Sistemi Informativi:

La specifica delle relazioni

Sommario

vari tipi di relazione tra classi ed oggetti

- associazioni
- aggregazioni
- generalizzazioni
- dipendenze

Relazioni:Legami e Associazioni

Legami (links) ed associazioni stabiliscono relazioni semantiche tra elementi del modello

Associazione:

- una relazione tra **classi**
- descrive un insieme di relazioni tra oggetti di classi diverse

Link:

- Una connessione tra un insieme **oggetti**. Un link e' come una tupla.
- Un link e' una **istanza** di una **associazione**

Links

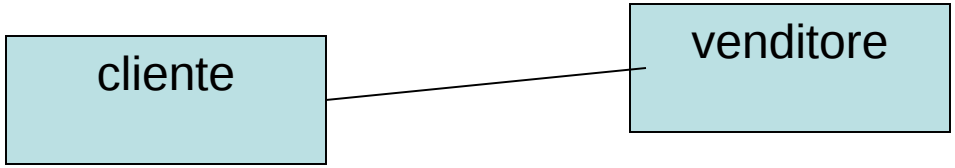
Un link e’ una connessione tra oggetti che comunicano tramite messaggi.Sono necessari perche’ un oggetto possa reperire l’altro al momento dell’esecuzione. I link sono dinamici, possono cambiare nel tempo. In pratica e' l'indirizzo.

unidirezionali se solo un oggetto comunica con l’altro.

Freccia verso il ricevente

bidirezionali se entrambi comunicano.

Senza freccia



Ruolo

e’ il nome che identifica un partecipante di una associazione o di un link

scritto vicino alla classe che riveste quel ruolo

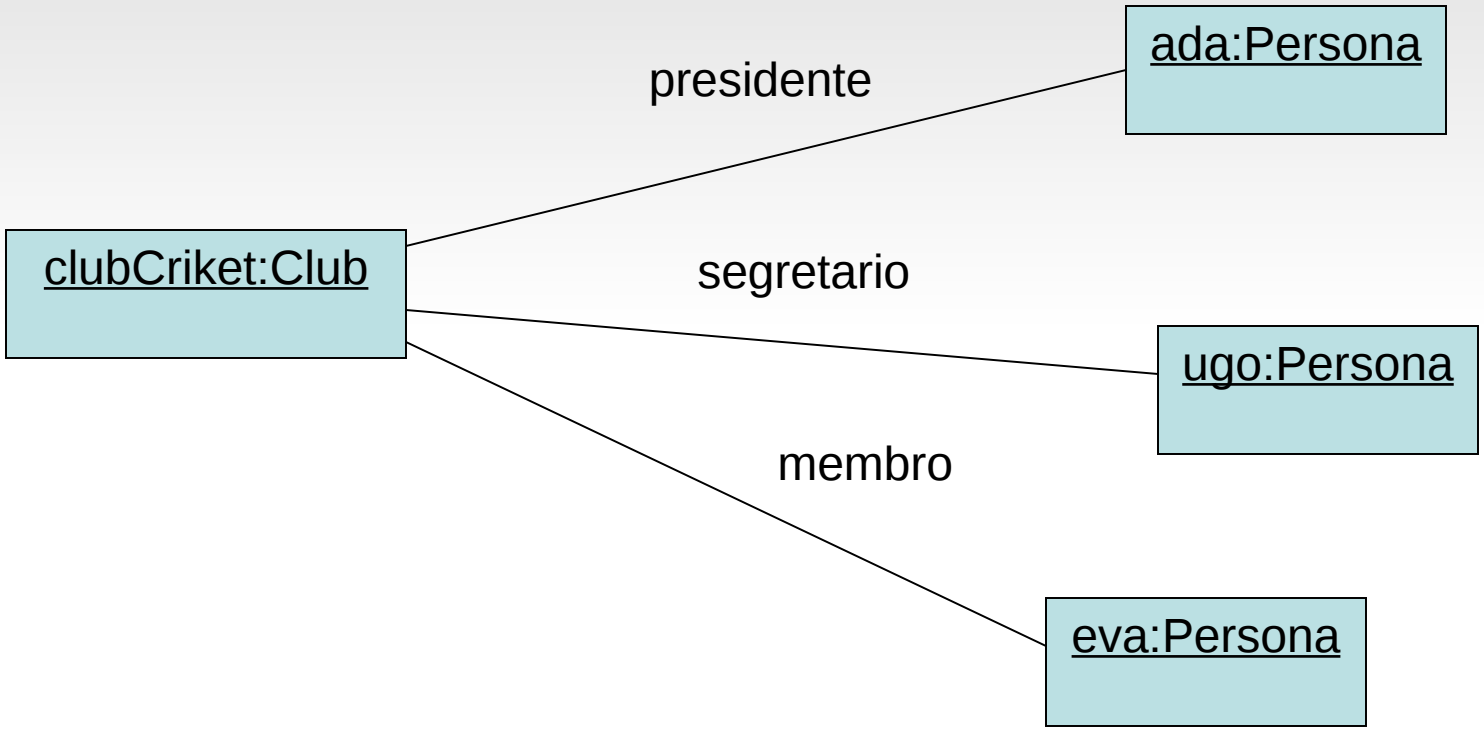
Uso:

- necessario per associazioni fra oggetti della stessa classe
- utile per distinguere tra associazioni multiple fra classi diverse

Quando non usarli

- se esiste una associazione sola tra due classi, I loro nomi sono gia’ adatti a diventare nomi per ruoli

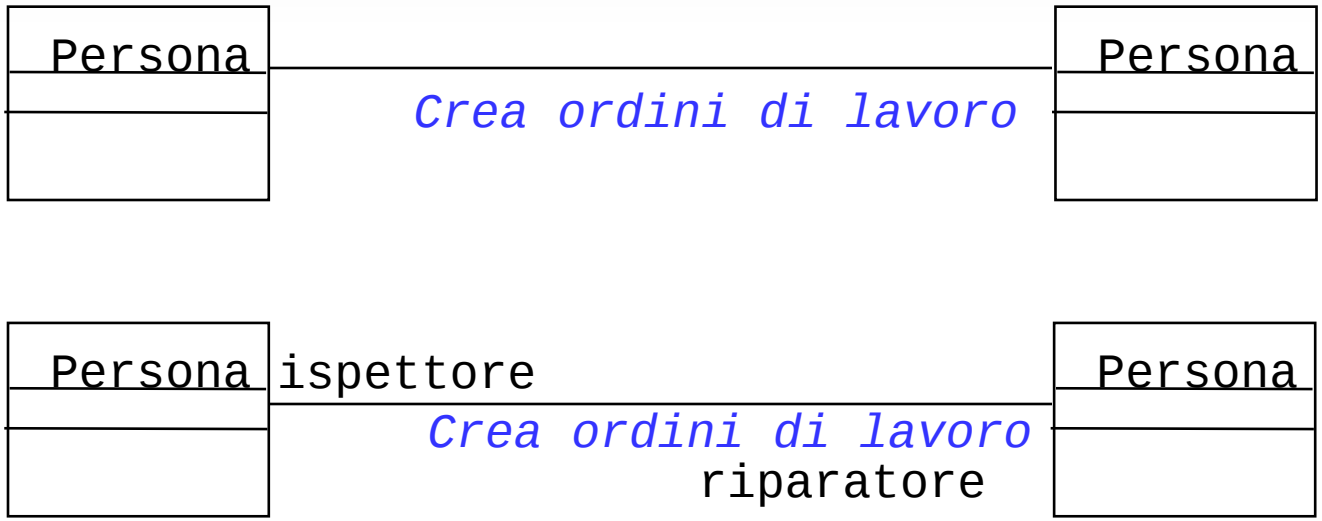
Esempio di ruolo



Esempio di ruolo

problema

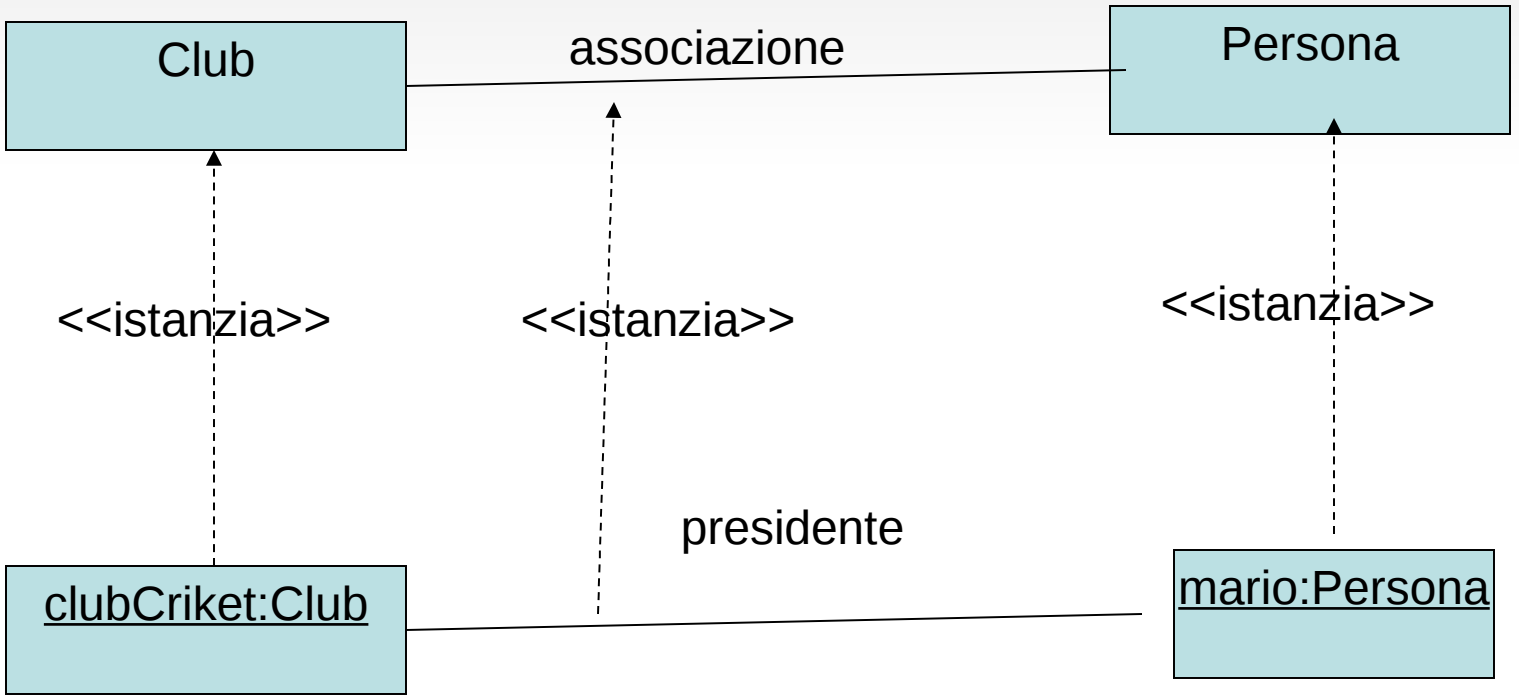
Una persona assume il ruolo di riparatore presso un altra, che assume a sua volta il ruolo di ispettore



Associazioni

sono relazioni tra classi

I link dipendono dalle associazioni e le istanziano



Proprieta’ delle Associazioni

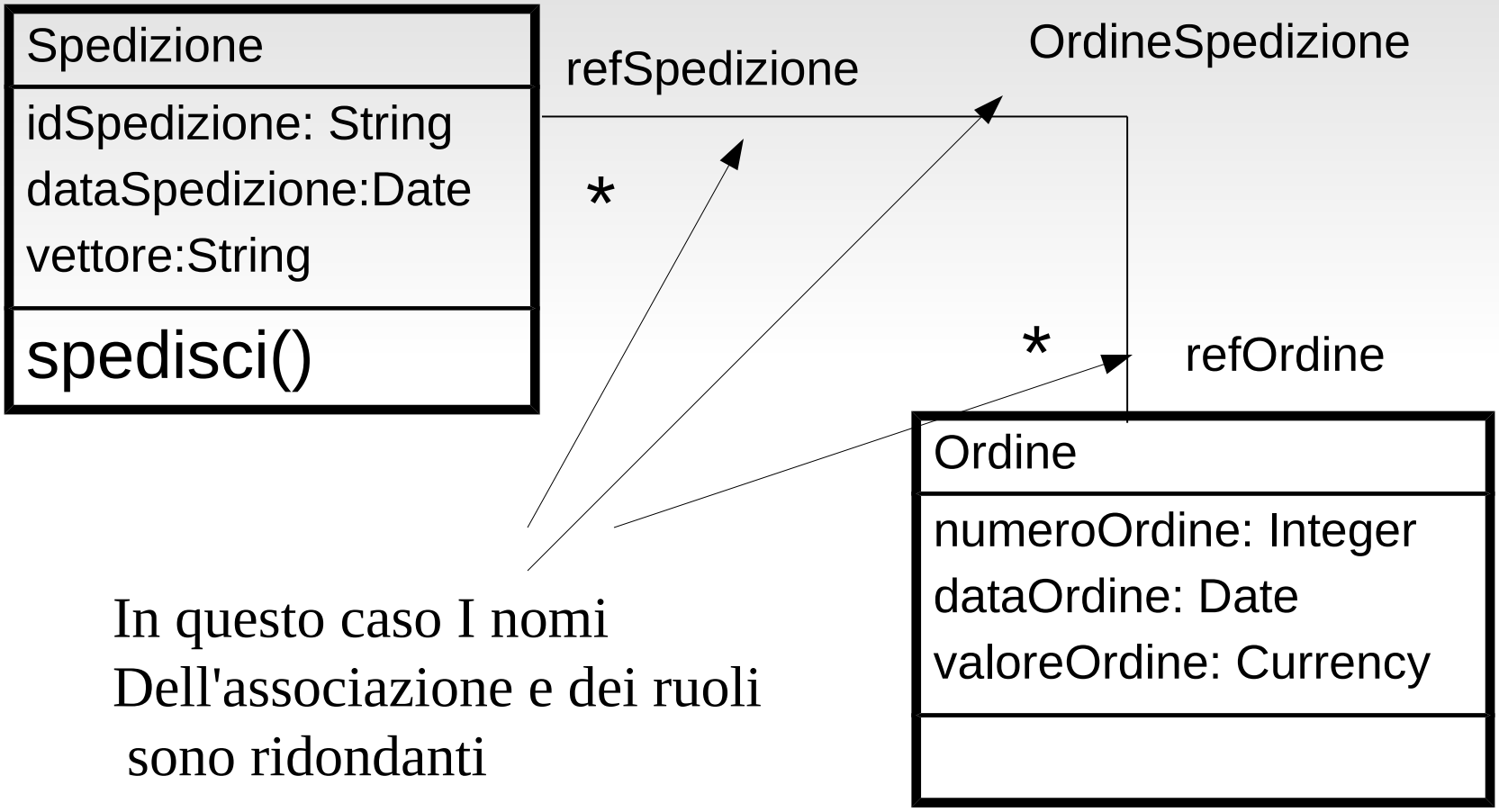
Nome (verbo)

Ruoli

Cardinalita’

Navigabilita’

Associazioni



In questo caso I nomi
Dell'associazione e dei ruoli
sono ridondanti

Ruoli nelle associazioni

Client Role:

- Un oggetto puo' operare su altri ma non accade il contrario.

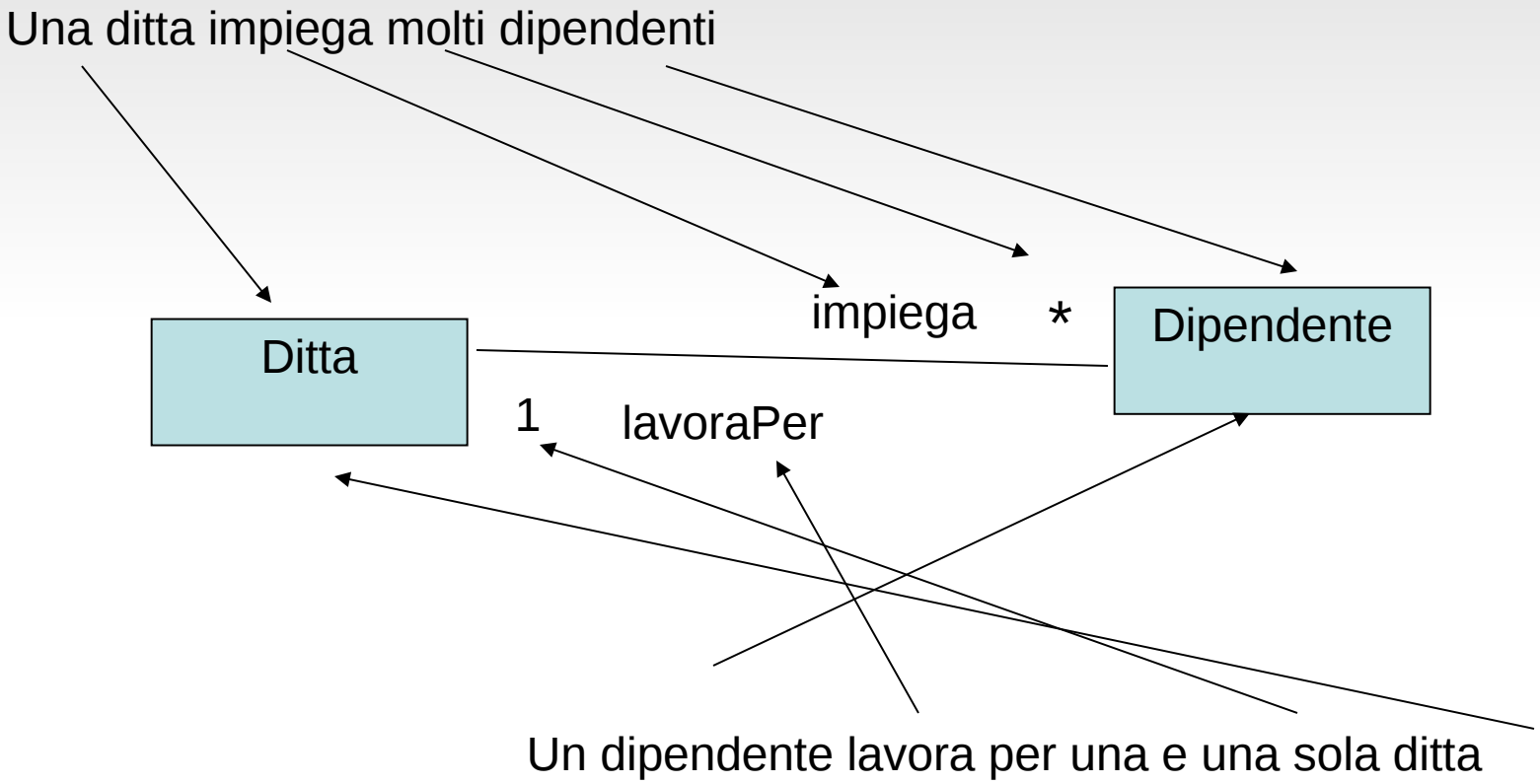
Server Role:

- l'opposto.

Agent Role:

- puo' fare da client e server.

Cardinalita’ nelle associazioni



Cardinalita’ delle associazioni

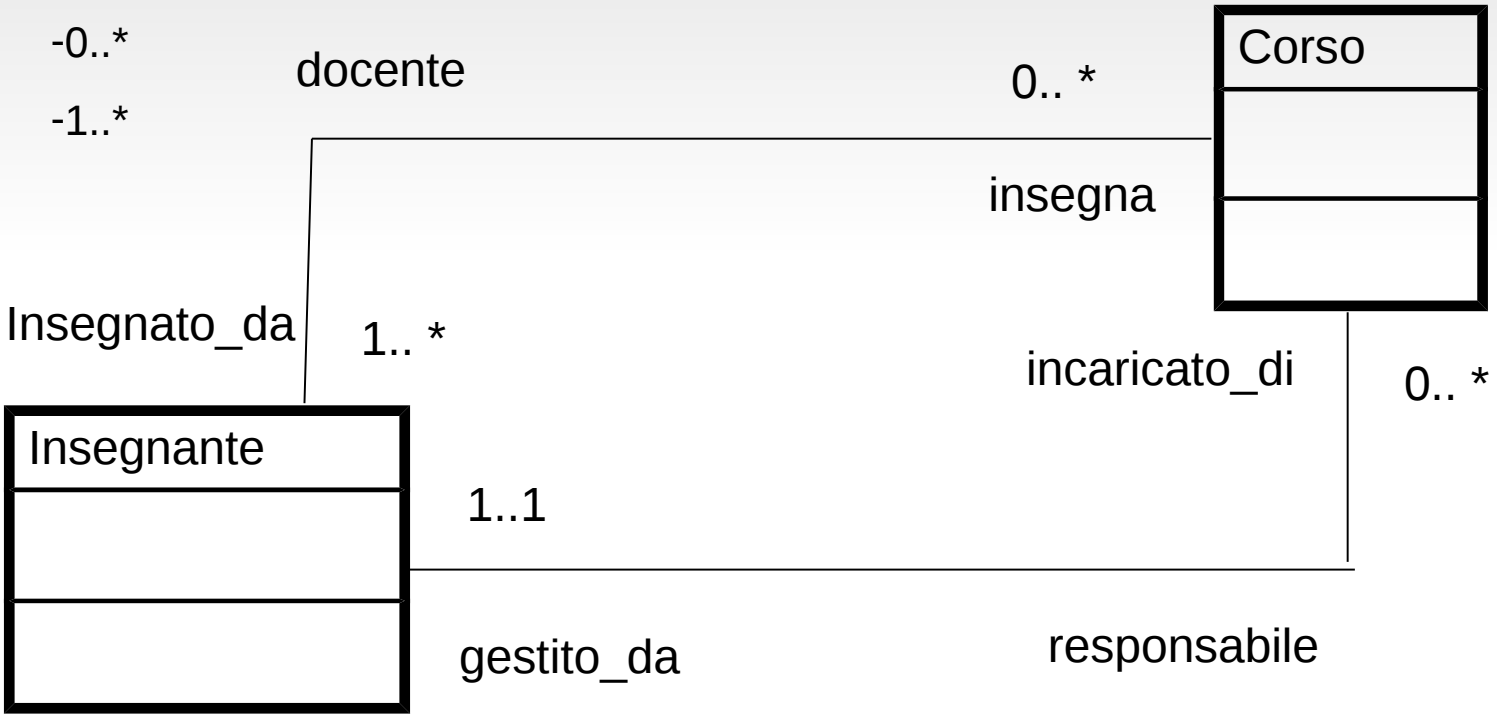
(limite inferiore .. limite superiore) **inclusi**

esempi comuni

- 0..1 da 0 a 1
- 1 uno e un solo
- 0..* zero o piu’
- * zero o piu’
- 1..* almeno 1

