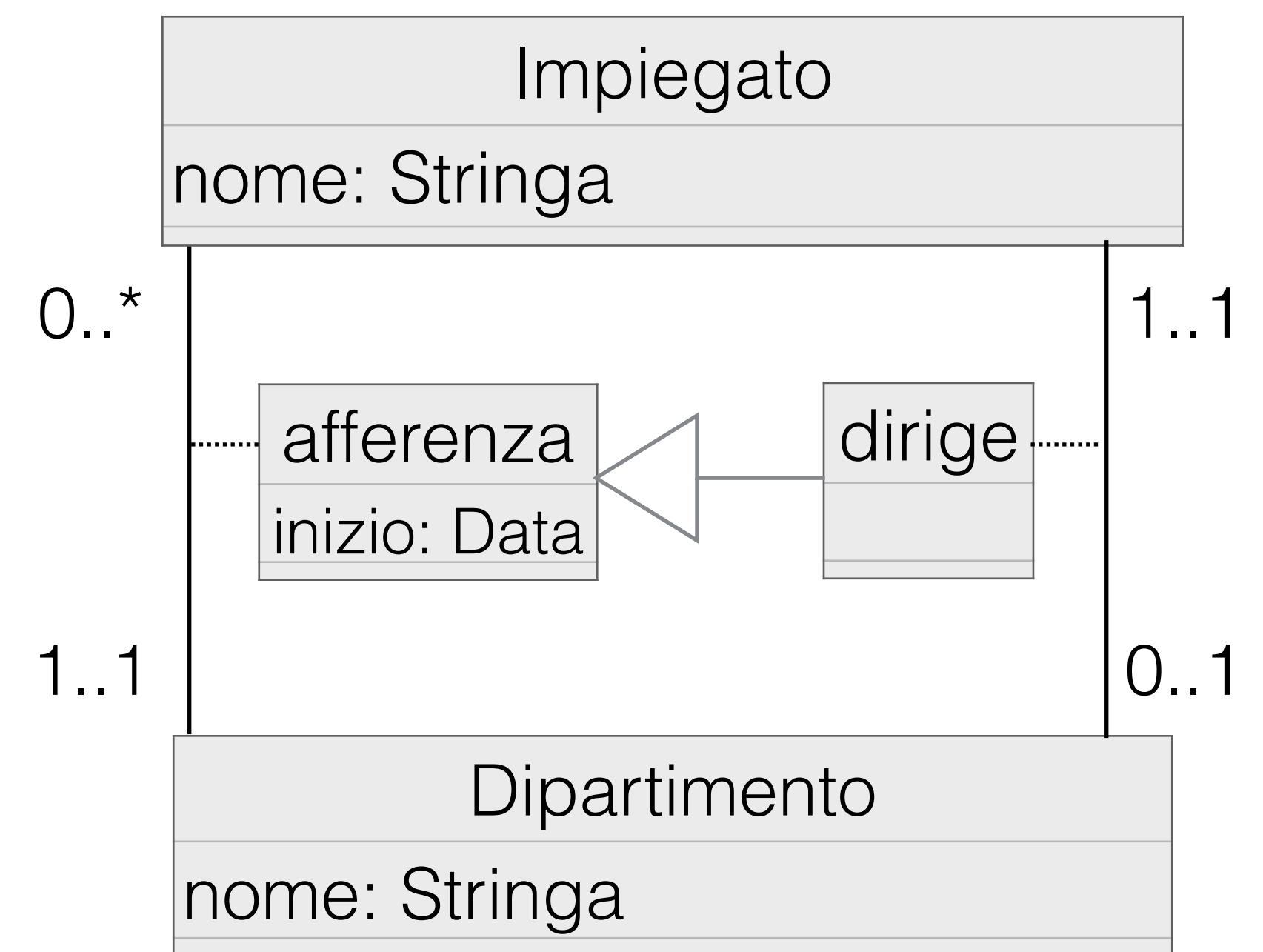
Attenzione: l'associazione *vend_nuovo* deve essere di tipo compatibile con il tipo dell'associazione *venditore*



Specializzazione di associazioni (3)

Esempio:

- Il sistema deve rappresentare impiegati, il dipartimento a cui afferisce ogni impiegato (con data di inizio afferenza) e direttore (un impiegato).
- I direttori devono afferire al dipartimento che dirigono.