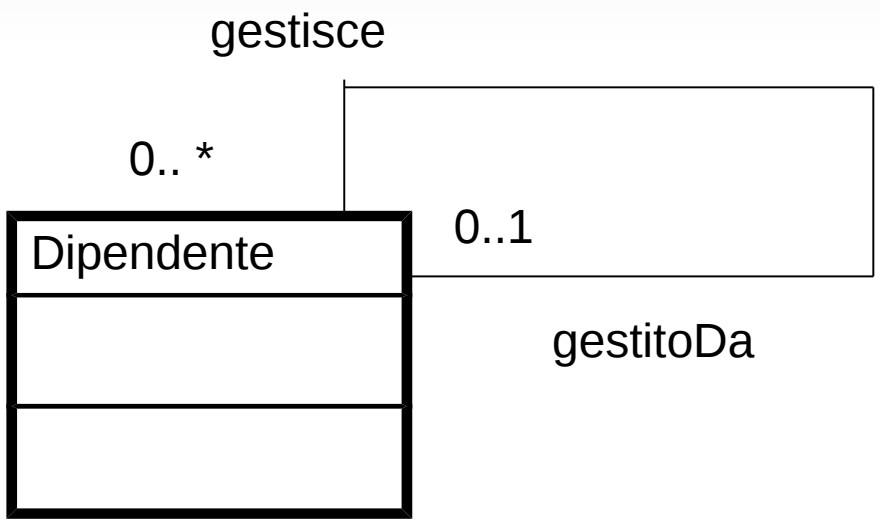
Cardinalita’ delle associazioni

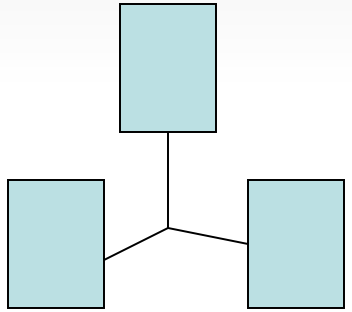
- 0..1
- 1..1
- 0..*
- 1..*



Grado delle associazioni

- Unaria
- Binaria
- Ternaria

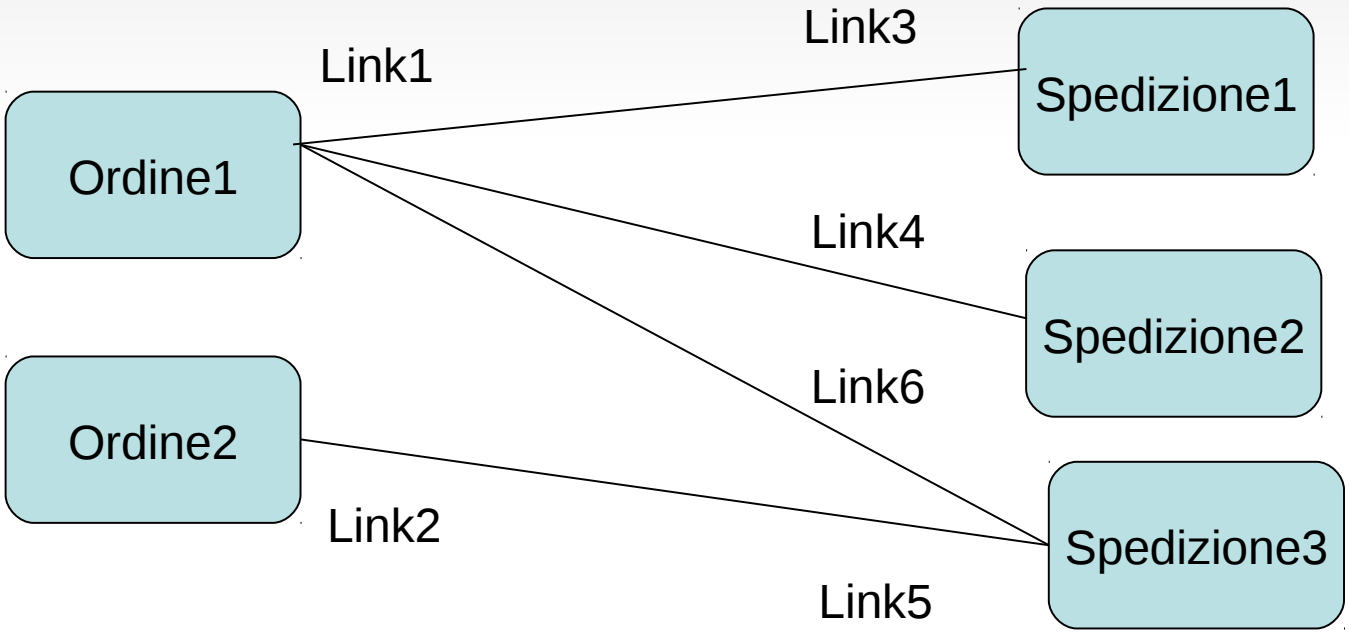




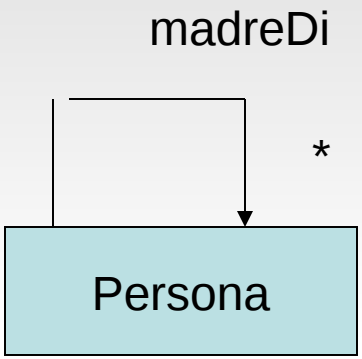
Link e dimensione delle associazioni

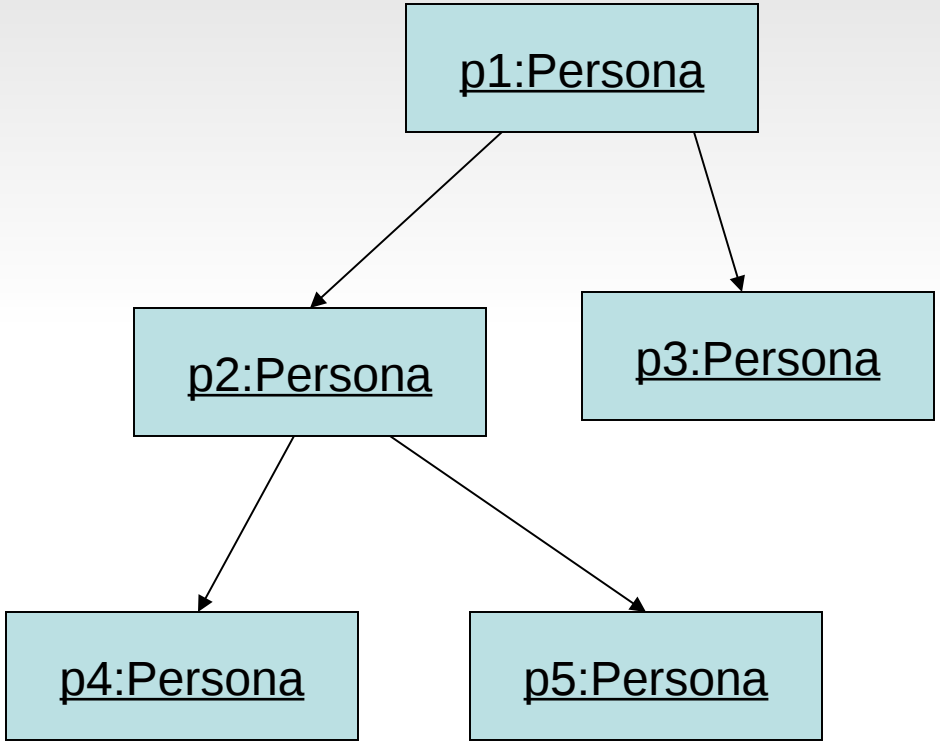
Link: istanza di associazione

Estensione di una associazione : insieme di istanze di una associazione



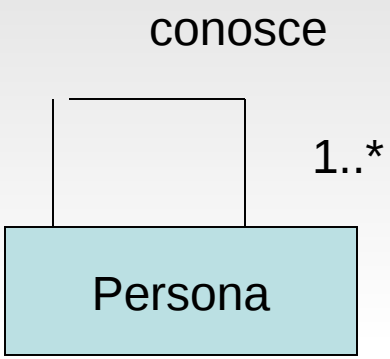
Associazioni che creano gerarchie di oggetti





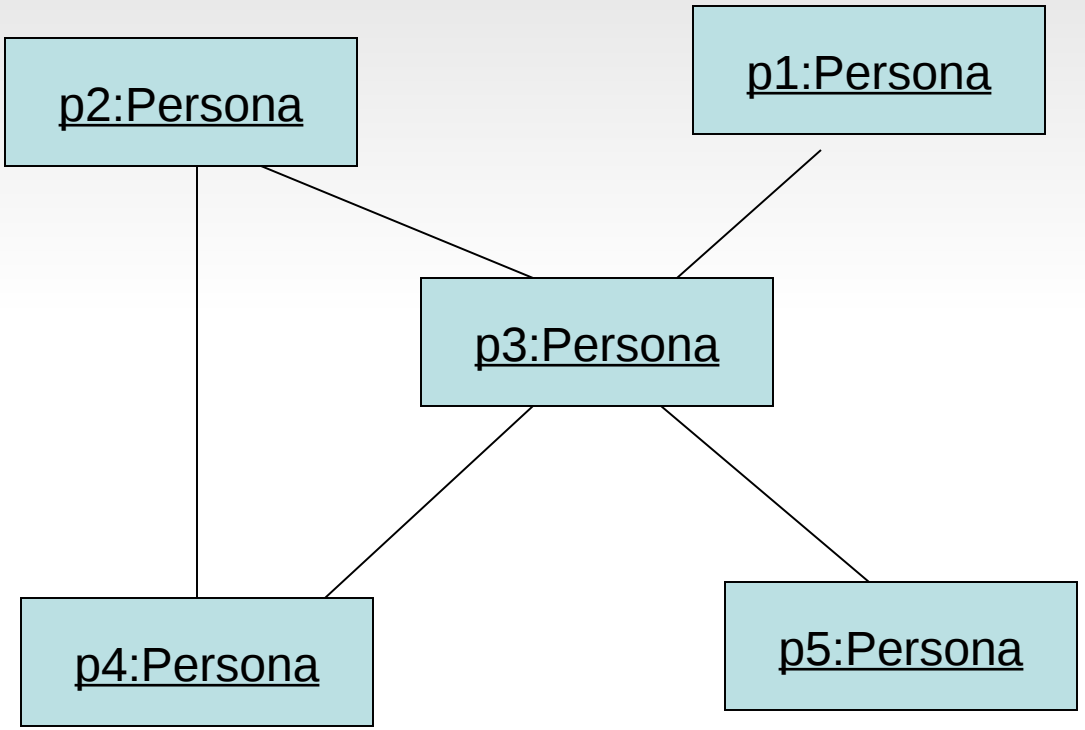
Navigabilita': in questo caso si assume che la madre conosce I suoi figli: e' vero il contrario?

Associazioni che creano reti di oggetti



associazione bidirezionale simmetrica.

perche' partiamo da 1 e non da 0?



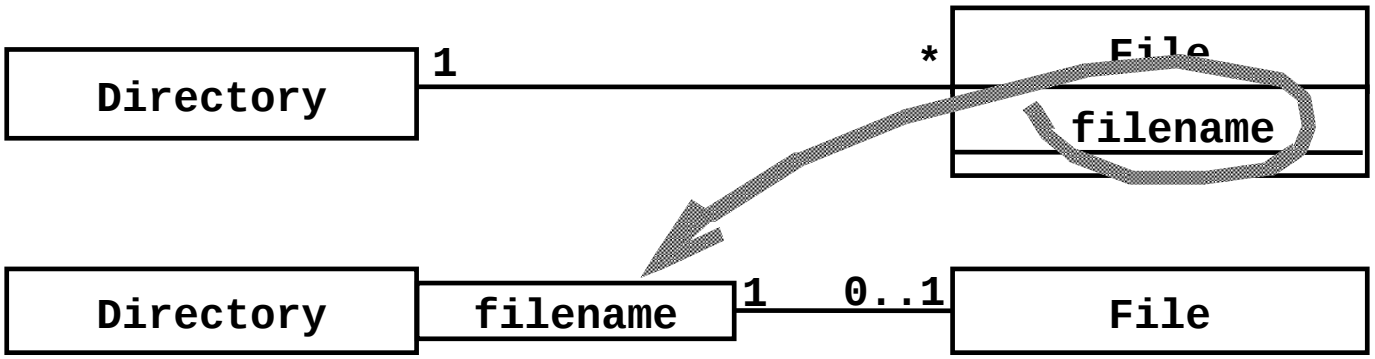
Qualificare un'associazione

Il qualificatore e' una notazione che migliora l'informazione sulla molteplicita' di una associazione

Specifica un attributo che individua il destinatario

usato per ridurre da 1..N a 1..1

Senza qualificazione: A Una cartella ha molti files. Un file appartiene ad una cartella

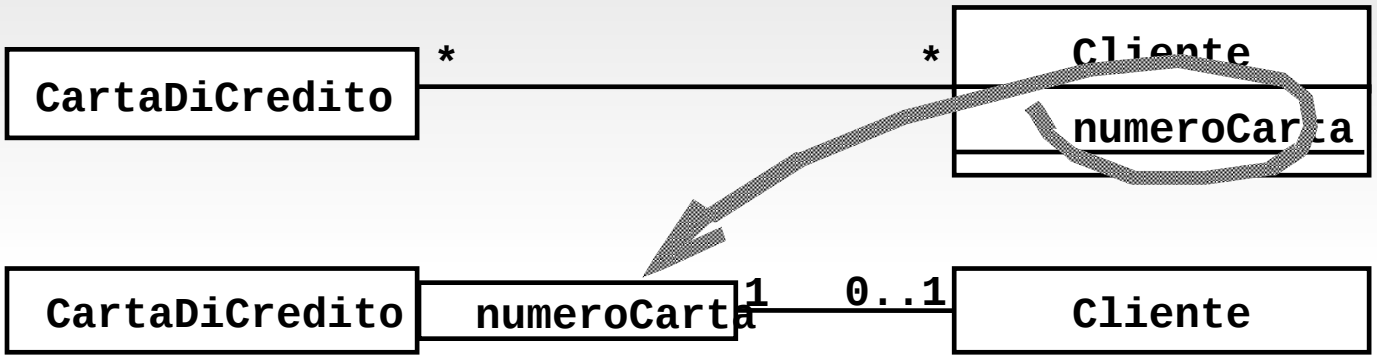


Con qualificazione: Una cartella ha molti file, ognuno con nome unico,

Esempio

Una societa' di Carta di Credito ha molti clienti. ogni cliente puo' avere molte carte di credito con numeri diversi

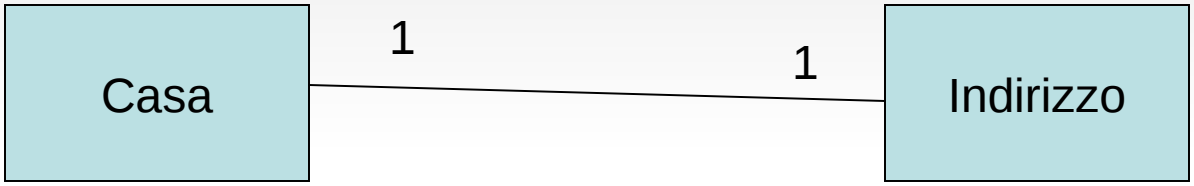
Riduzione della cardinalita' tramite qualificazione



la qualificazione non e' una proprieta' tradotta automaticamente nell'implementazione, e' invece una indicazione di come a runtime si naviga l'associazione per individuare l'oggetto esatto in un insieme di possibili candidati

Modellare le associazioni come attributi

le associazioni 0..1 , 1..1 si possono modellare come attributi (referenze)

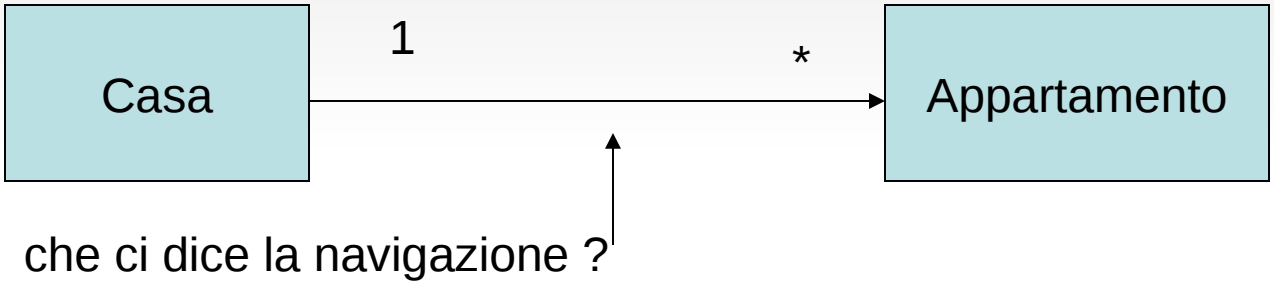


Casa
indirizzo:Indirizzo

Indirizzo
casa:Casa

Modellare le associazioni come attributi

le associazioni * si possono modellare come gruppi di istanze (Liste, Insiemi..)

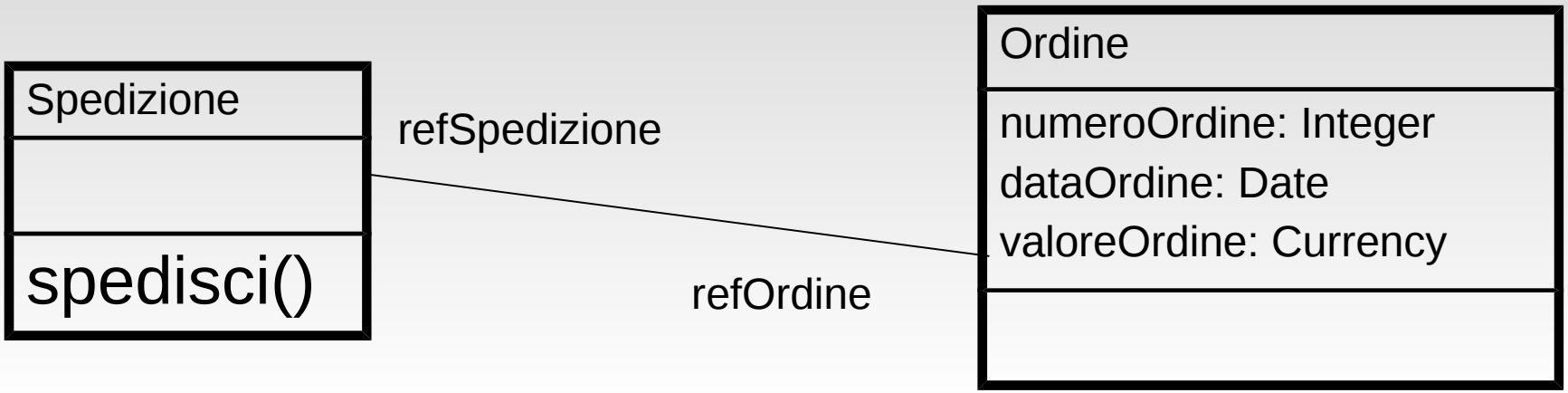


Casa
appartamenti: List(Appartamento)

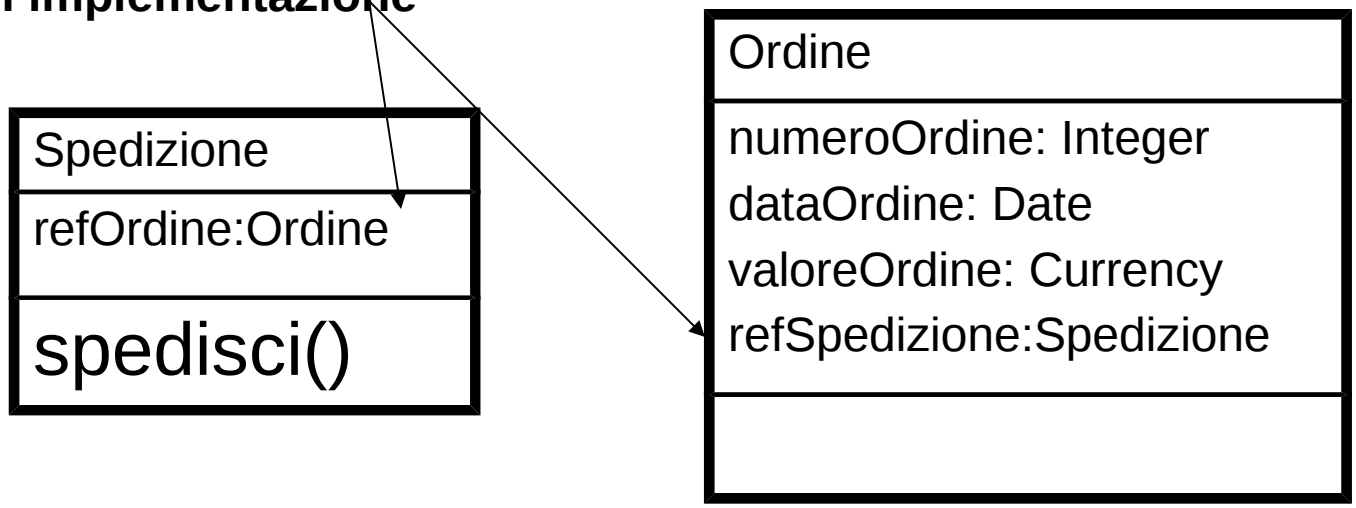
Appartamento

Attributi come referenze a classi

in UML



in implementazione



confronto con relazionale: chiavi

Docente
nome: “MarinoSegnan“ corso:”SistemiInformativi” telefono:”012345678”