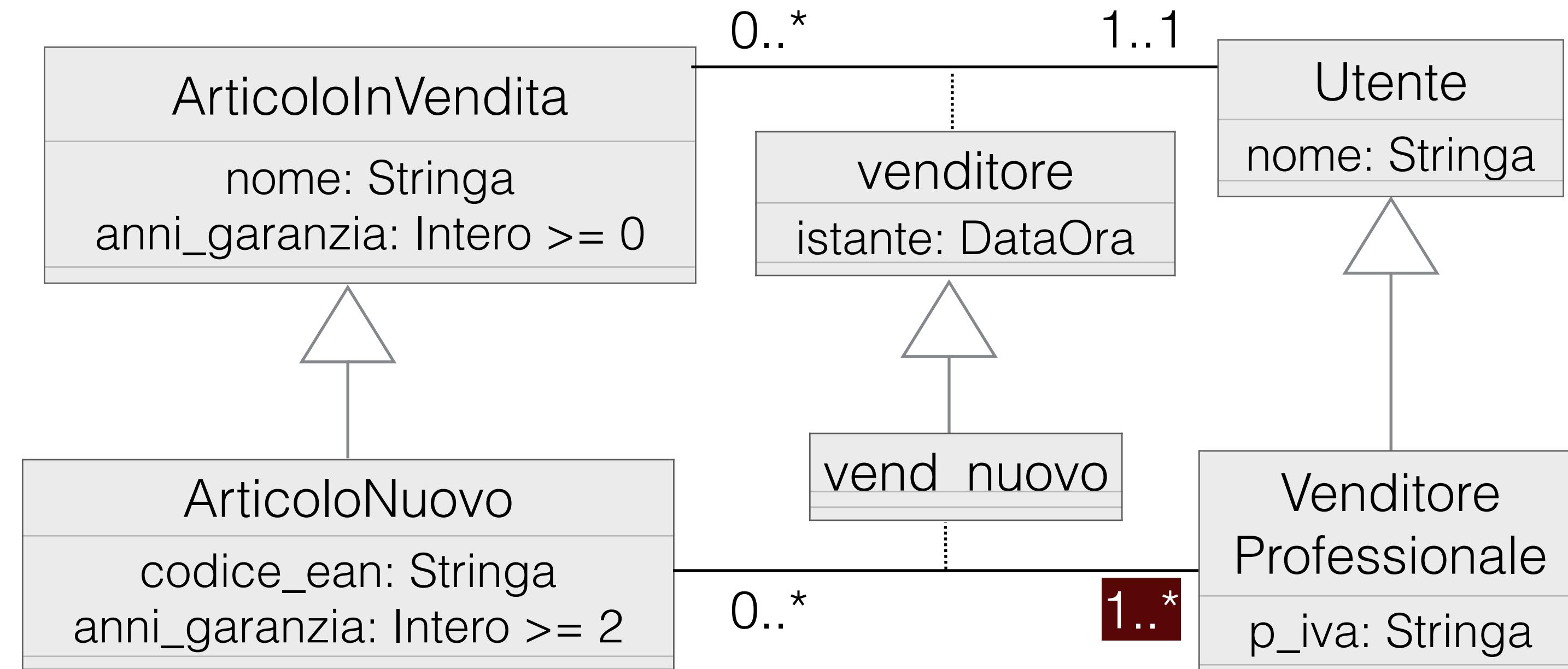


Specializzazione di associazioni: condizioni per la correttezza

Le specializzazioni di associazioni necessitano di particolare attenzione relativamente ai vincoli di molteplicità.

Esempio:

- Ogni articolo è venduto da **esattamente un** utente
- Gli articoli nuovi devono essere venduti da **almeno un** venditore professionale



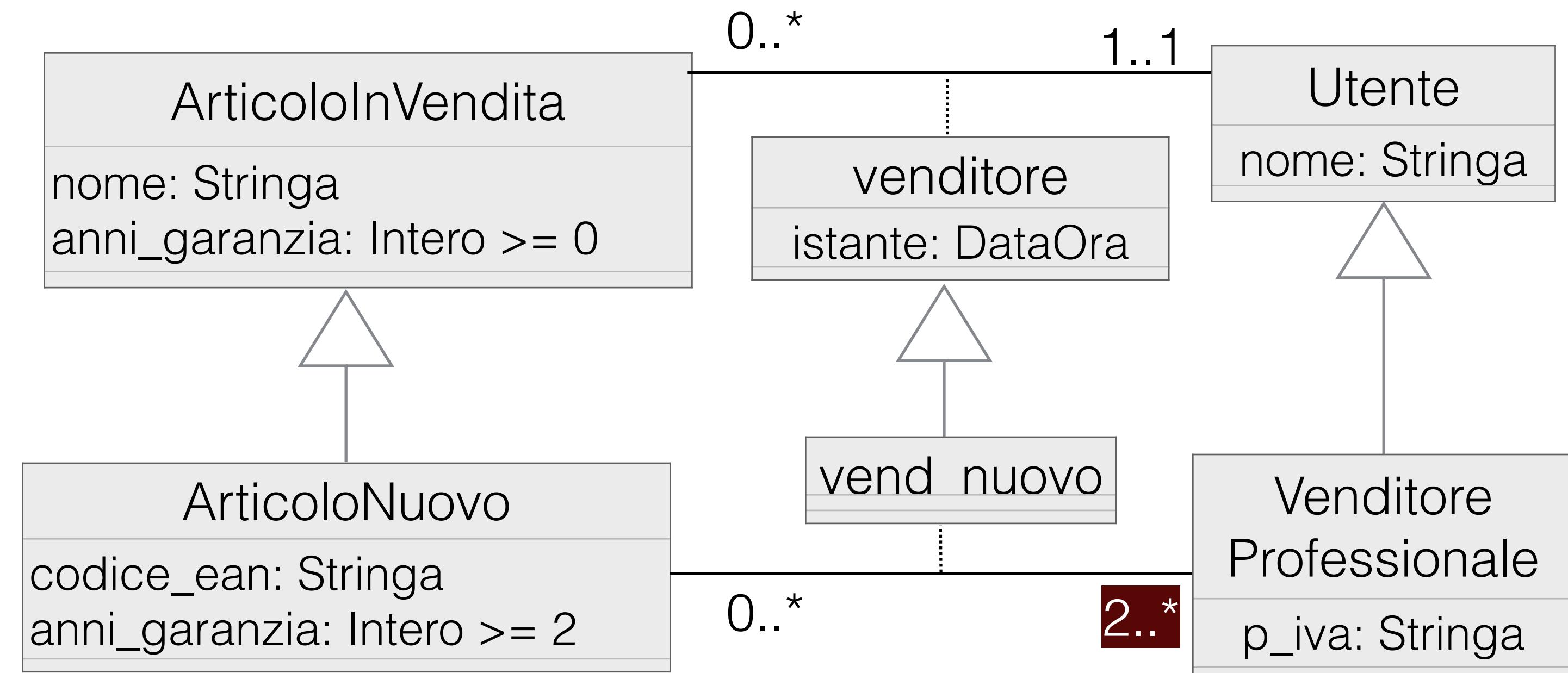
Il vincolo **1..*** è **fuorviante**. In realtà diagramma implica che ogni articolo nuovo è venduto da **un solo** venditore professionale!