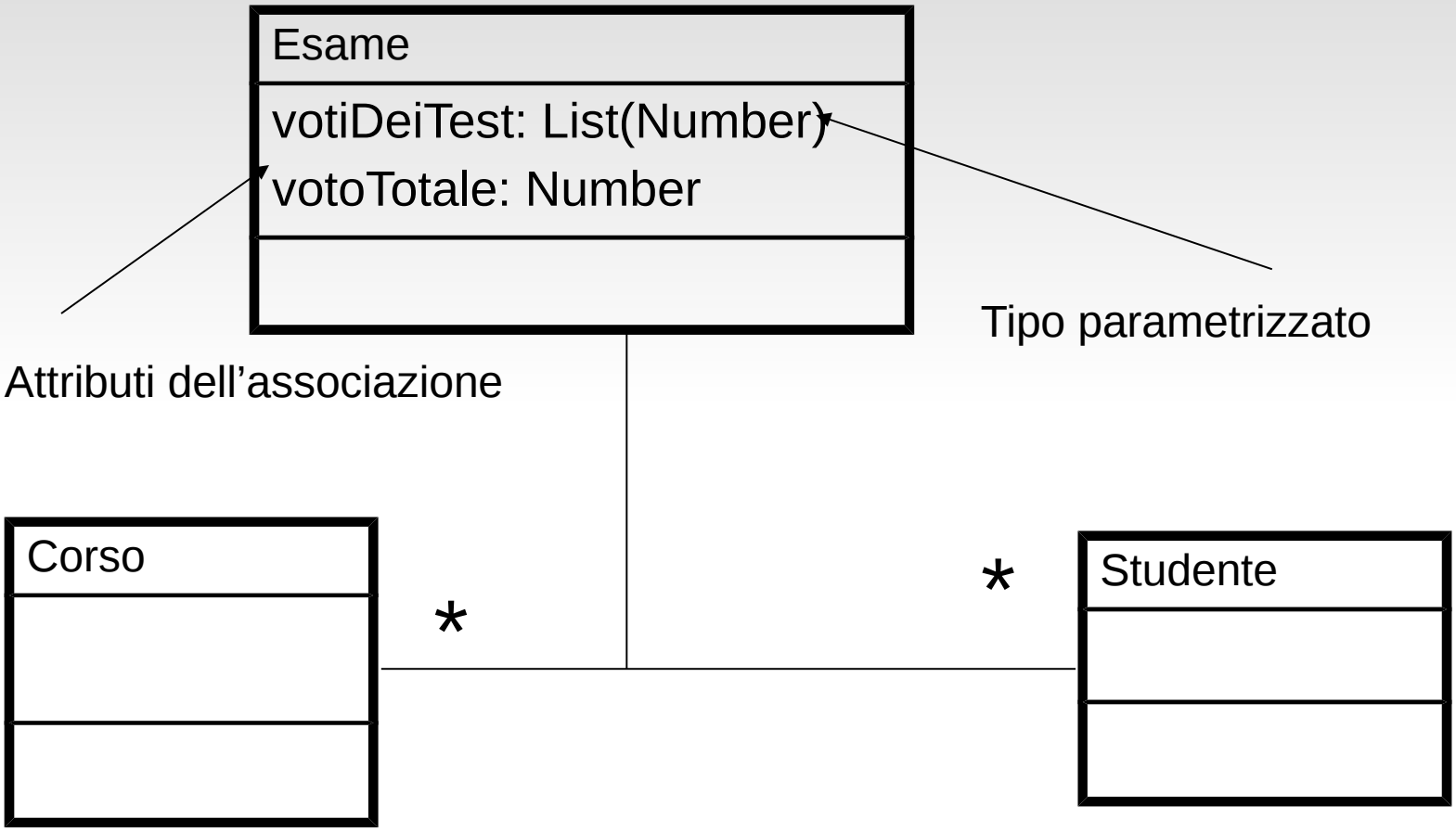
Corso
nome:”SistemiInformativi” docente: “MarinoSegnan“ crediti: 12

Nel modello relazionale le due istanze di Docente e Corso non hanno un link diretto; hanno dei **campi (colonne)** in comune.
La navigazione fra tabelle si fa tramite “join” per poterle collegare.
Le associazioni sono costruite a richiesta. (**Vantaggi**?)
nel modello ad oggetti abbiamo riferimenti diretti

Docente
nome: “MarinoSegnan“ refCorso: Corso telefono:”012345678”

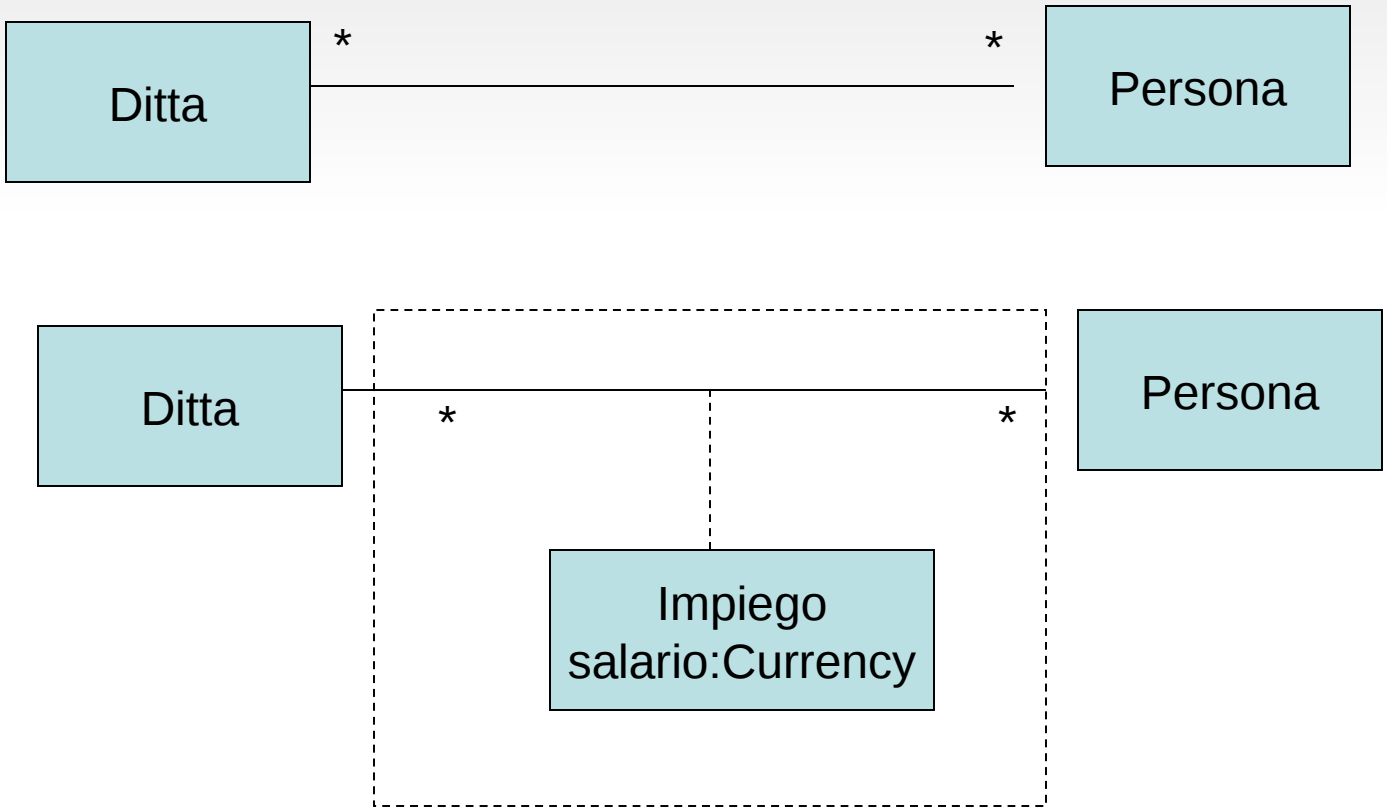
Corso
nome:”SistemiInformativi” refDocente: Docente crediti: 12

Associazioni come classi



Associazioni N..N

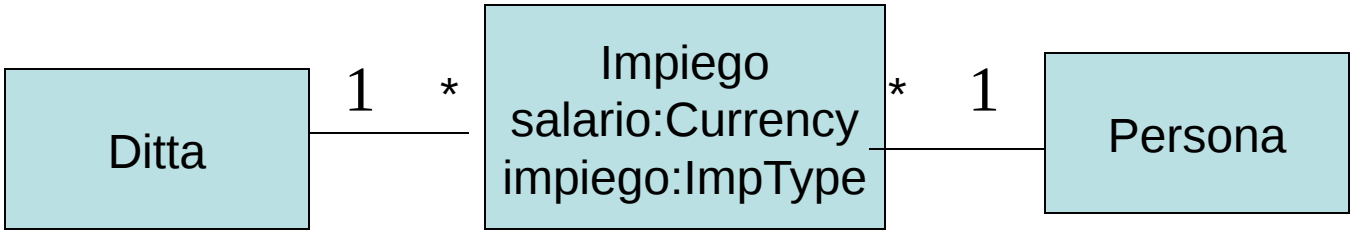
si possono modellare introducendo una classe associativa se esiste una singola istanza per coppia di oggetti (impiego)



Associazioni N..N

se avessimo piu' impieghi con la stessa ditta introduciamo una classe vera e propria (associazione **reificata**). In tal caso possiamo avere tante istanze quante vogliamo di Impiego per ogni coppia Ditta-Persona.

In sostanza , se abbiamo attributi che non appartengono ad una delle classi esistenti, creiamo una nuova classe!



Riepilogo specifica delle associazioni

Nominare le associazioni

Suggerimento:lettere minuscole, parole separate da _

Nominare i ruoli delle associazioni

Individuare la cardinalita’

- Inizialmente si puo’ omettere

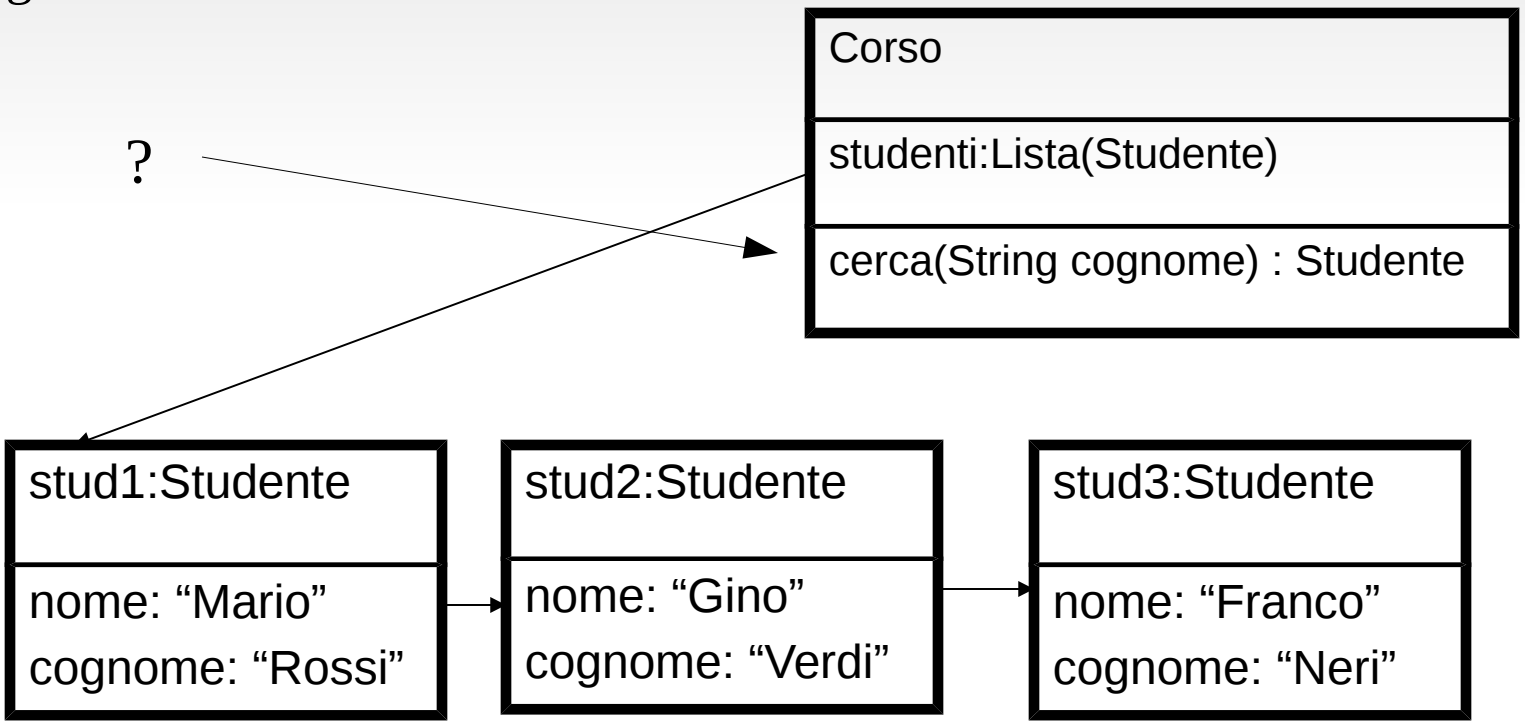
Associazioni ricorsive (sullo stesso insieme)

Implementare associazioni

se si guarda il livello implementativo le cose sono semplici:

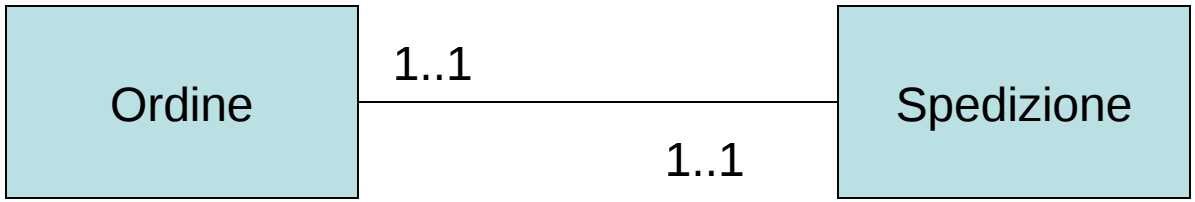
le associazioni 1 a molti si realizzano tramite Liste, Insiemi etc.

per trovare un elemento nella associazioni a molti, si usa un attributo che li distingua es.



IMPORTANTE

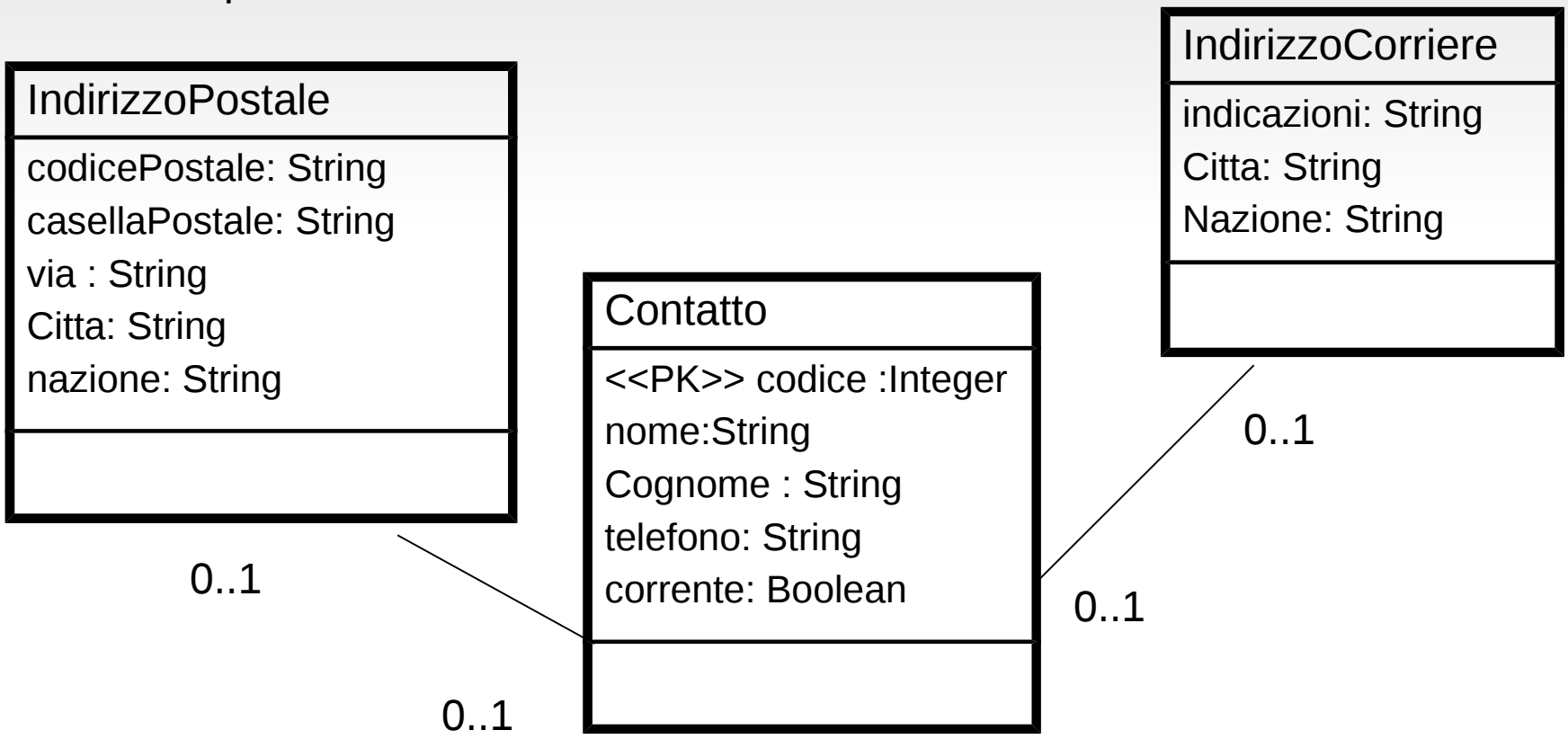
Ogni volta che definiamo le proprieta' di una associazione, introduciamo dei vincoli nel sistema. Occorre evitare per quanto possibile che essi siano inutilmente restrittivi: inserirli piuttosto a parte nel codice (piu' modificabile che la struttura dati). Es.



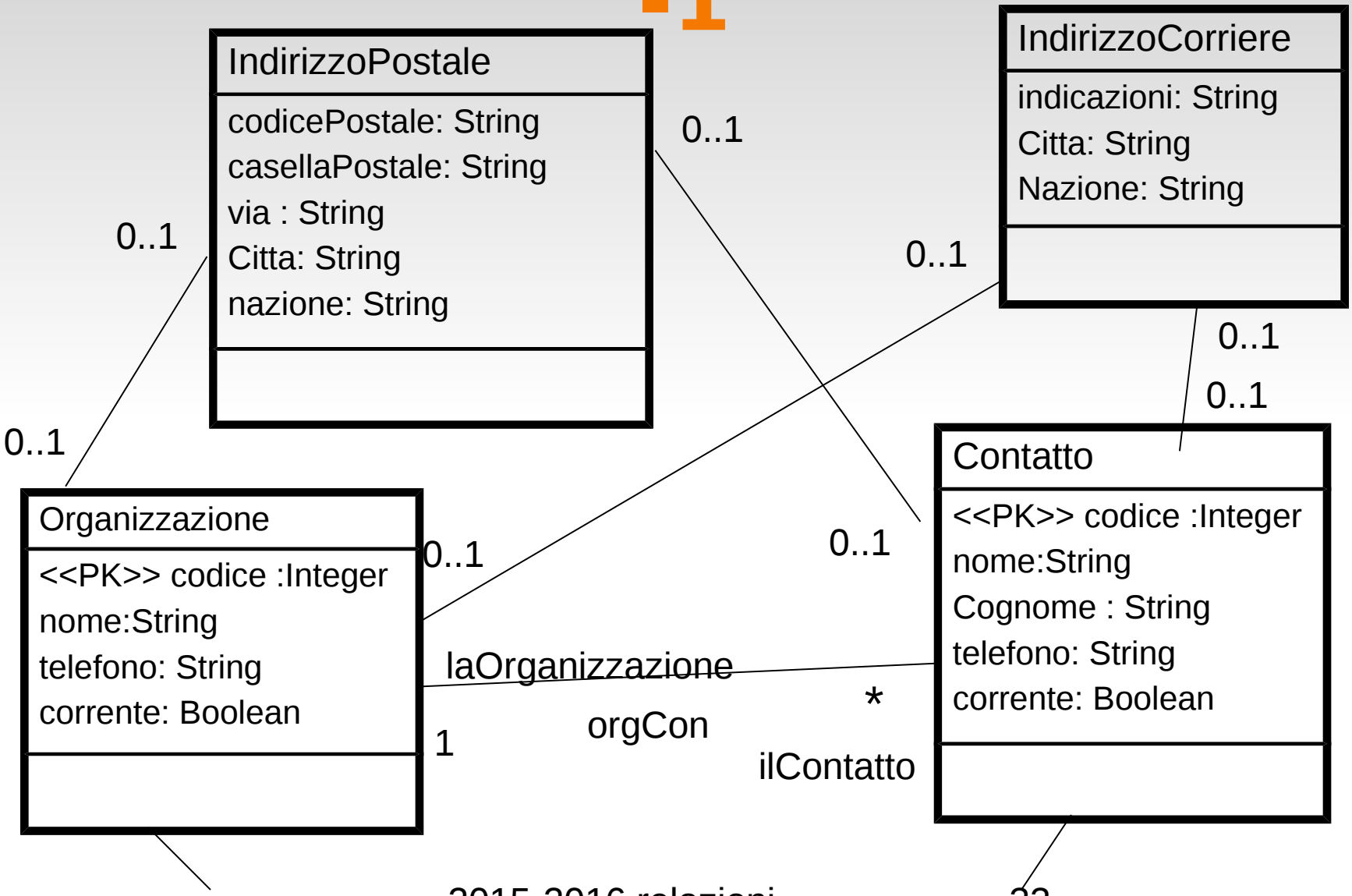
Esempio: gestione contatti

Requisito:

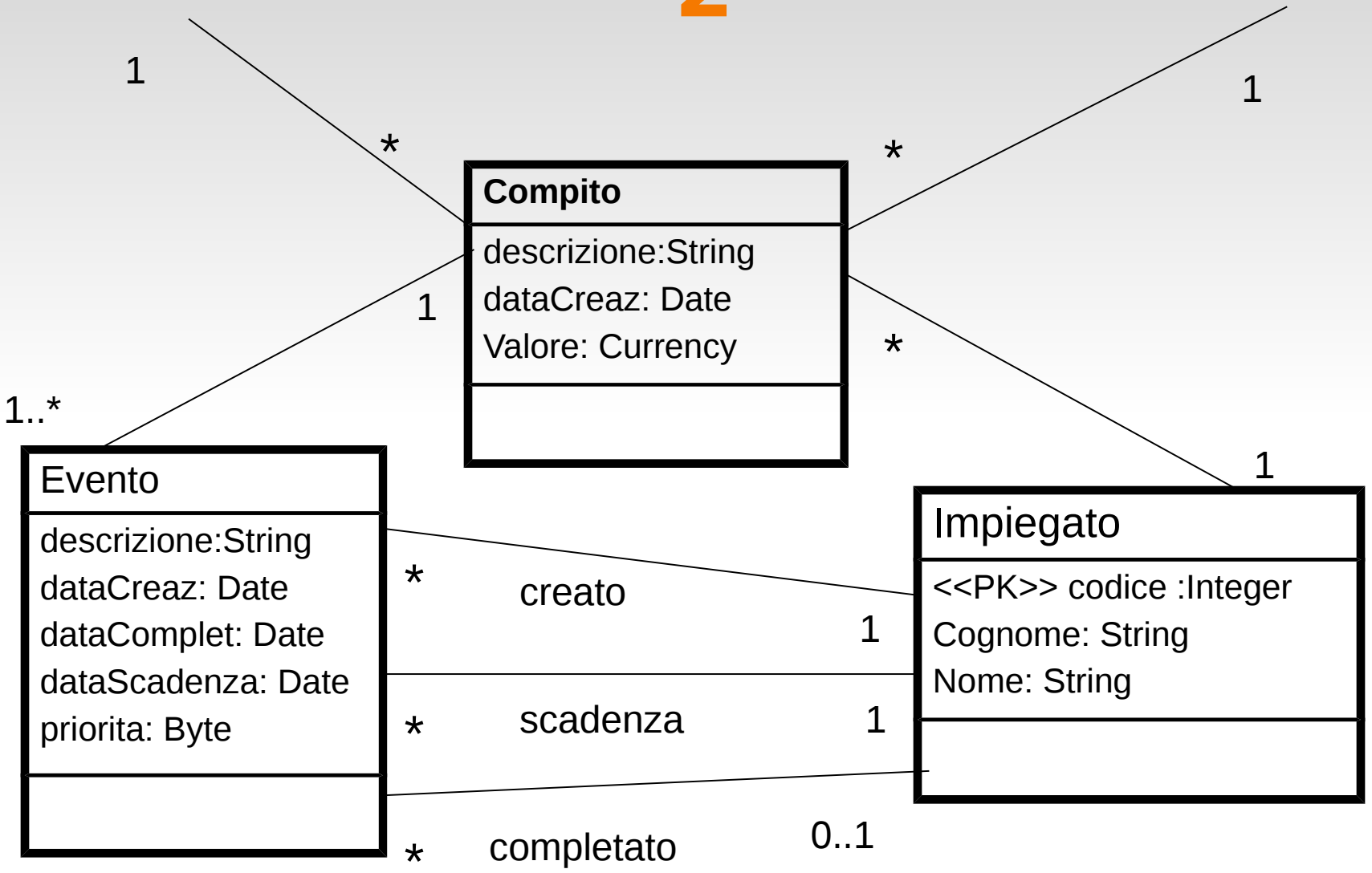
Il sistema permette la produzione di report sui contatti in relazione a indirizzo postale o di corriere



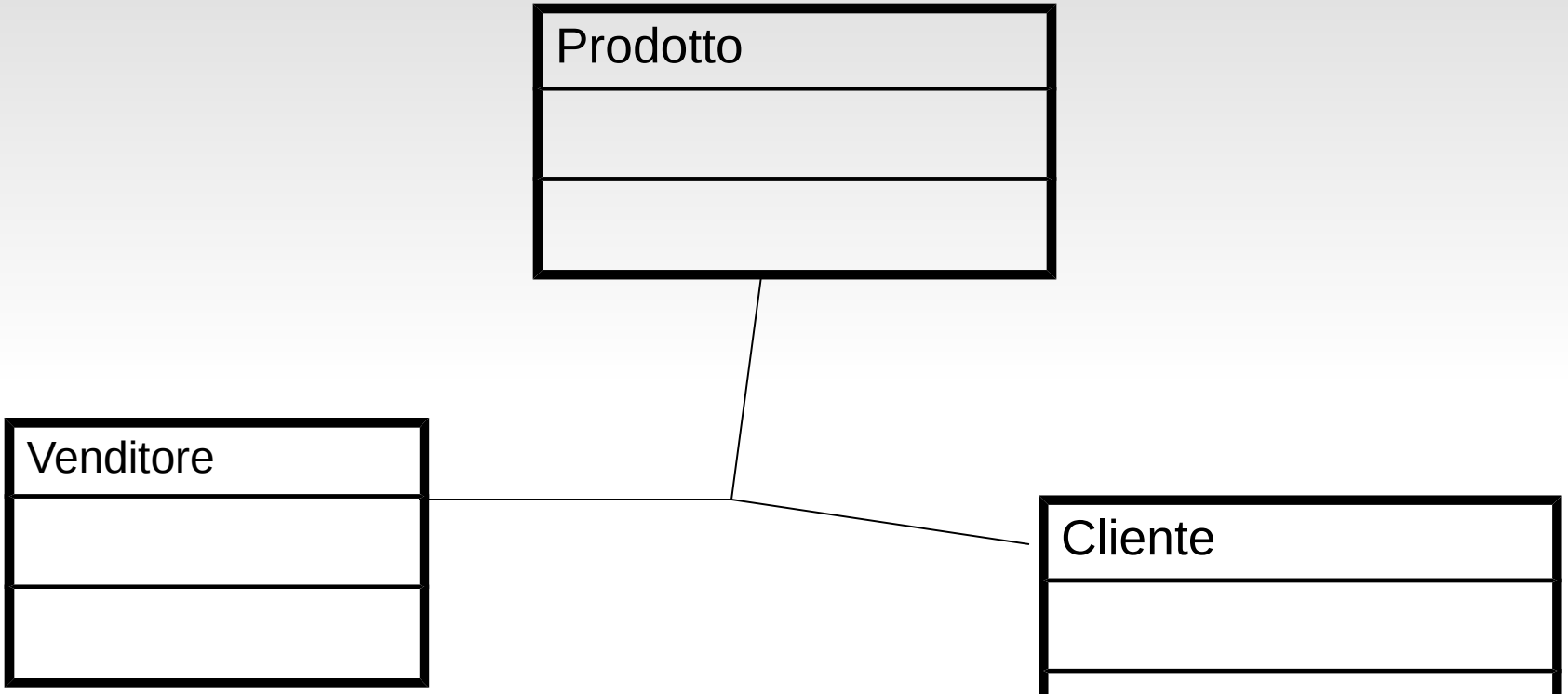
Esempio: gestione contatti -1



Esempio: gestione contatti -2

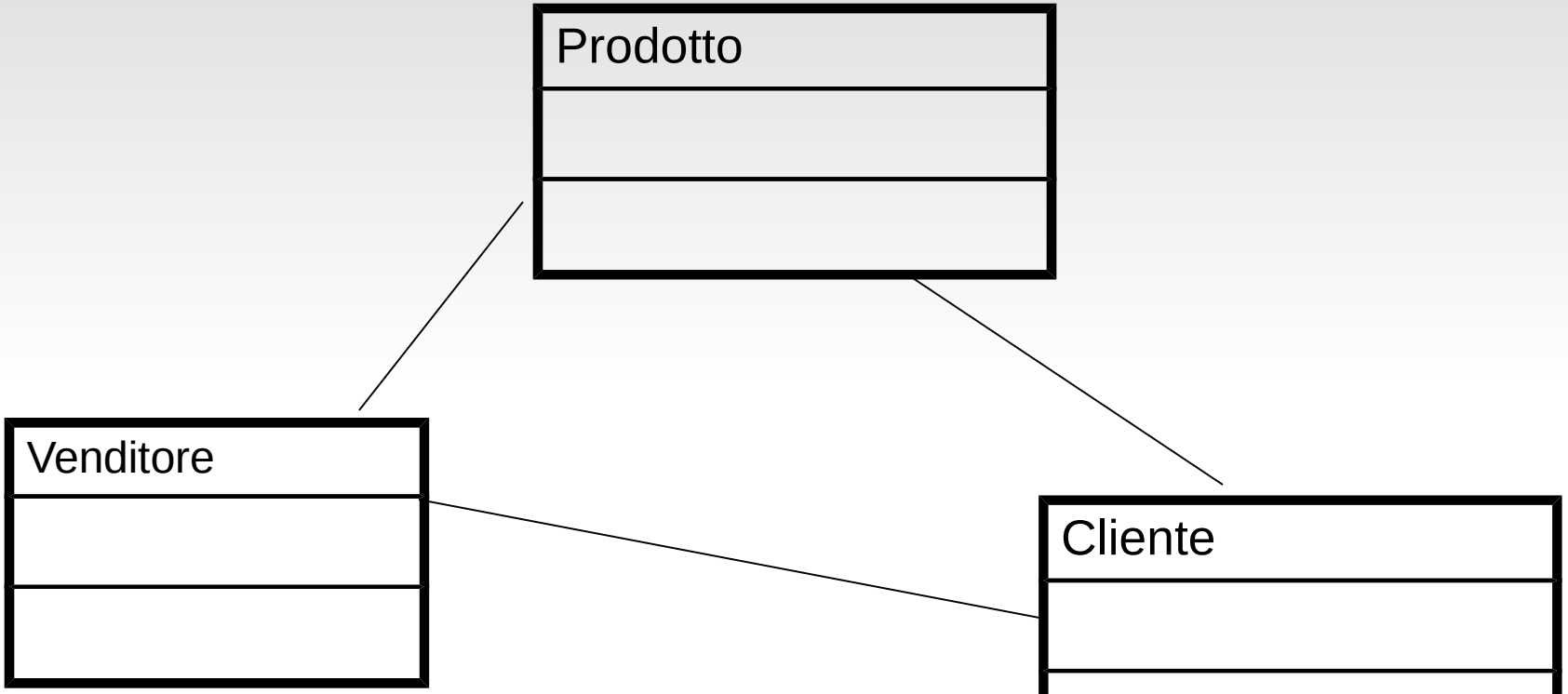


Trasformazione associazioni n-arie



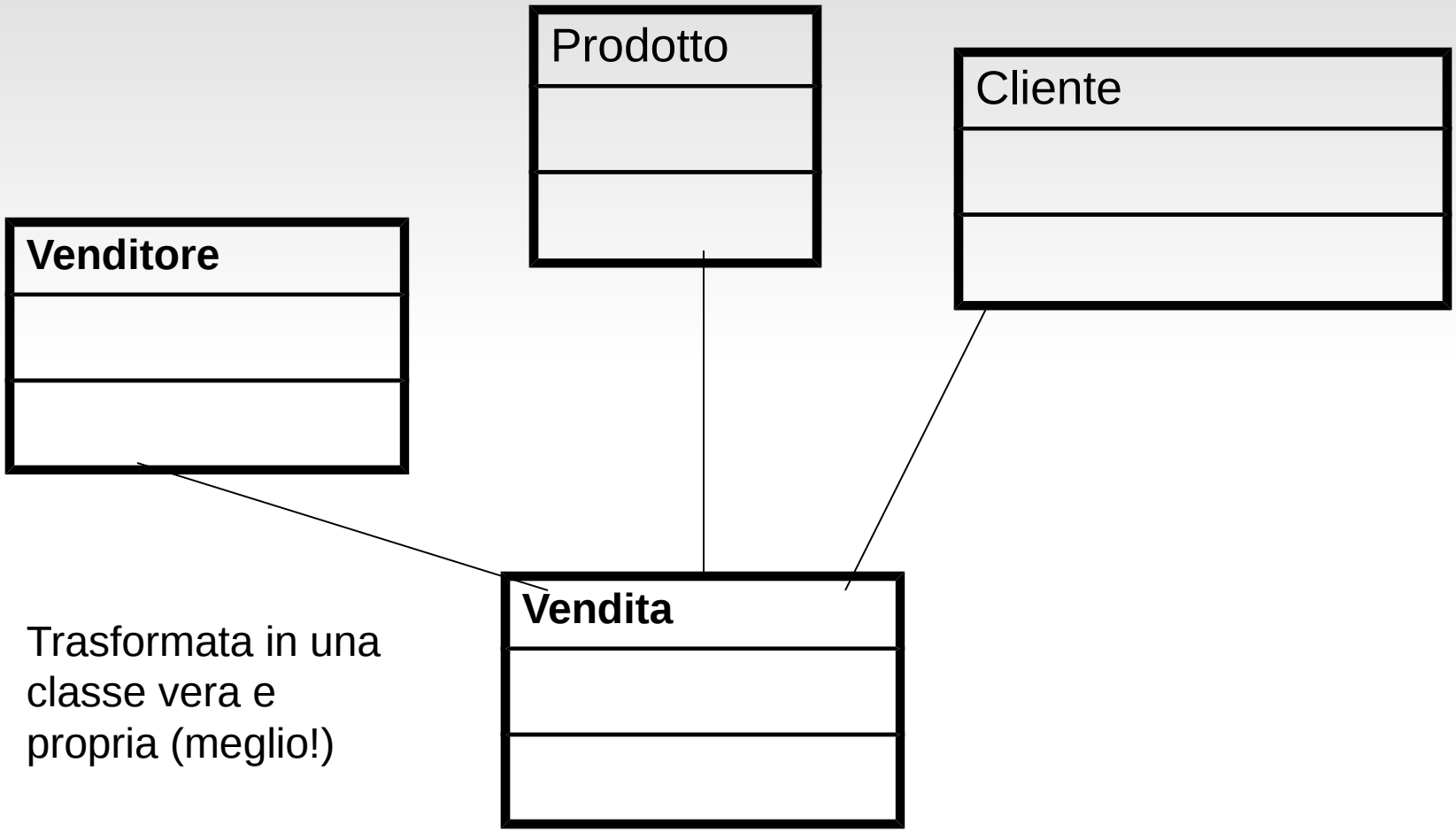
Associazione ternaria

Trasformazione associazioni n-arie



Trasformata in 3 associazioni binarie

Trasformazione associazioni n-arie



Trasformata in una classe vera e propria (meglio!)

trasformazioni di associazioni

a livello implementativo le cose sono molto semplici

nel caso precedente la classe Vendita avra' tre attributi per riferirsi alle altre classi

nelle relazioni unarie specificare se riflessive o no (es vicino vs genitore)

esempio

date le seguenti classi, derivare le associazioni

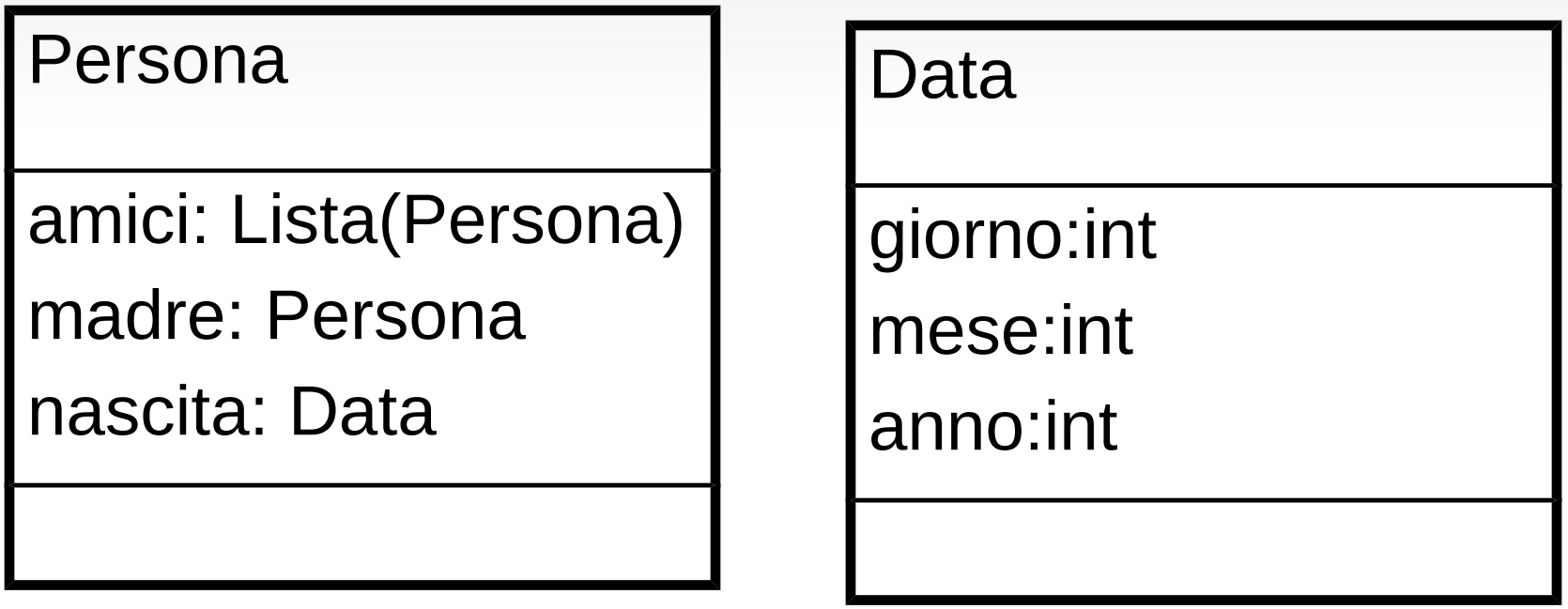
Venditore
prodotti: List(Prodotto) cliente:Cliente corriere: Corriere

Spedizione
prodotti: List(Prodotto) cliente:Cliente

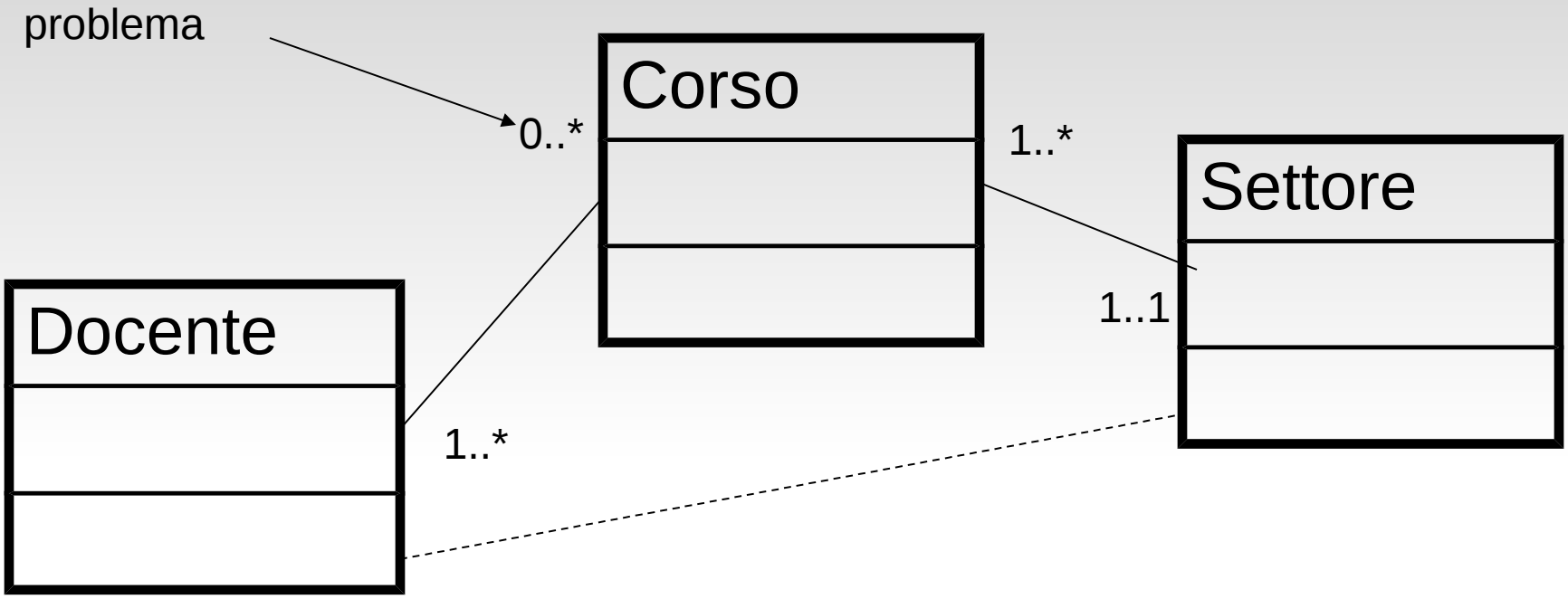
Corriere
spedizioni: List(Spedizione) cliente:Cliente

ridondanze?

trasformazioni di associazioni-backward engineering



navigazione di associazioni



Possiamo capire se un Docente appartiene ad un solo settore?