Specializzazione di associazioni: condizioni per la correttezza (2)

Le specializzazioni di associazioni necessitano di particolare attenzione relativamente ai vincoli di molteplicità.

Esempio:

- Ogni articolo è venduto da **esattamente un** utente
- Gli articoli nuovi devono essere venduti da **almeno due** venditori professionali



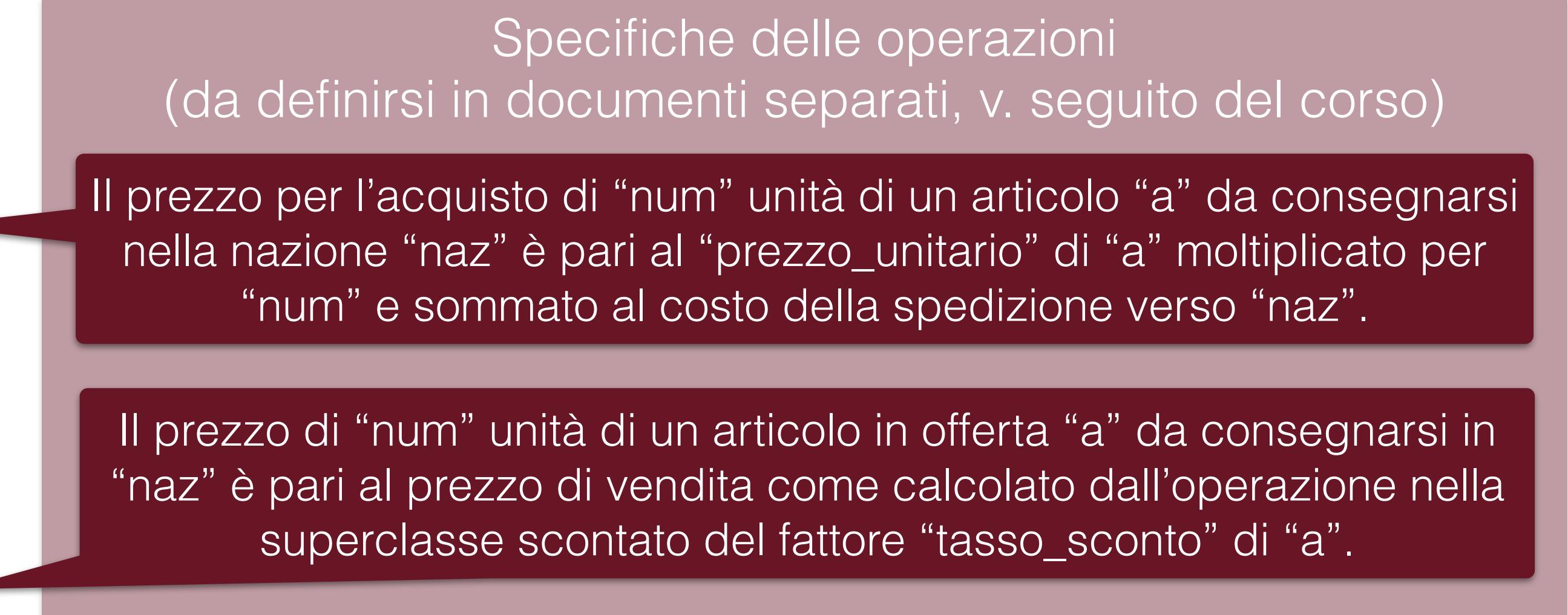
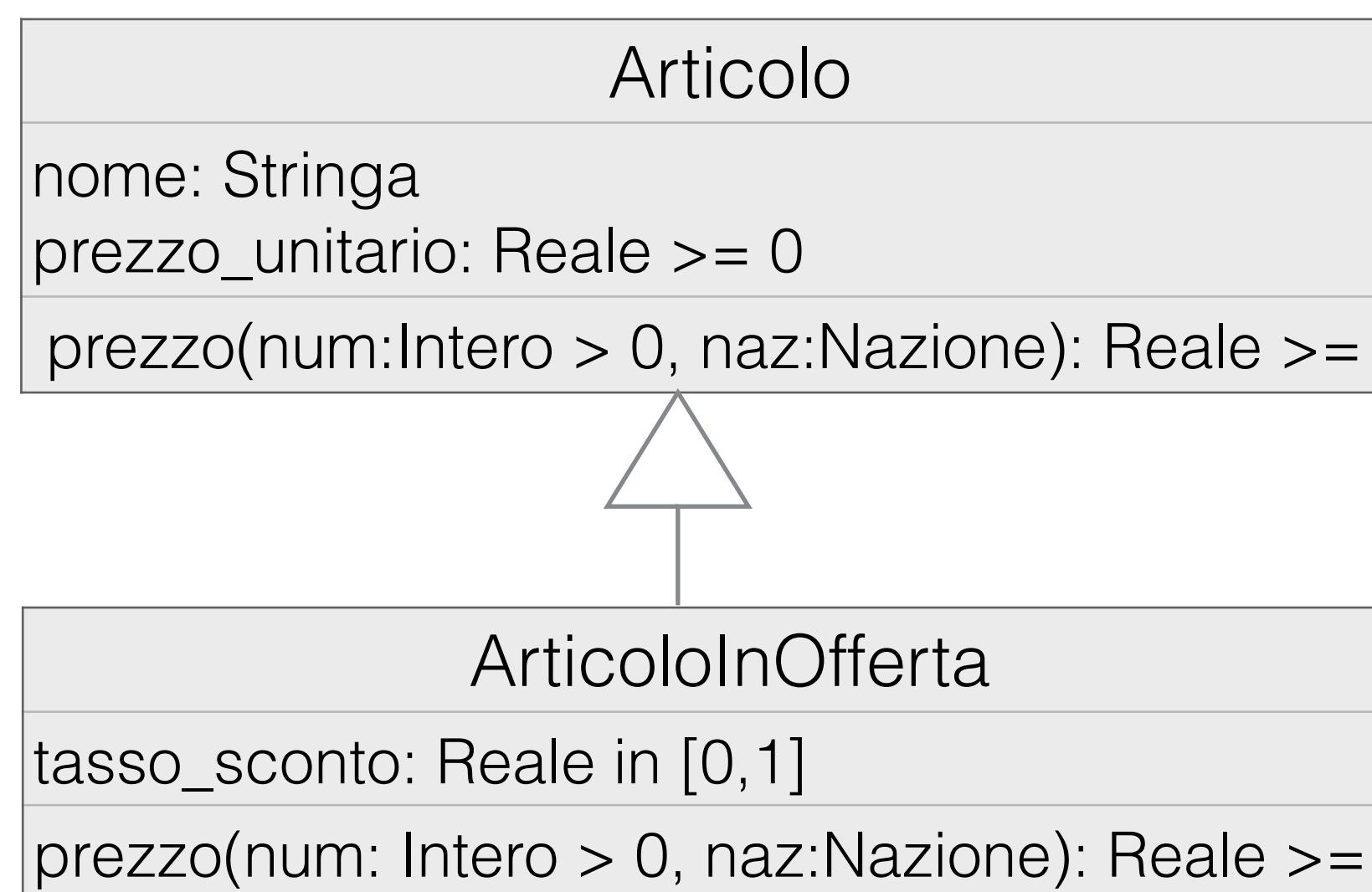
Il vincolo **2..*** comporta che non possano esistere articoli nuovi!