Tipi di dipendenze tra elementi

Uso

- il cliente usa qualche servizio del fornitore per implementare il suo comportamento (il piu' comune)

Astrazione

- Tante possibilita': Un modello e' prima nella fase di sviluppo, un modello nella fase di analisi e' piu' astratto che nella fase di implementazione

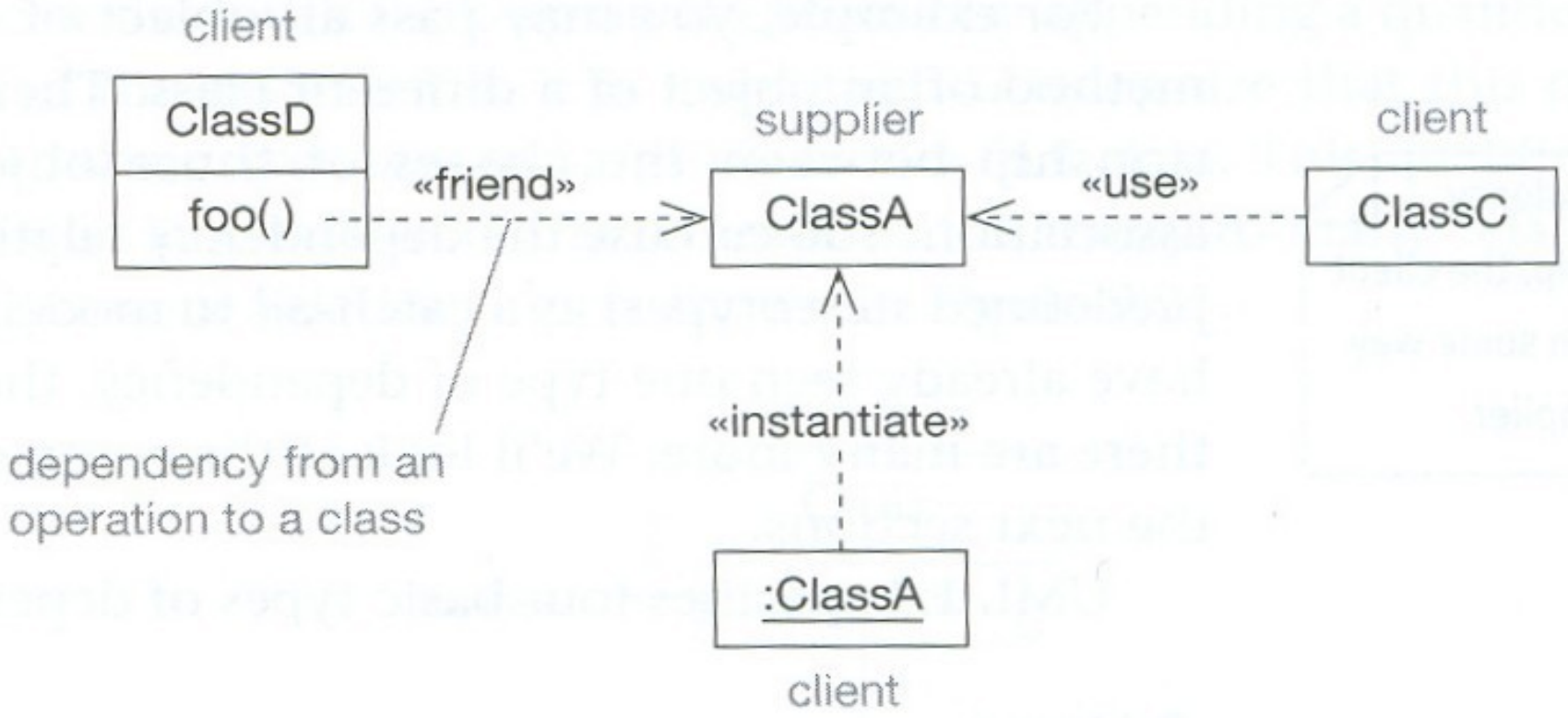
Permesso

- il fornitore concede al cliente il permesso di accedere a qualche sua risorsa

Binding

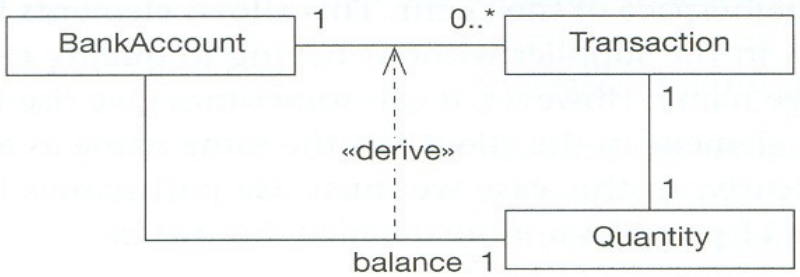
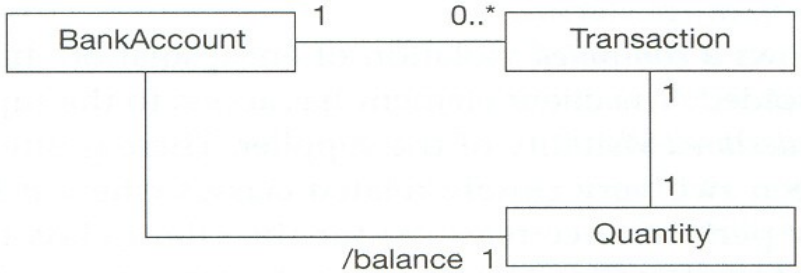
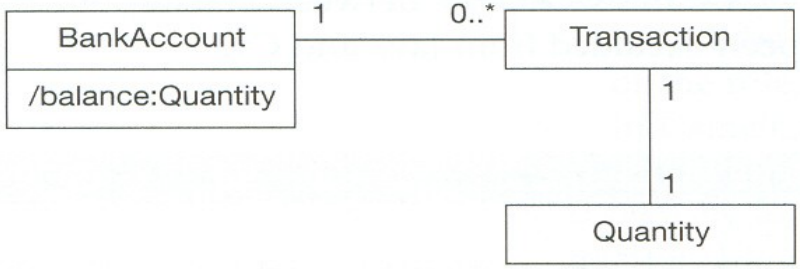
- usato per il C++, per l'istanziazione dei tipi (vedi piu' avanti)

Esempi di dipendenze



Dipendenza d'astrazione

Table 9.3

Model	Description
	The BankAccount class has a derived association to Quantity where Quantity plays the role of the balance of the BankAccount This model emphasizes that the balance is derived from the BankAccount's collection of Transactions
	In this case a slash is used on the role name to indicate that the relationship between BankAccount and Quantity is derived This is less explicit as it does not show what the balance is derived from
	Here the balance is shown as a derived attribute – this is indicated by the slash that prefixes the attribute name This is the most concise expression of the dependency

Dipendenza di permesso

indicano la capacita’ di un elemento di accedere ad un altro

esempi:

<<access>>

<<import>>

<<friend>>

Permesso di accesso

Perche' tanti dettagli sui permessi di accesso?

- Perche' si vuole che i vari oggetti del sistema funzionino in modo il piu' possibile indipendente, senza conoscere i dettagli di funzionamento interni degli altri oggetti
- Allora si modula l'accesso in tanti gradi diversi, cercando di ridurlo al minimo indispensabile

Modellare le aggregazioni

4 tipi di semantica possibili

- PossessoreEsclusivo (Albero ha Tronco)
  - esistenza condizionata
  - transitiva
  - asimmetrica
  - Indissolubile
  - Ingl: ExclusiveOwns
- Possessore (La bici ha due ruote)
  - Manca l' indissolubilita'
  - Ingl: Owns
- Aggregazione (Cesto ha uova)
  - Manca l' indissolubilita'
  - Manca esistenza condizionata
  - Ingl: Has
- Membro (Processo)
  - Nessuna proprieta' speciale: relazione paritaria tra molti
  - Ingl: Member

Scoprire le aggregazioni

Parallelamente alle associazioni

test di verifica:

- La bici ha le ruote
- Le ruote sono parte della bici

Spesso coinvolge piu’ classi (manubrio freni luci ..)

Definire le aggregazioni

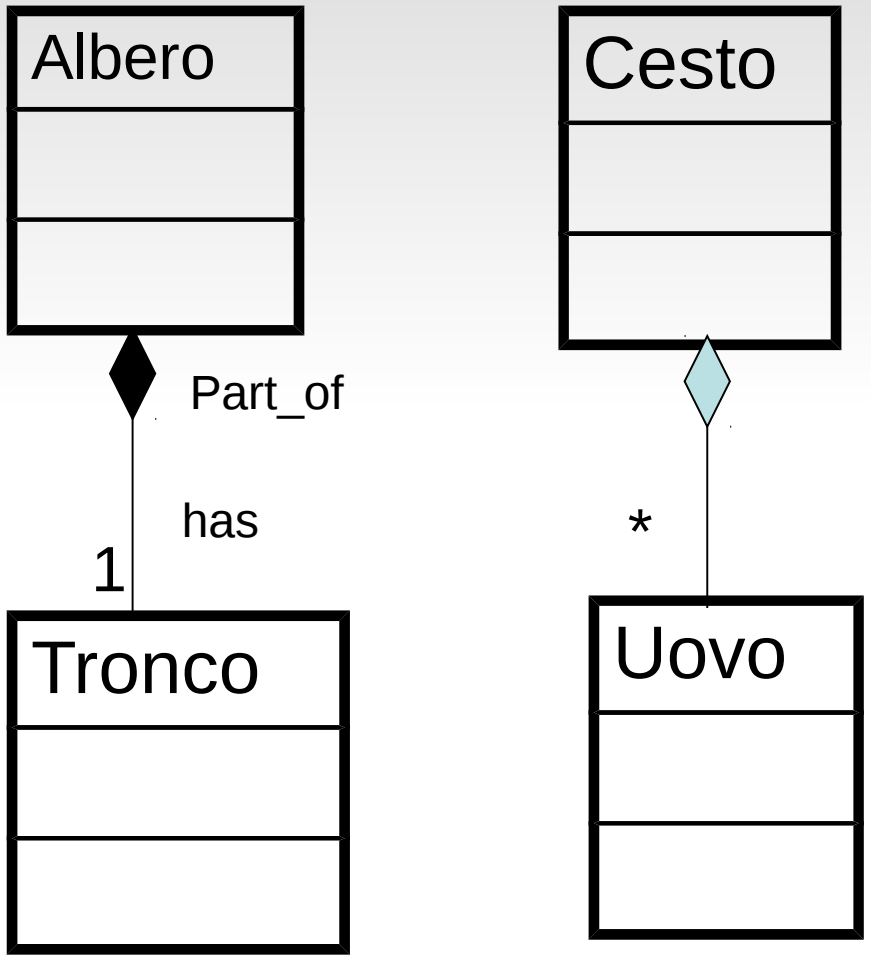
In UML

- Aggregazioni:
  - Semantica a referenza
  - Rombo vuoto
  - Corrisponde a Has e Member
- Composizioni
  - Semantica a valore
  - Rombo pieno
  - Corrisponde a Owns ed ExclusiveOwns

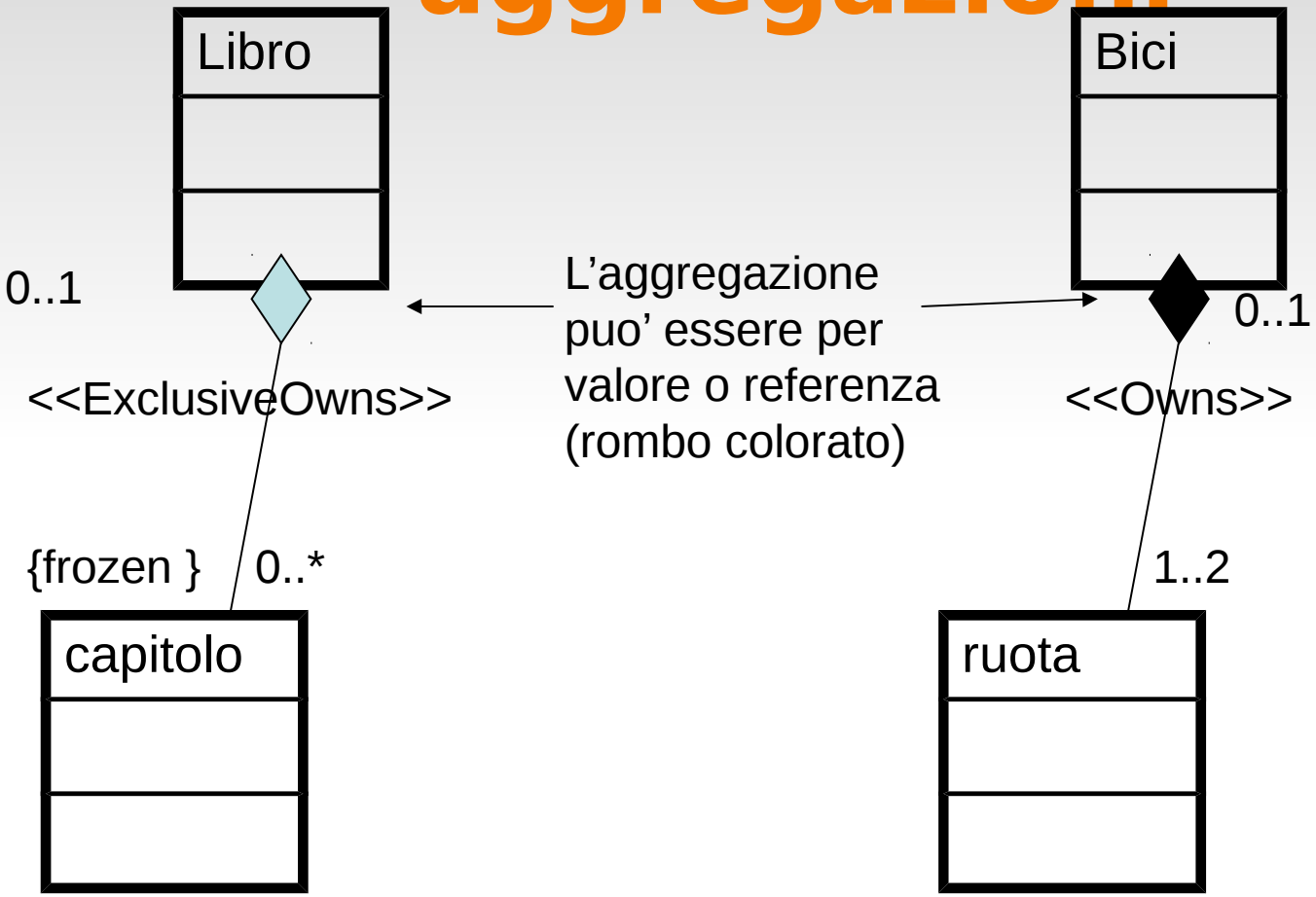
Relazioni di composizione ed aggregazione

Composizione: aggregazione per valore (forte)

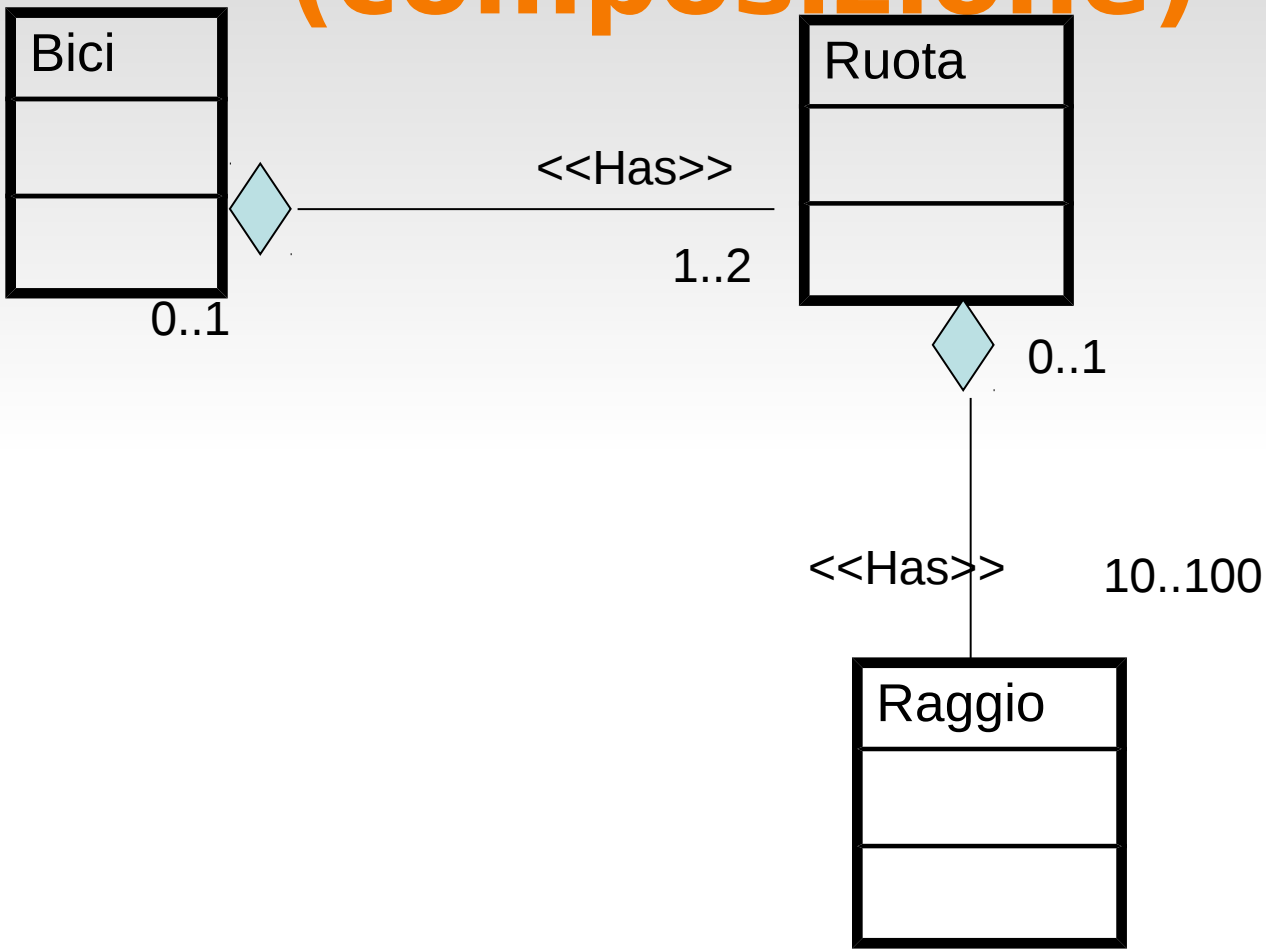
Aggregazione : per referenza (debole)



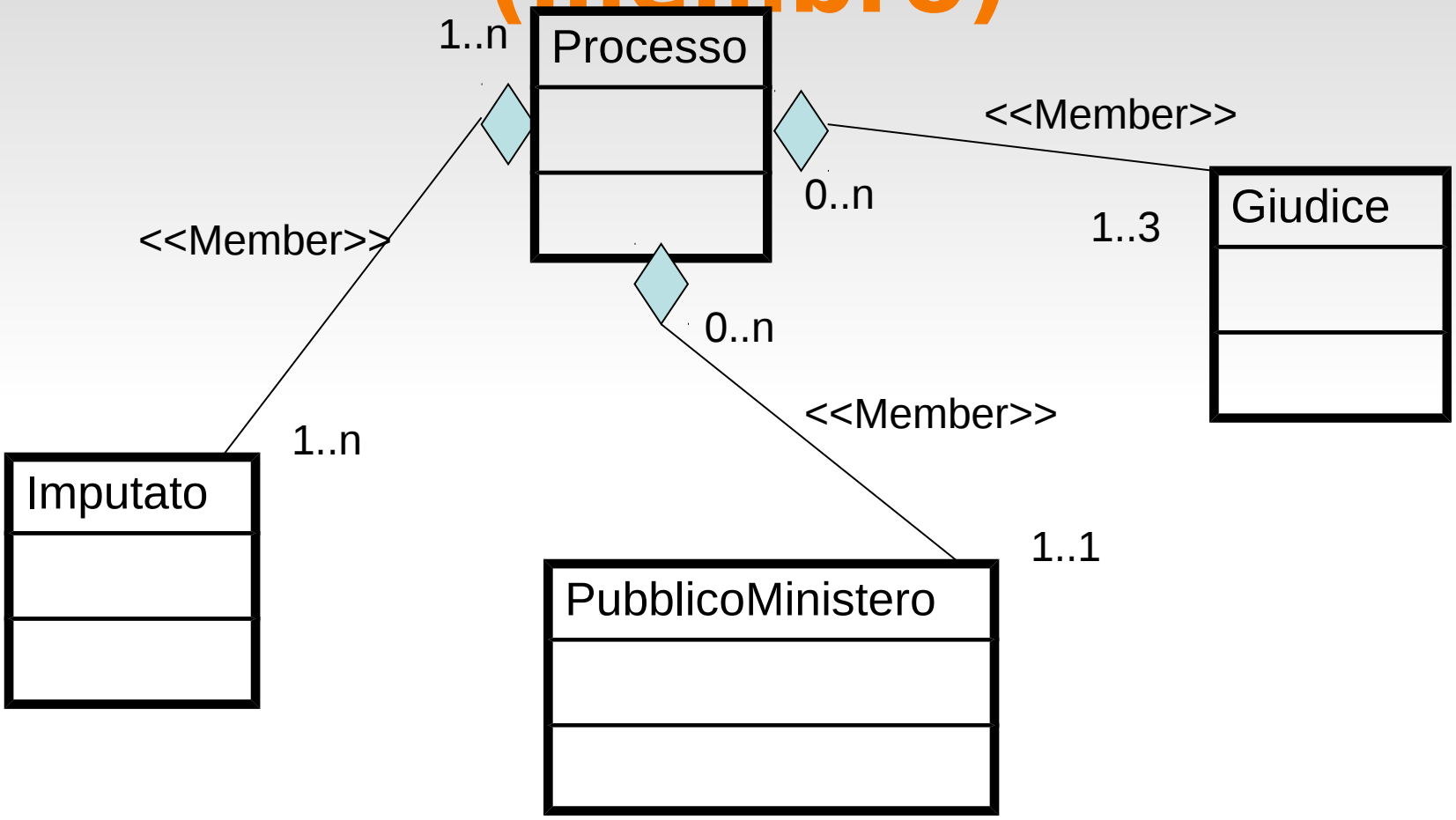
Notazione uml per le aggregazioni



Aggregazione “has” (composizione)



Aggregazione “member” (membro)



Proprieta' di aggregazione e composizione

Proprieta' di entrambe:

- Transitivita' (se A contiene B , B contiene C **ALLORA** A contiene C)
- Asimmetria (se A contiene B **ALLORA** B non contiene A)

Solo la composizione

- Esistenza condizionata (se A contiene B e distruggo A allora anche B viene distrutto)

In UML, si puo' usare la notazione di aggregazione se non si e' ancora deciso fra le due

aggregazioni o associazioni ?

Pane per gli studiosi di semantica

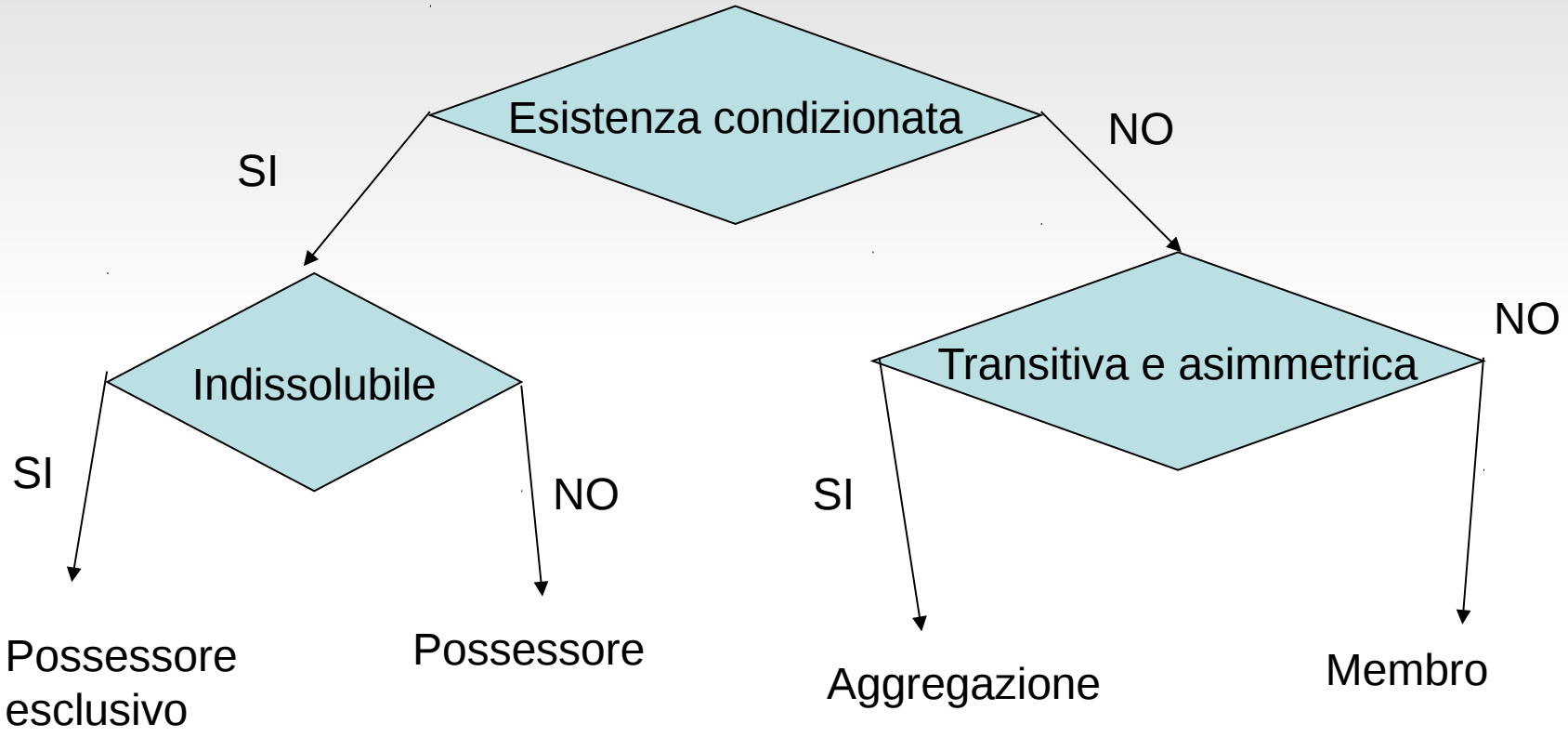
Spesso con le aggregazioni implichiamo una relazione asimmetrica di

- 1 contenitore → tanti contenuti

A livello implementativo una aggregazione e' rappresentata identicamente ad una associazione, link diretto o lista a seconda che 1-1 o 1 -*

Se abbiamo a disposizione solo l'implementazione, non possiamo distinguerle!

Aggregazioni



esercizio

fare due esempi di associazione, composizione, aggregazione

Esempio: iscrizione universitaria

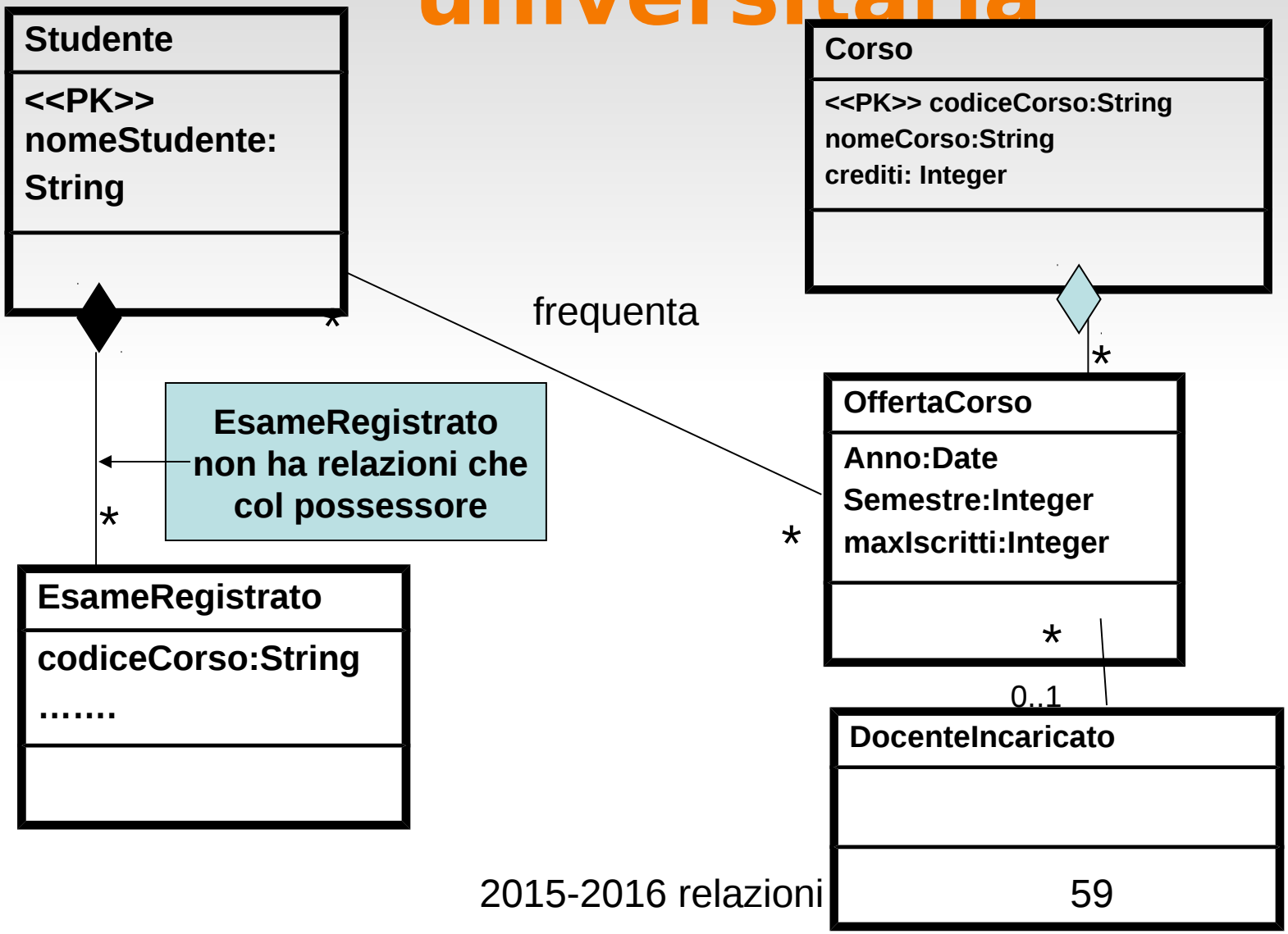
Il curriculum deve essere disponibile a richiesta

Il curriculum deve contenere esami, voti

Ogni corso ha un docente incaricato, ma altri possono collaborarvi

- L'incaricato puo' variare ogni semestre
- I docenti possono variare ogni semestre

Esempio: iscrizione universitaria



Commento agli esempi precedenti

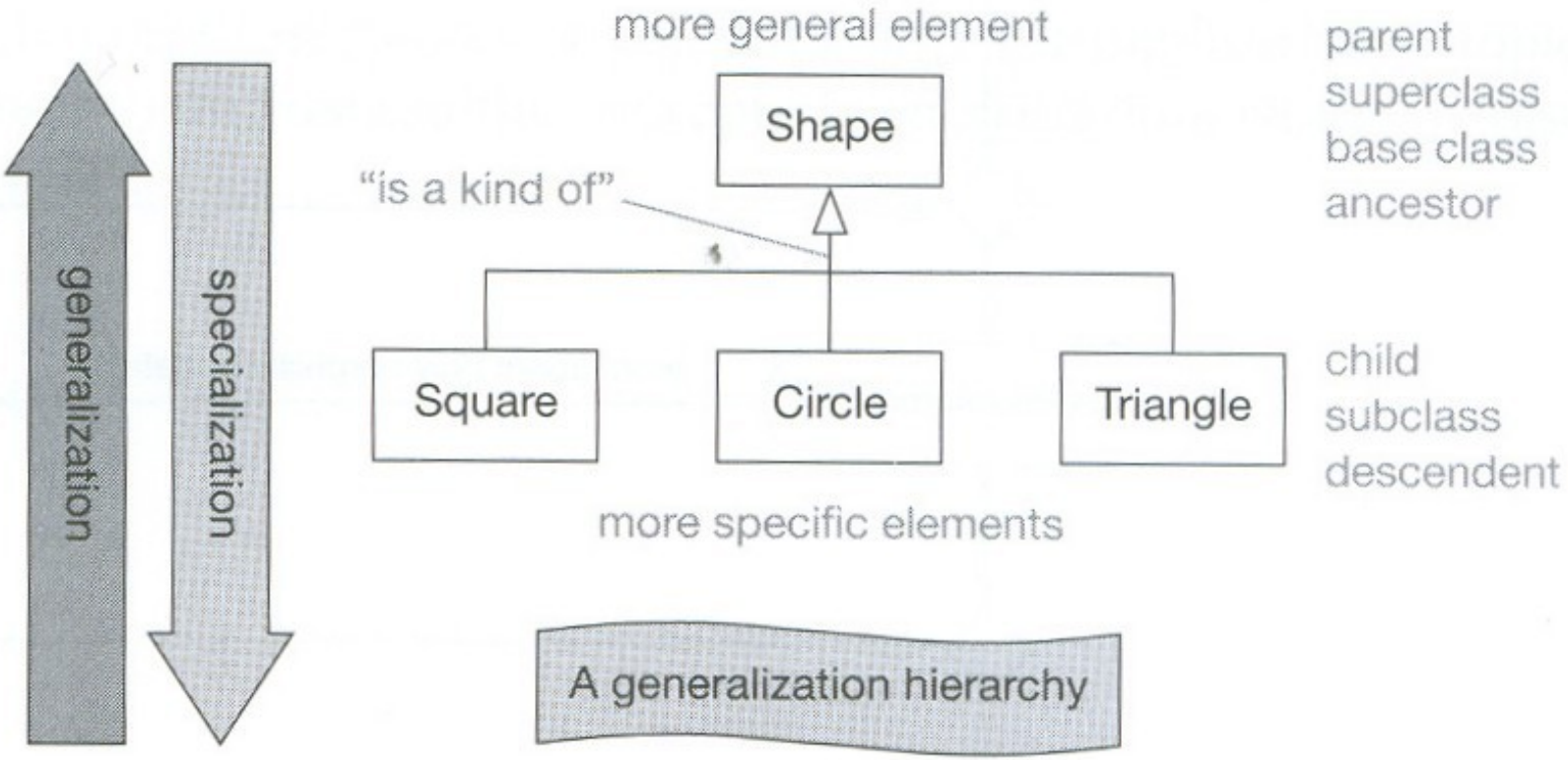
Conviene sempre pensare ad una soluzione il piu' generale possibile: e' piu' facile aggiungere vincoli ad una soluzione generale per limitarla, che non estendere una soluzione specifica causa rimozione vincoli

Usare la semantica per valore (Owns) puo' precludere svliluppi successivi ad un sistema:

- Se un libro ha dei capitoli, essi potrebbero un domani far parte di un nuovo libro organizzati in modo diverso (es. albero tronco segheria)
- se cancello lo studente, potrei voler conservare gli esami a fini statistici.
- A livello implementativo, la garbage collection semplifica molto la gestione e sfuma le differenze
- Owns puo' andare benissimo per oggetti poco costosi (es. Poligono -> Punti)

La generalizzazione (Arlow..)

Concetto applicabile a molti elementi



Modellare le generalizzazioni

- Le caratteristiche comuni sono fattorizzate in una classe piu' generale
- Le sottoclassi ereditano le caratteristiche della superclasse
- Una istanza di una sottoclasse puo' essere usata al posto della superclasse (arancia al posto di frutto)
- **Polimorfismo**: la stessa operazione puo' essere implementata in modo differente in classi diverse
- **Operazione astratta**: implementata nella sottoclasse
- **Classe astratta**: classe che non puo' essere istanziata Le classi figlie partizionano le istanze

Scoperta e specifica delle generalizzazioni

Scoperta assieme alle associazioni

Frase di test:

- Puo' essere
- E' del tipo di

E' possibile l'ereditarieta' multipla

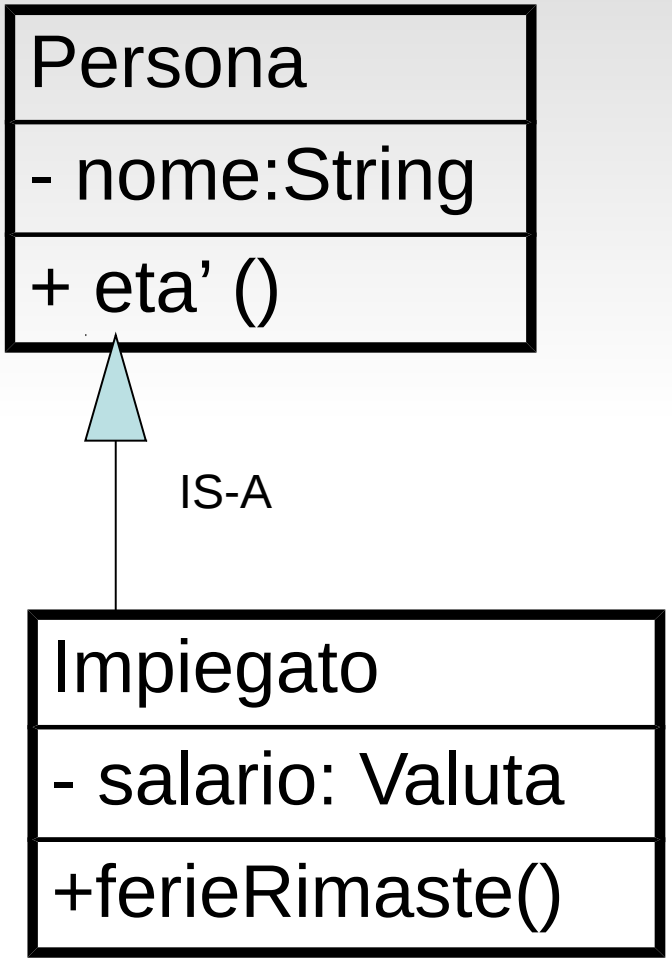
Linea con freccia che punta alla superclasse

Generalizzazione

Ereditarieta’ di dati ed attributi

Relazione IS-A

Riuso



Overriding (prevalere)

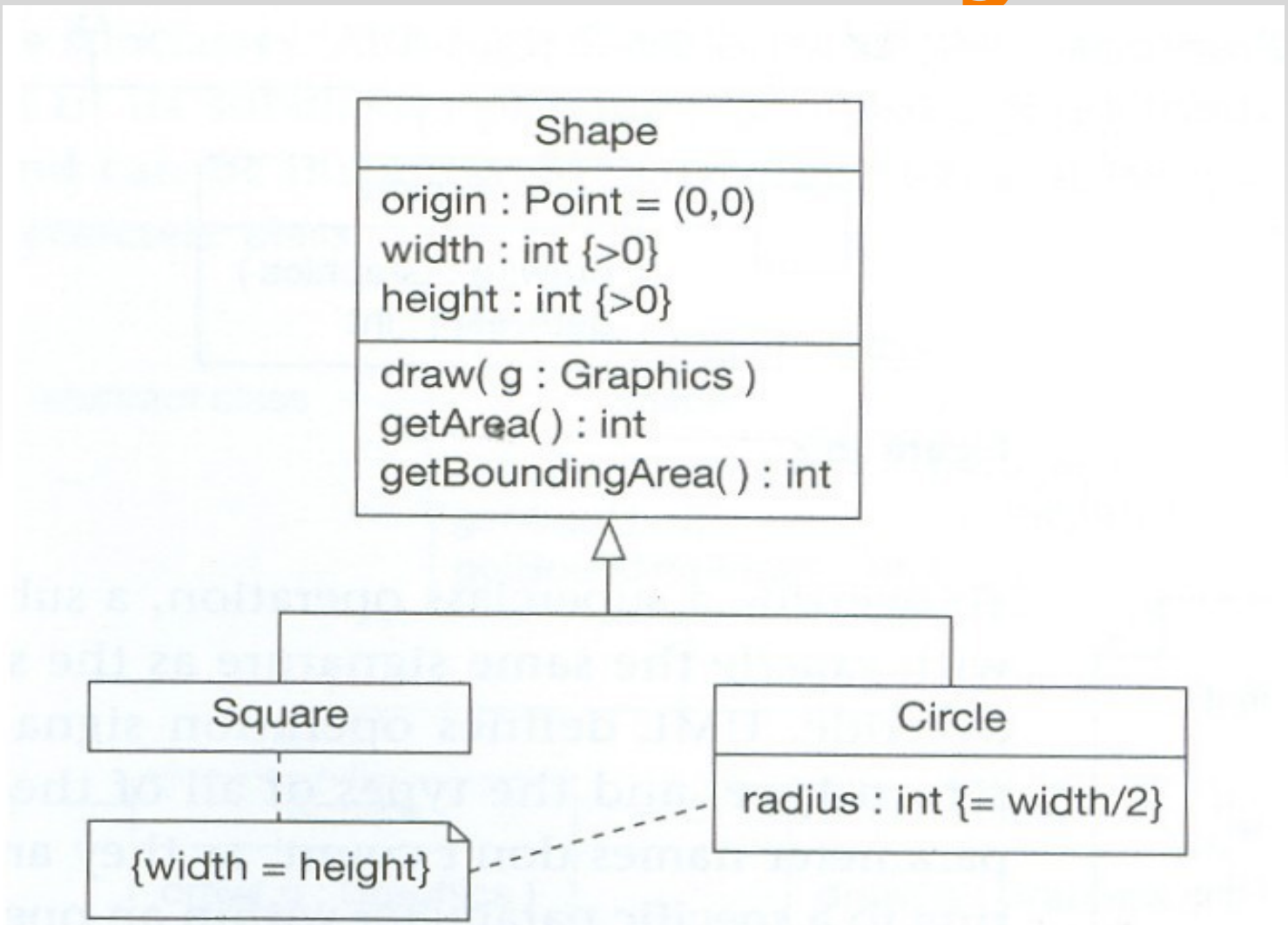
Quando una operazione e’ definita in una classe antenata, essa puo’ essere ridefinita nella discendente

deve avere la **stessa firma**

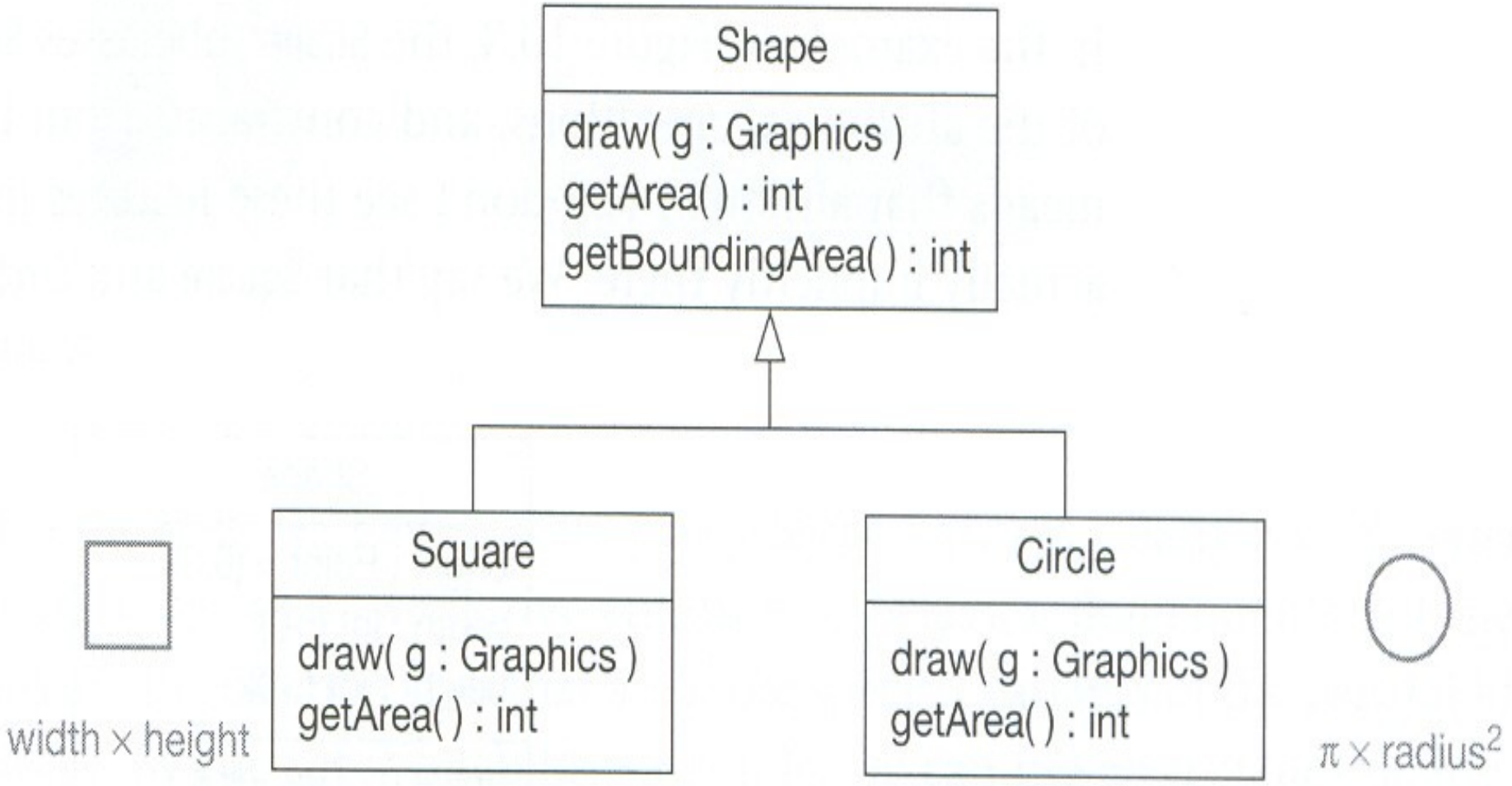
e’ una specializzazione con la **stessa semantica** dell’antenato;

Perfeziona o completa l'operazione dell'antenato

Overriding


Shape
origin : Point = (0,0)
width : int {>0}
height : int {>0}
draw(g : Graphics)
getArea( ) : int
getBoundingArea( ) : int
Square
{width = height}
Circle
radius : int {= width/2}

Overriding


Shape
draw(g : Graphics)
getArea( ) : int
getBoundingArea( ) : int
Square
draw(g : Graphics)
getArea( ) : int
Circle
draw(g : Graphics)
getArea( ) : int
width × height
π × radius²

Classi ed operazioni astratte

Quando una classe e’ semanticamente incompleta per attributi od operazioni

non ha senso crearne una istanza a se’ stante

per usarne le proprieta’ occorre derivare classi discendenti es.