Specializzazione di operazioni di classe

Anche le operazioni di classe possono essere oggetto di specializzazione nelle sottoclassi

Esempio:

- Di tutti gli articoli in vendita va calcolato il prezzo, che è uguale al prezzo unitario moltiplicato per il numero di pezzi richiesto, ed incrementato delle spese di spedizione (che dipendono dalla nazione dove la merce sarà consegnata)
- Alcuni articoli sono in offerta. Il loro prezzo è scontato di un certo fattore di sconto.



Come per la specializzazione di attributi, la segnatura dell'operazione nella sottoclasse deve essere **compatibile** con quella dell'operazione definita nella superasse, ovvero:

- Stesso numero e tipo di argomenti
- Il tipo di ritorno dell'operazione nella sottoclasse è dello stesso tipo o un sotto-tipo di quello dell'operazione nella superclasse.

ITC INFORMATION AND COMMUNICATIONS TECHNOLOGY ACADEMY

MODULO: Progettazione
UNITÀ: Progettazione.1

Prof. Toni Mancini
Dipartimento di Informatica
Sapienza Università di Roma



A.1.2
Analisi dei Requisiti
Unified Modeling Language
Diagrammi degli use-case

Diagramma UML degli use-case

Modellano le **funzionalità** che il sistema deve realizzare, in termini di **use-case** (scenari di utilizzo)

Use-case

Cattura un **insieme omogeneo di funzionalità** accedute da un **gruppo omogeneo di utenti**.

Tipicamente coinvolge concetti rappresentati da più classi e associazioni del diagramma delle classi.

Attore

Ruolo che un utente (umano o sistema esterno) gioca interagendo con il sistema.

Lo stesso utente può essere rappresentato da più attori (può giocare più ruoli).

Più utenti possono essere rappresentati dallo stesso attore.

Diagramma UML degli use-case

Un diagramma UML degli use-case è un grafo in cui:

- i **nodi** rappresentano attori e use-case
- gli **archi** rappresentano:
 - la possibilità per un attore di invocare uno use-case
 - la possibilità per uno use-case di invocare un altro use-case
 - la generalizzazione tra attori e tra use-case

Definizione: cosa è un grafo?