- veicolo->automobile,bici…
- persona->uomo,donna

Da dove sbucano?

Astraendo da classi concrete!

Definisco uomo, donna,

scopro che hanno proprieta' in comune,

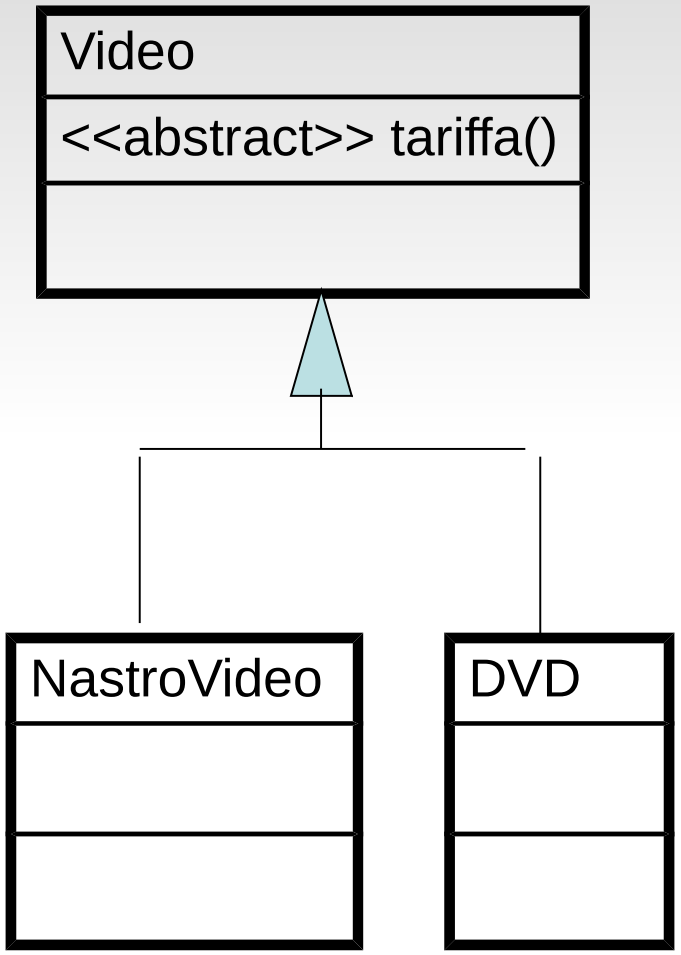
le raggruppo in una classe antenata, che e' un contenitore “di comodo”, che serve ad organizzare I dati, ma di cui non creero' mai istanze ad esempio se facessi un sistema per l'Anagrafe del comune

Classe astratta

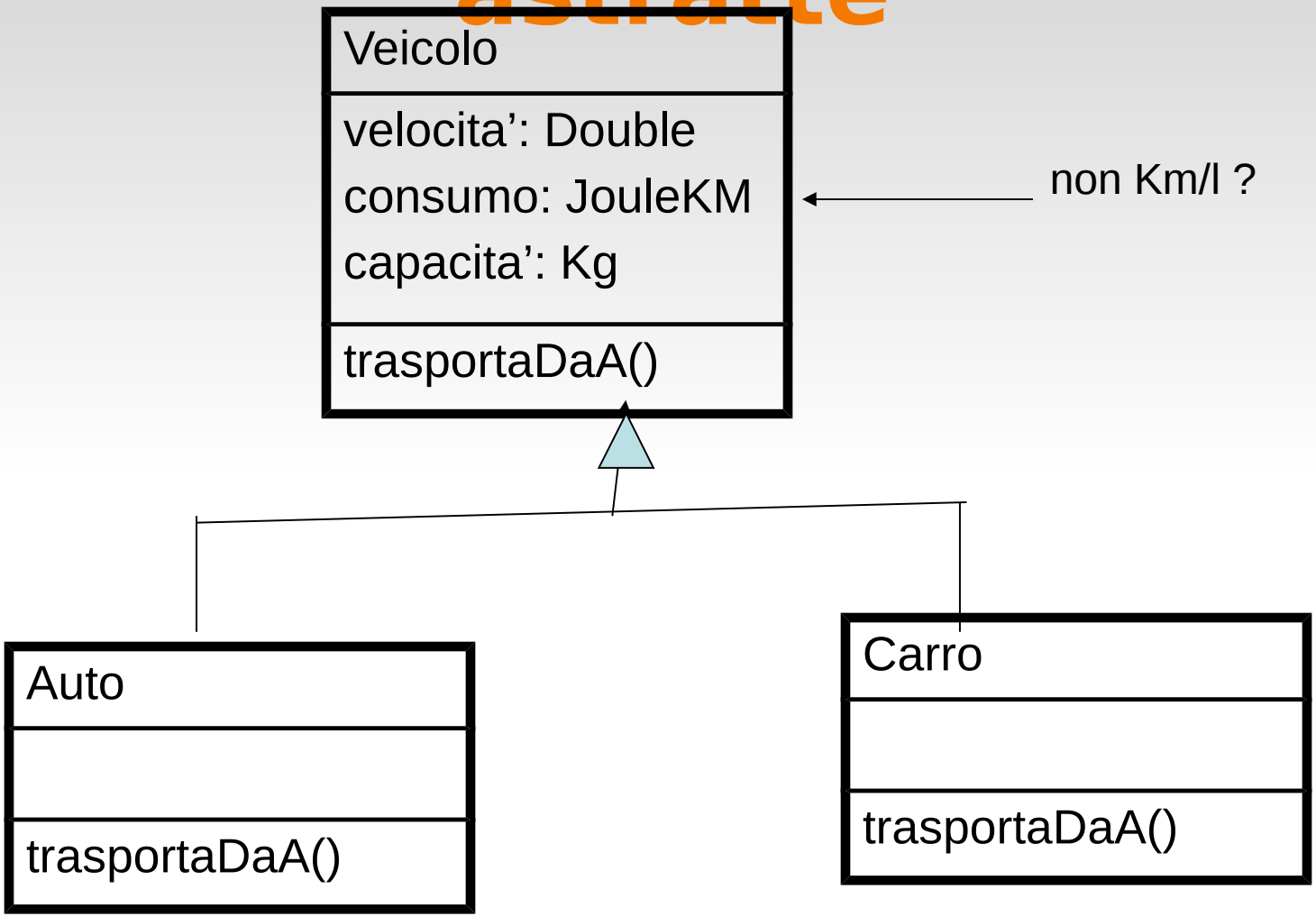
Una classe genitore che non puo' avere istanze

La classe astratta non si puo' istanziare perche' ha almeno una operazione astratta

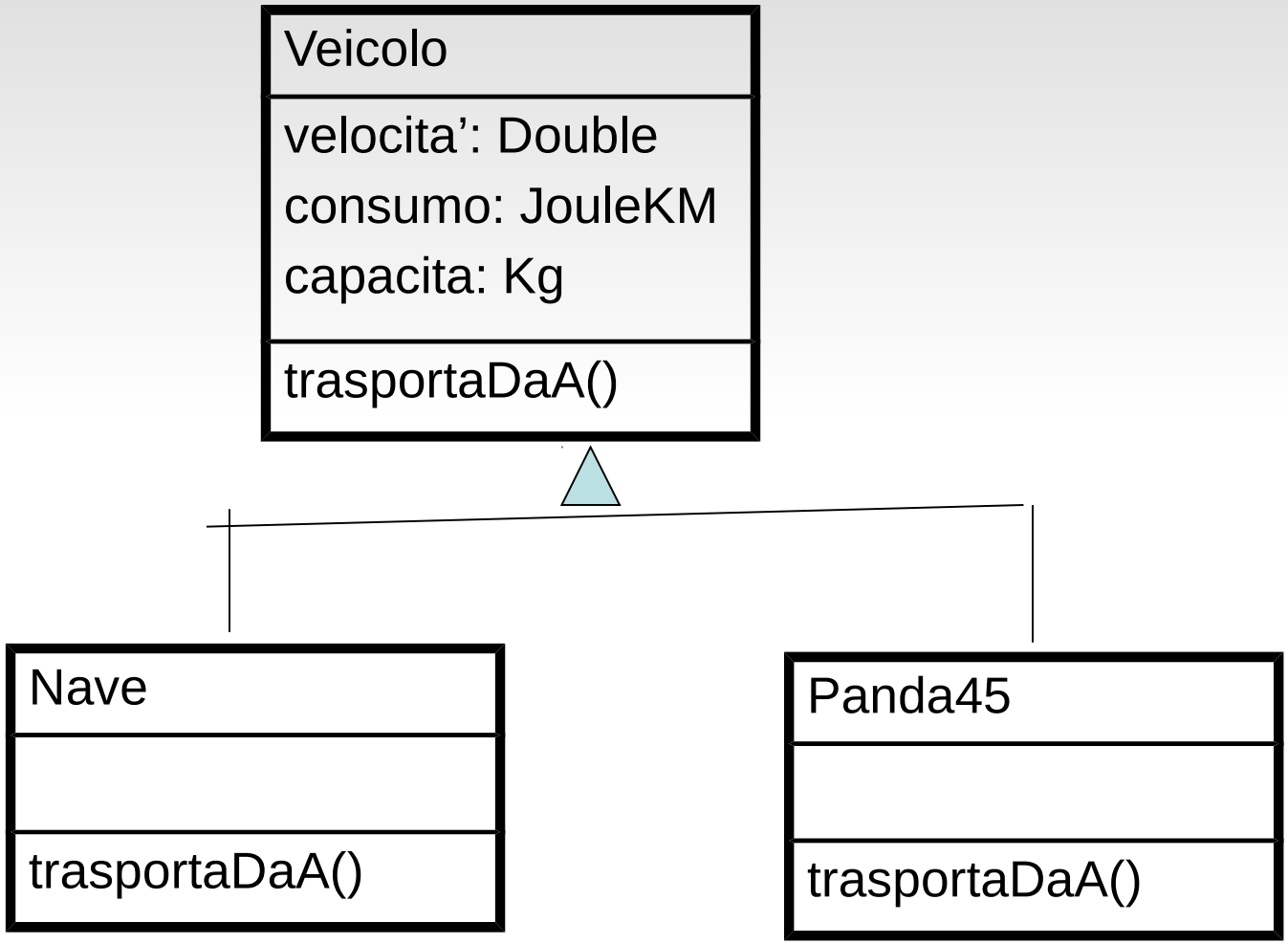
Possibile se le sottoclassi partizionano completamente le istanze (EssereUmano,uomo,donna)



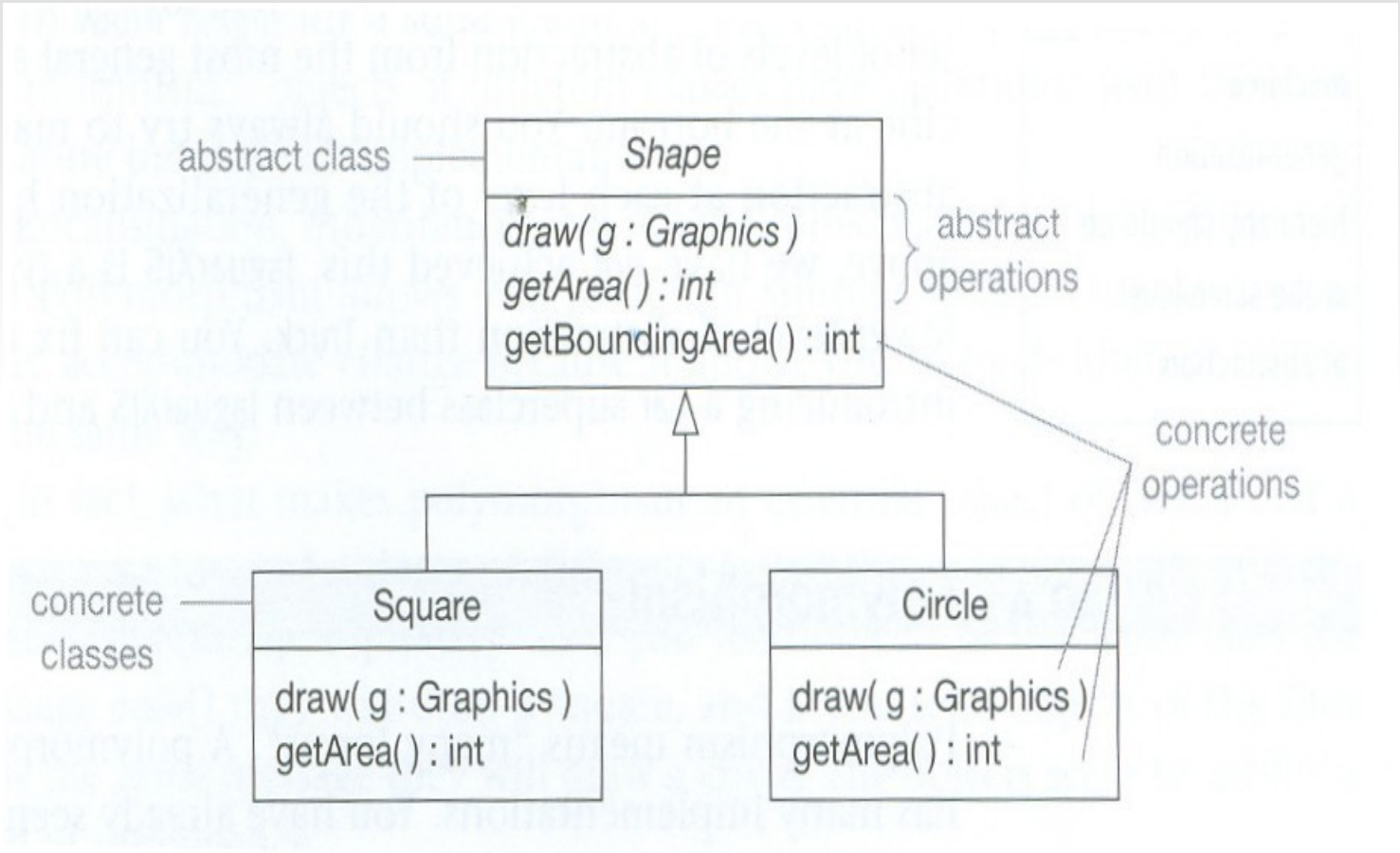
Classi ed operazioni astratte



Generalizzazione disomogenea!

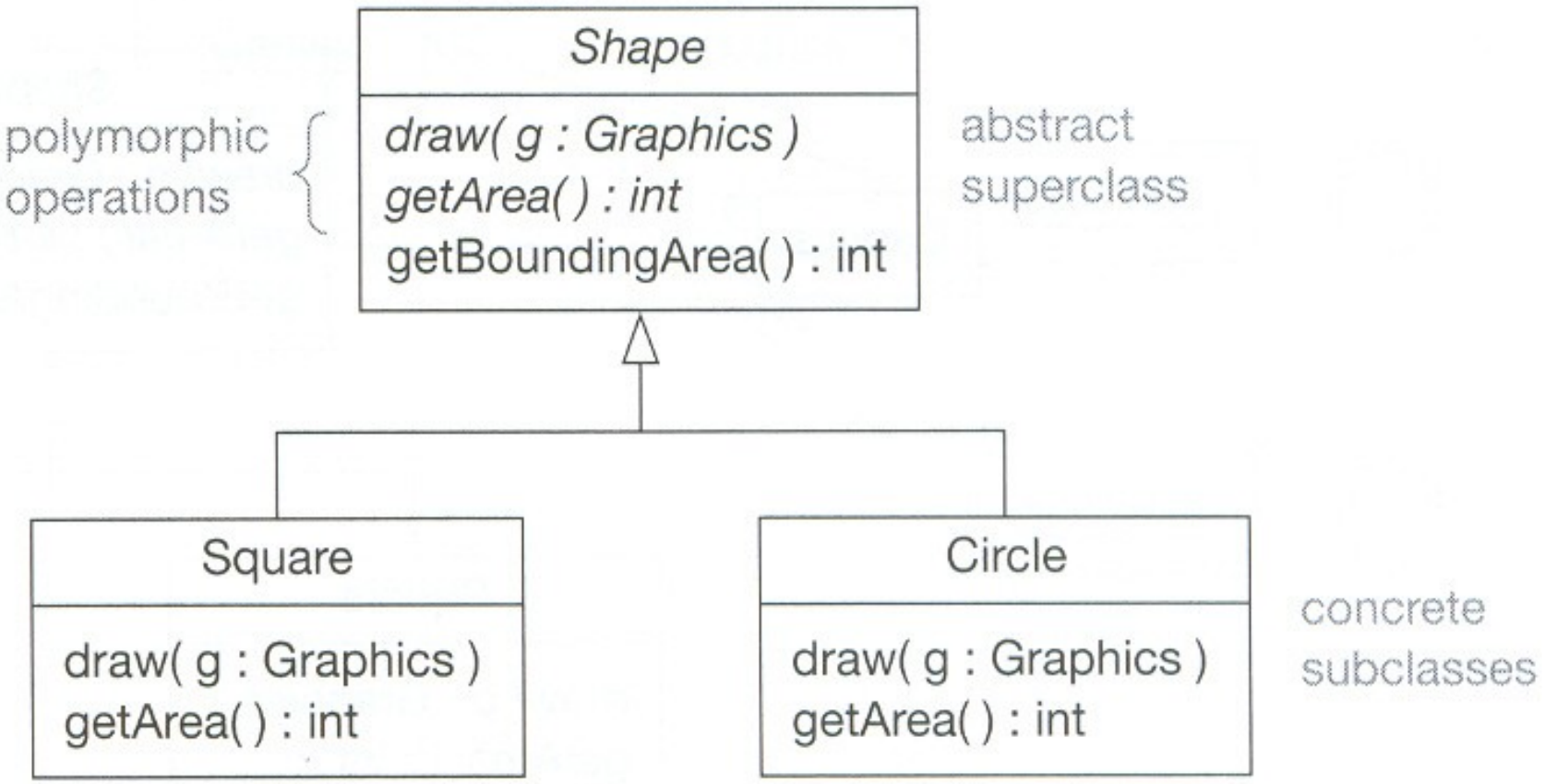


Classi ed operazioni astratte (Arlow..)

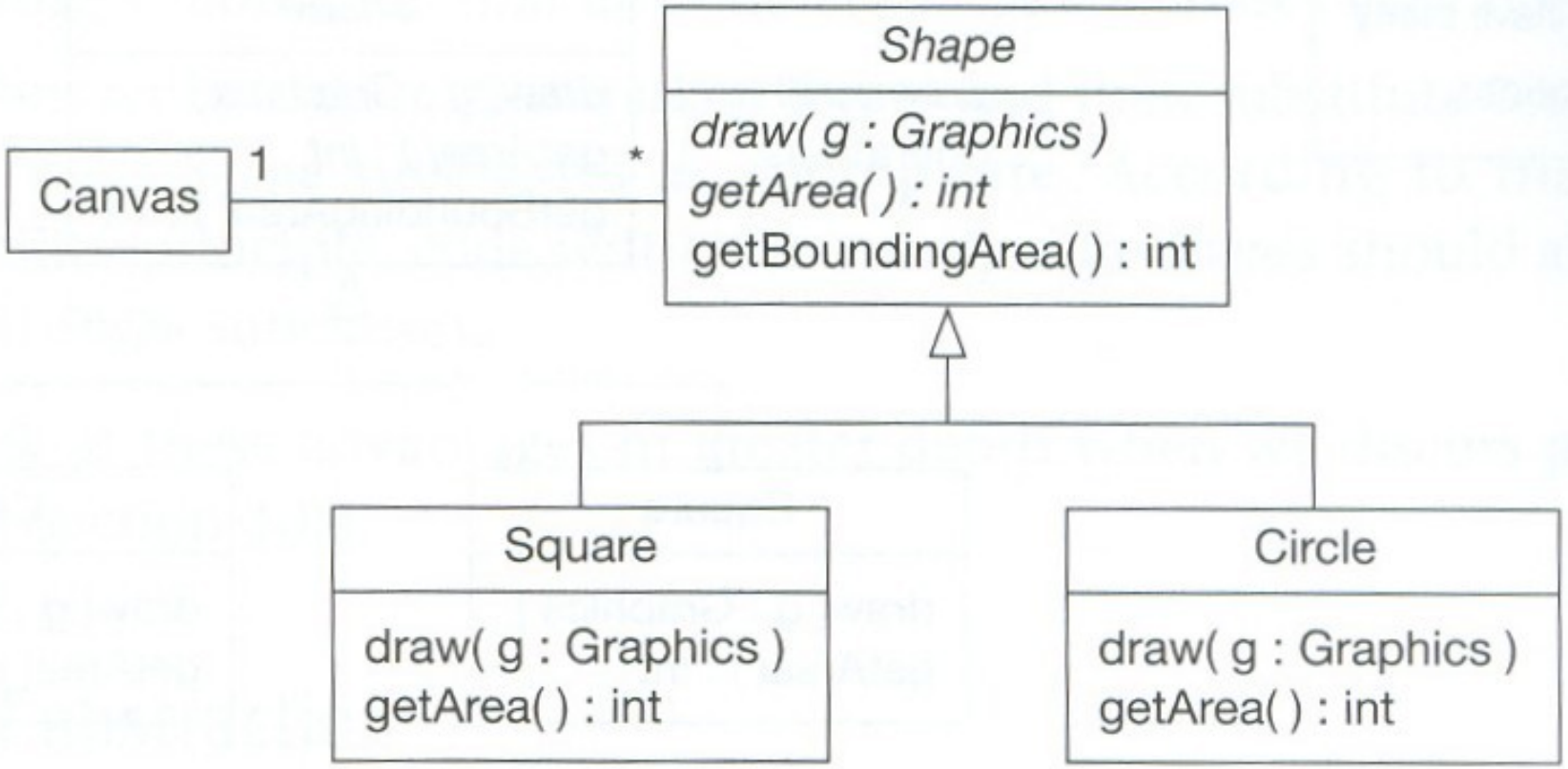


Polimorfismo (Arlow..)

quando una operazione e' realizzata in molti modi

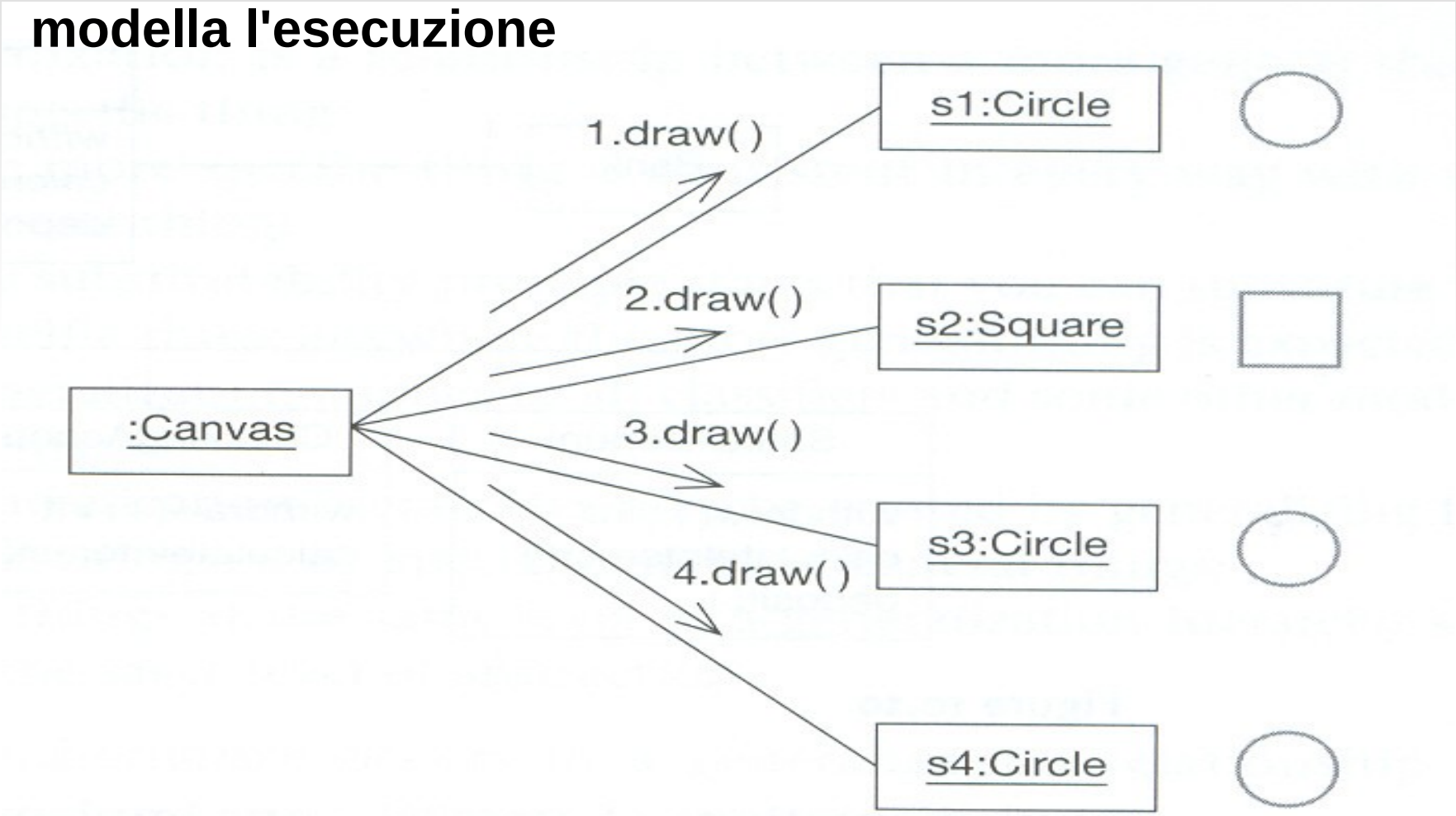


Polimorfismo (Arlow..)