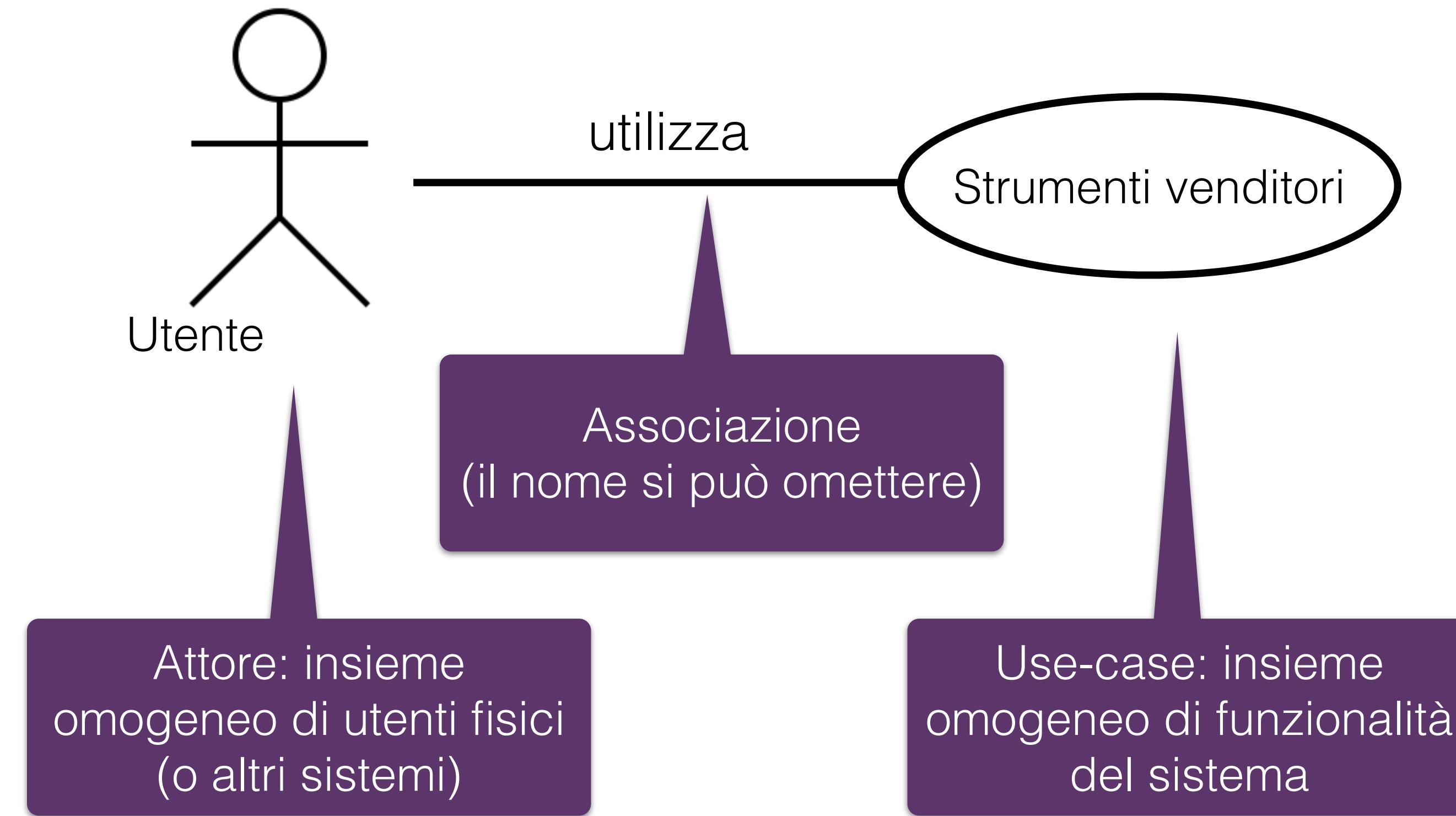
Una struttura molto comune in informatica. Rappresenta una **rete** di elementi, chiamati **nodi**, collegati a coppie da **archi**

- Esempio: mappa di una rete di metropolitane:



Diagramma degli use-case: associazione

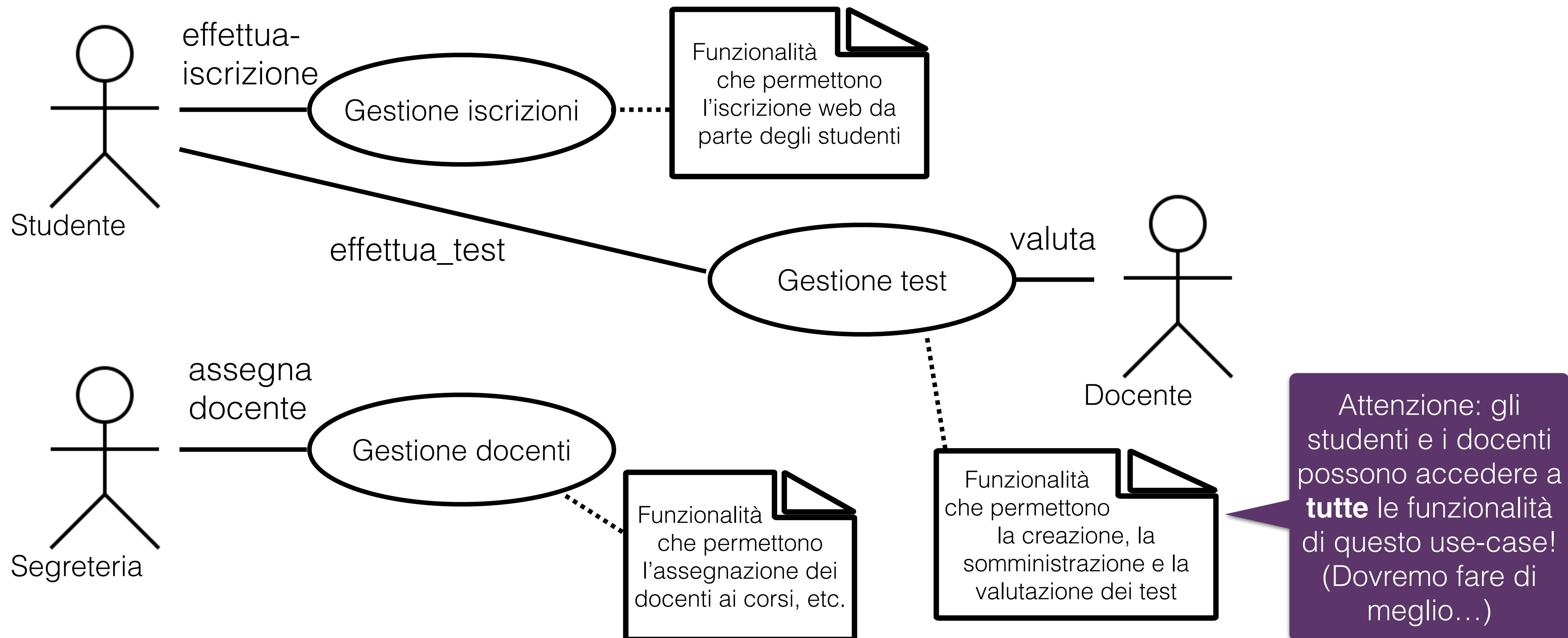
- Modella la possibilità di accesso, da parte di un attore, alle funzionalità di uno use-case



Attenzione. L'esistenza dell'attore Utente non implica l'esistenza della classe Utente nel diagramma delle classi. Avremo la classe Utente **solo se** il sistema deve rappresentare **dati** sugli utenti.

Esempio

Il sistema deve permettere agli studenti di iscriversi, via web, ai corsi offerti. La segreteria deve poter assegnare i docenti ai singoli corsi. I docenti devono poter inserire i risultati dei test degli studenti: tali test sono somministrati agli studenti utilizzando il sistema.

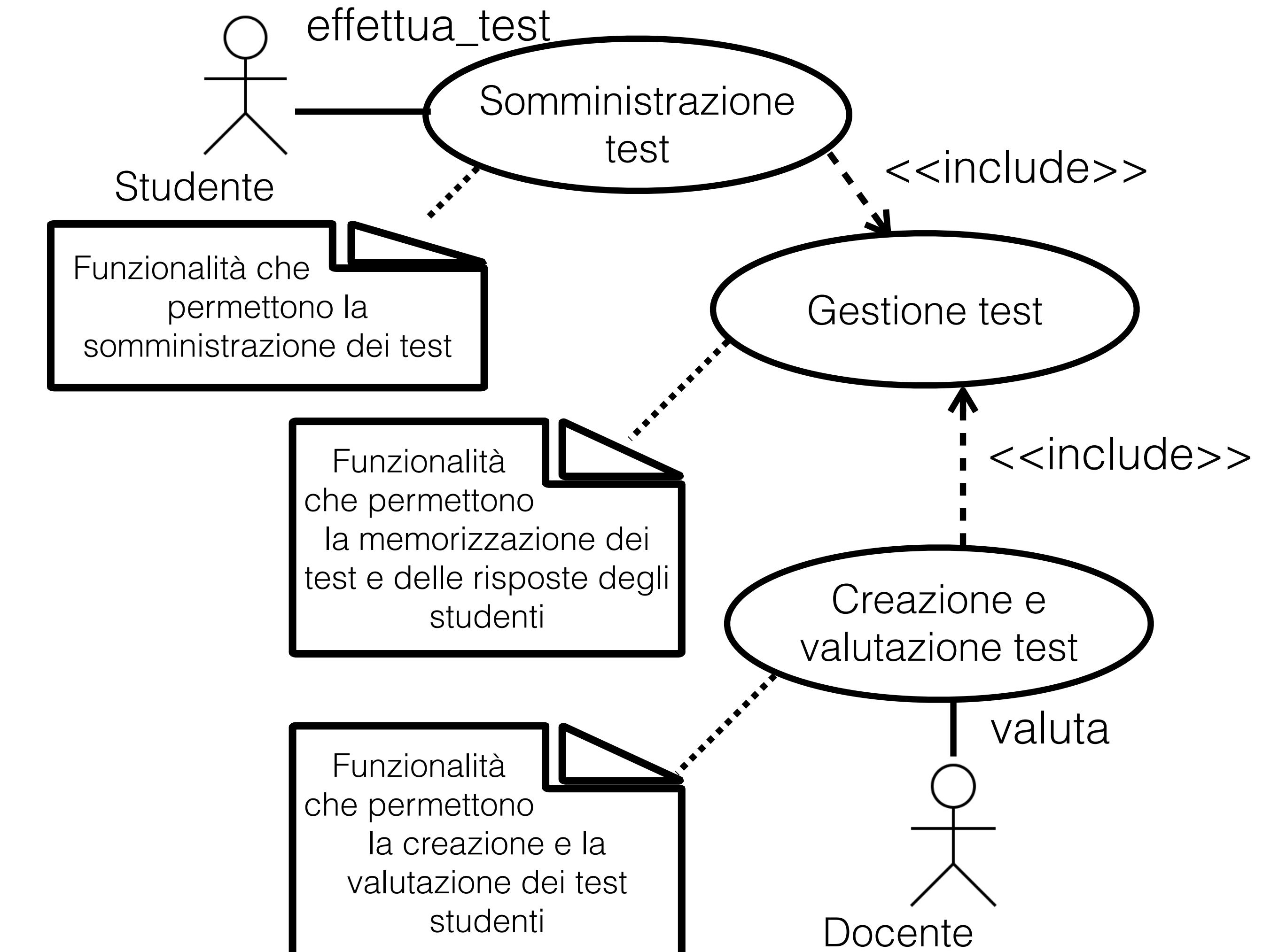


Dipendenze tra use-case: inclusione

- Alcune funzionalità dello use-case A hanno bisogno di usare alcune funzionalità dello use-case B

Esempio:

- i docenti possono creare e valutare i test degli studenti
- gli studenti possono rispondere ai test
- i test e le risposte degli studenti vanno memorizzati nel sistema

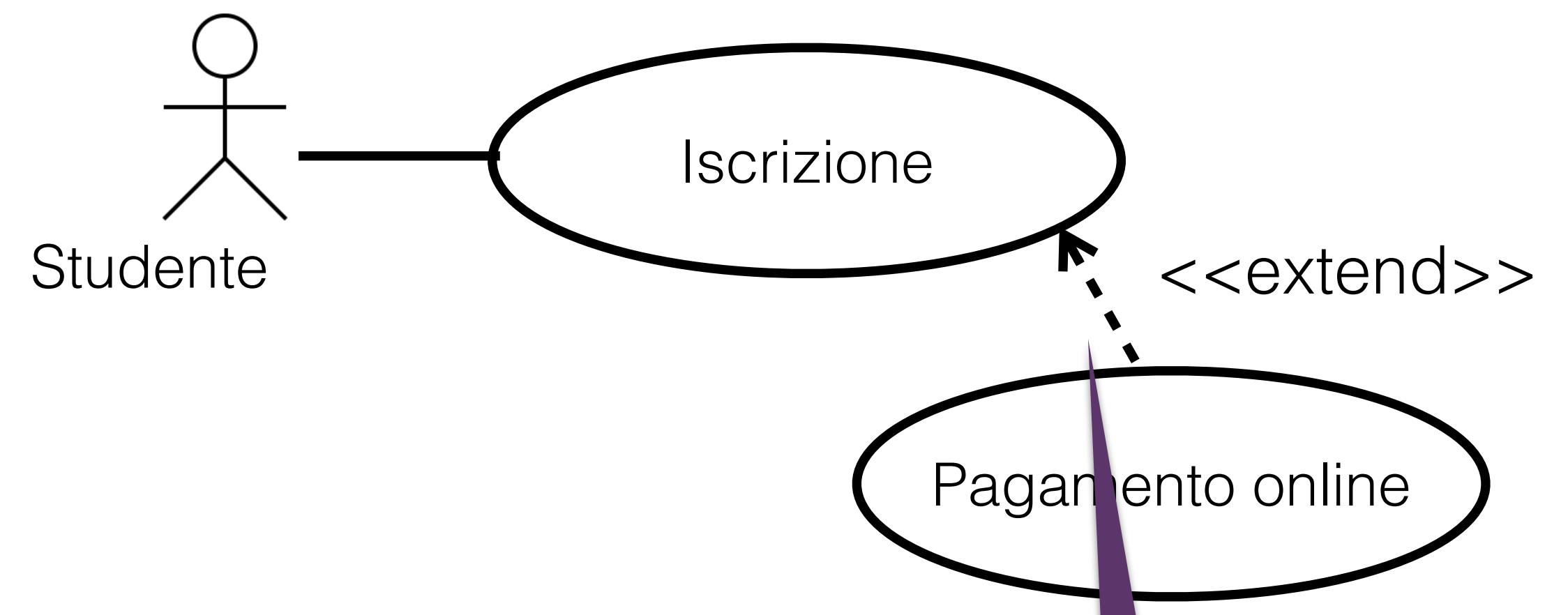


Dipendenze tra use-case: estensione

- Alcune funzionalità dello use-case A, solo in alcuni casi particolari, sono estese con le funzionalità dello use-case B

Esempio:

- Gli studenti possono iscriversi a corsi
- Durante il processo di iscrizione, gli studenti possono optare per il pagamento online



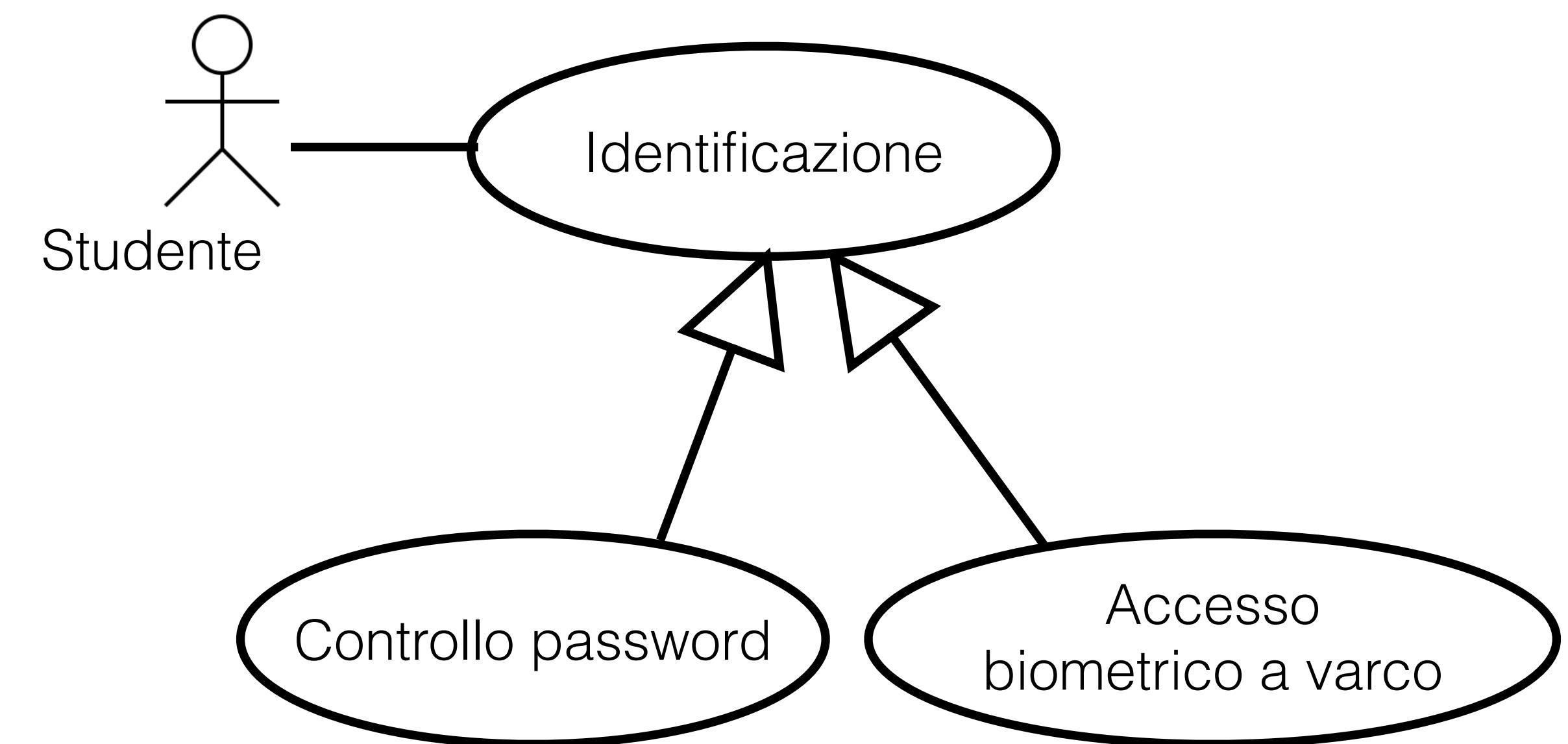
Nota il verso della dipendenza: dallo use-case che estende (B) allo use-case esteso (A)

Generalizzazione tra use-case

- Alcune funzionalità dello use-case A, solo in alcuni casi particolari, sono rimpiazzate con le funzionalità dello use-case B

Esempio:

1. Gli studenti devono potersi identificare
2. L'identificazione online avviene tramite password
3. La registrazione delle presenze ai corsi avviene tramite impronta digitale/iride scansionata dal lettore del tornello



Nota: nei diagrammi degli use-case non possiamo usare generalizzazioni uniche che coinvolgono più sotto-usecase, né tantomeno vincoli {disjoint} e {complete}

Generalizzazione tra attori

- L'attore B può fare le veci dell'attore A, e ne eredita tutte le associazioni
- Esempio: i manager possono fare le veci della Segreteria, ed accedere a tutti gli use-case accessibili dalla Segreteria

Attenzione. Il diagramma non implica che esistano le classi Segreteria e Manager nel diagramma delle classi, né tantomeno che la classe Manager sia una sottoclasse di Segreteria

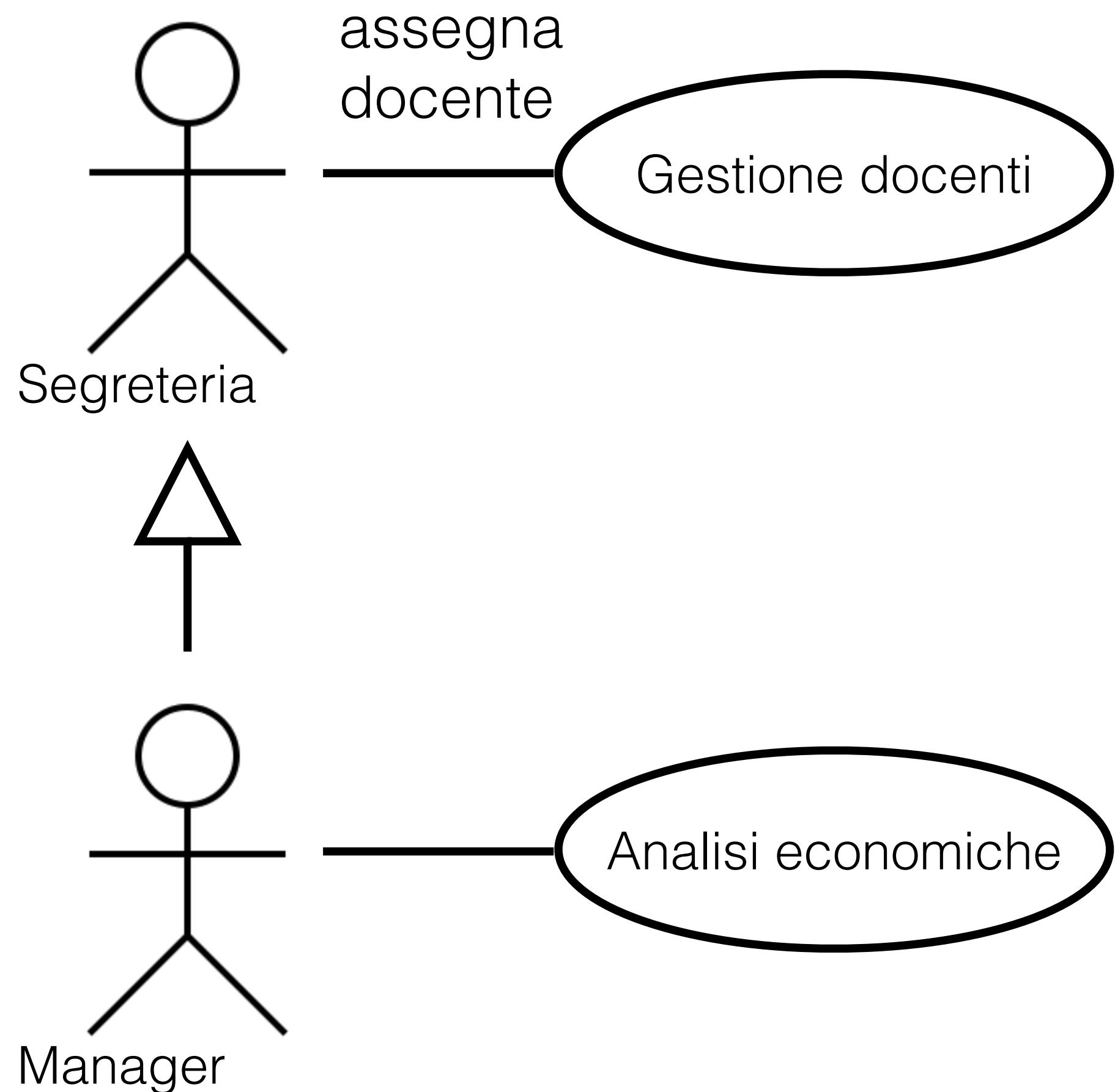


Diagramma degli use-case: specifiche

- Il diagramma degli use-case è molto semplice, e dà solo una visione di alto livello di:
 - quali attori possono usare il sistema
 - quali macro-funzionalità sono accessibili ai diversi attori
- Definisce inoltre come le diverse macro-funzionalità vadano modularizzate
- Si tratta di un diagramma facilmente comprensibile anche al committente
- Il diagramma **non** definisce le singole operazioni all'interno di ogni use-case
- Ogni use-case del diagramma andrà affiancato da un **documento di specifica** che entra nel dettaglio (v. seguito)