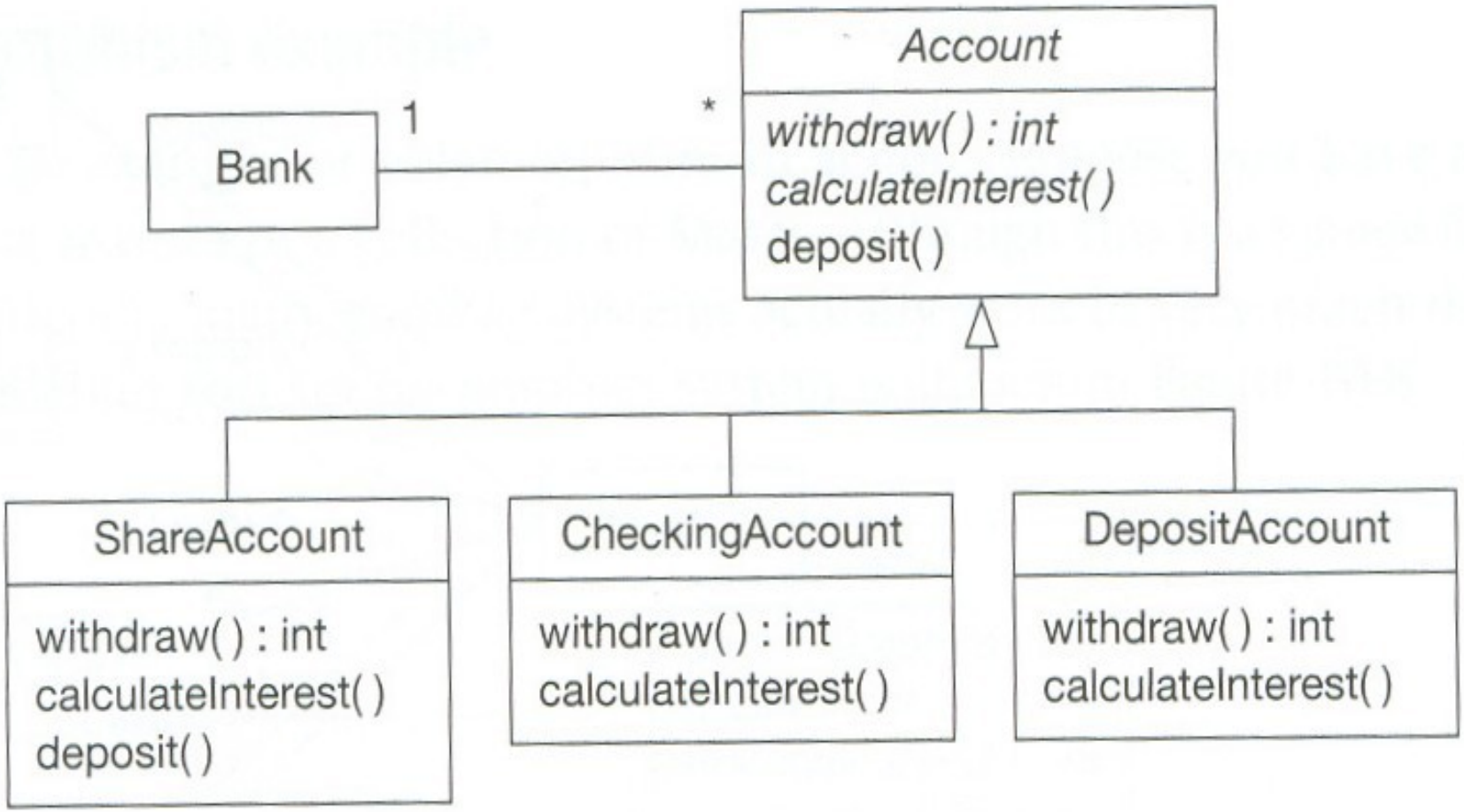


Polimorfismo (Arlow..)

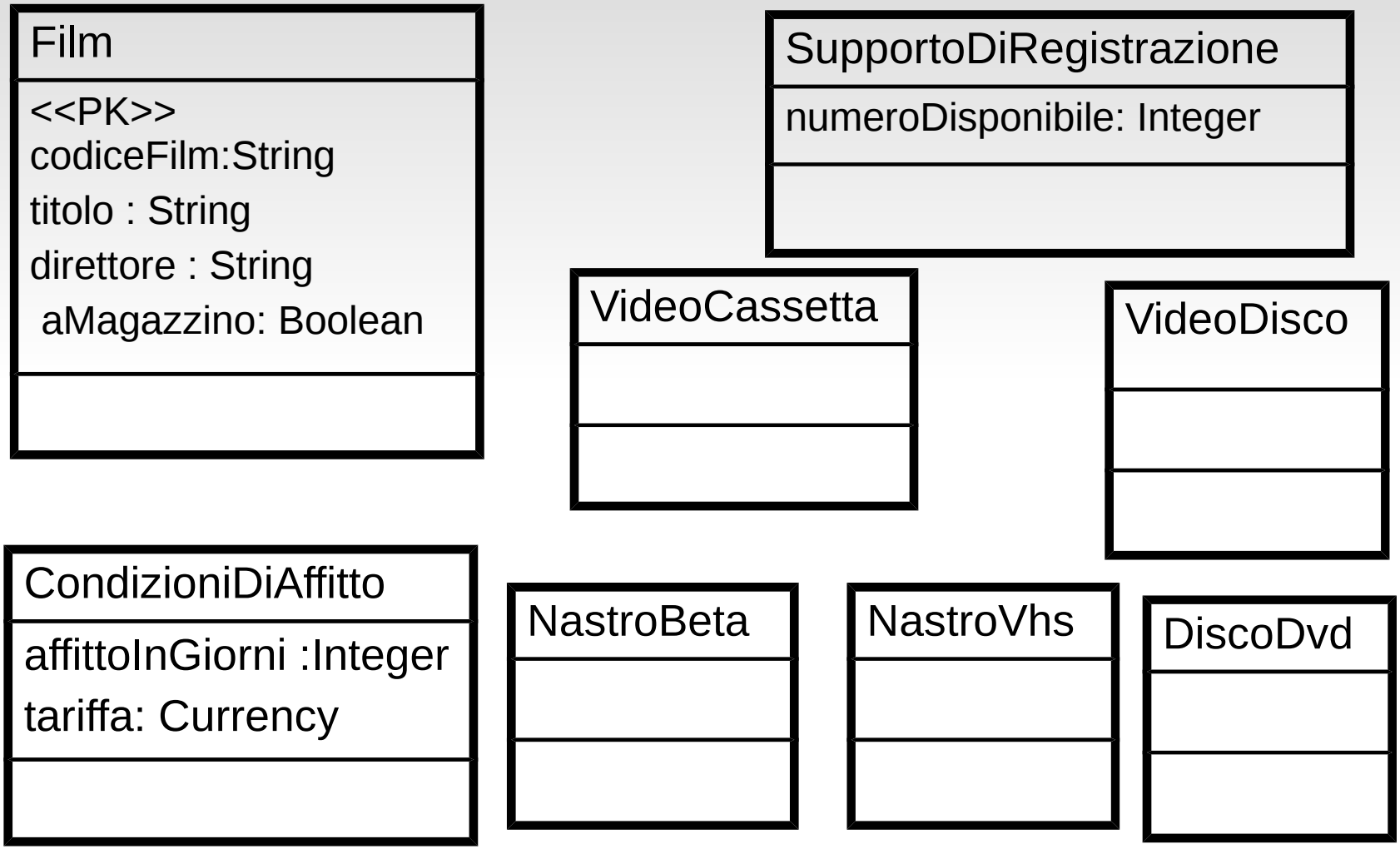
diagramma degli oggetti che modella l'esecuzione



Polimorfismo (Arlow..)



Esempio: affitto video



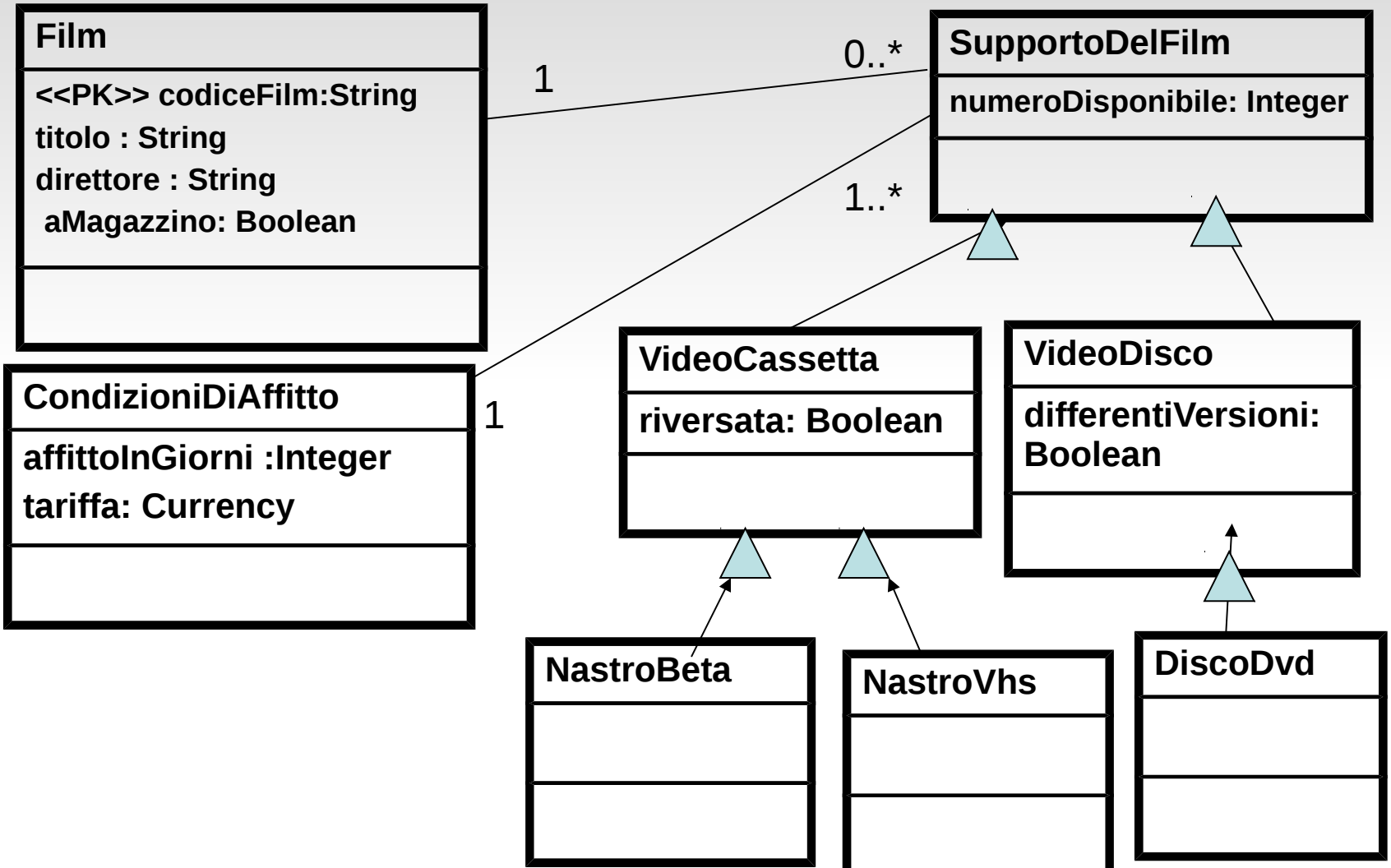
Esempio: affitto video

Si puo' immaginare di generalizzare i vari tipi di supporto ed ottenere una classe SupportoDiRegistrazione

Occorre sapere se la videocassetta e' originale o riversata

Un videoDisco puo'contenere piu' versioni di uno stesso film, in lingue diverse

Esempio: affitto video



Modellazione e specifica con diagramma degli oggetti

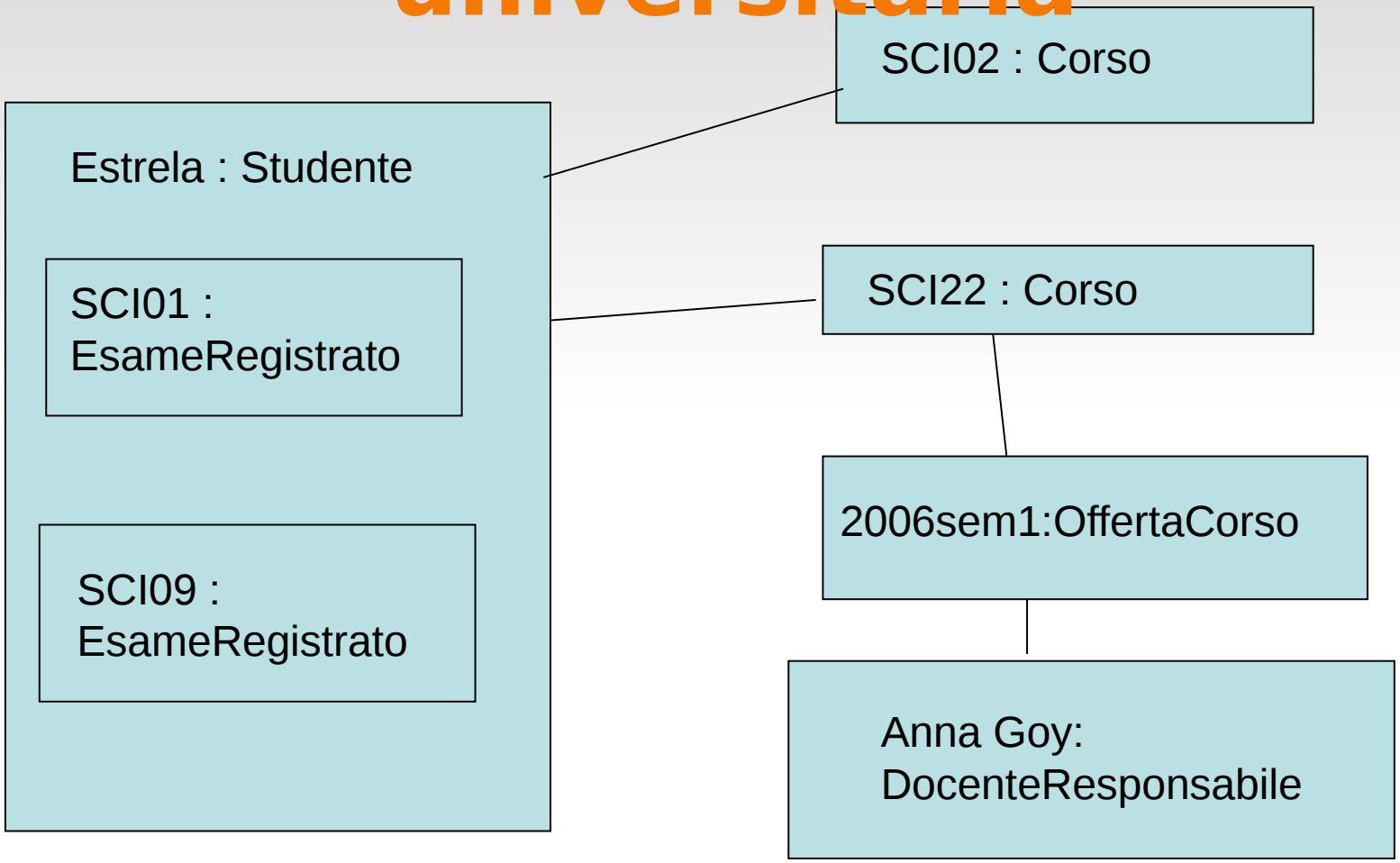
Solo a titolo d’esempio

Per mostrare relazioni complesse

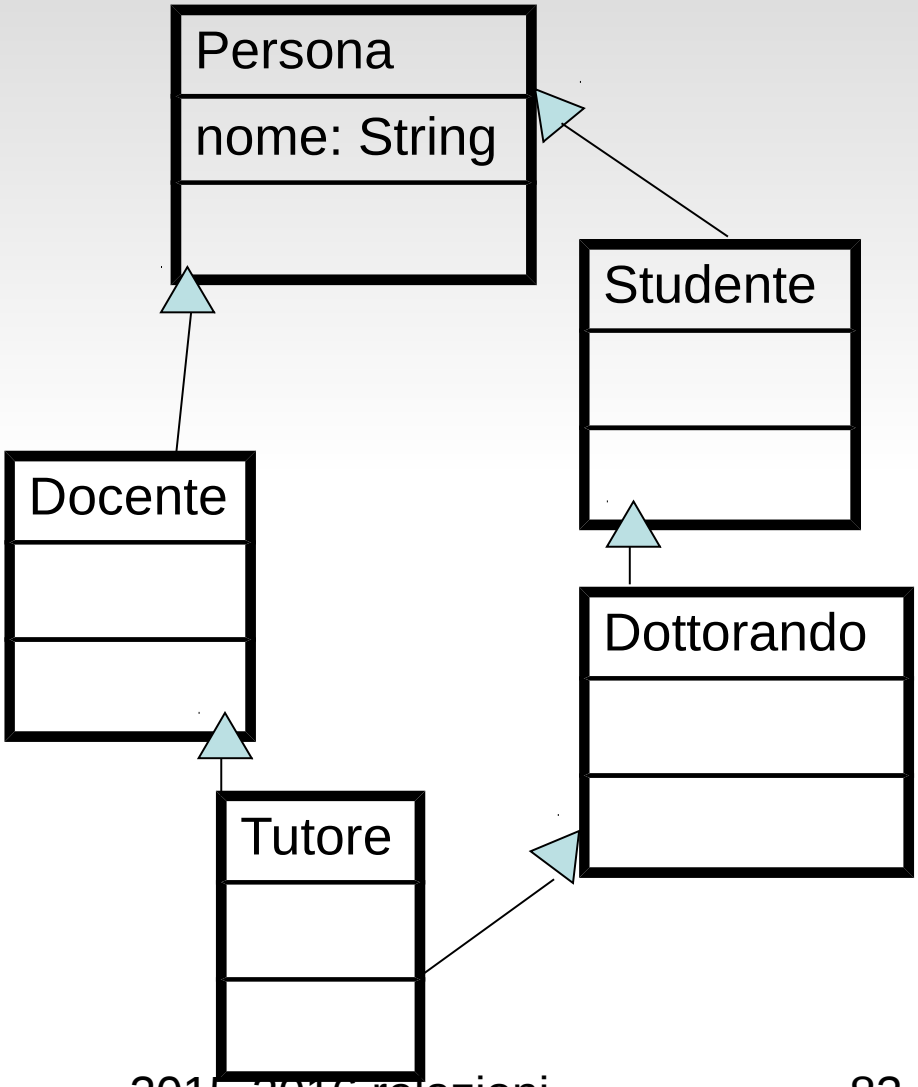
Per mostrare cambiamenti nel tempo

Per mostrare la collaborazione tra oggetti

Esempio: iscrizione universitaria



Ereditarieta’ multipla



Classificazione multipla

Nell'ereditarieta' multipla un oggetto puo' appartenere ad una sola classe (e costerebbe molto cambiarla)

Nella classificazione multipla ogni oggetto e' l'istanza (ha il comportamento) di piu' classi contemporaneamente

Il problema si verifica se la persona puo'avere classificazione diverse ortogonali tra loro: uomo/ donna, studente/ insegnante

Senza classificazione multipla, occorre definire tutte le possibili combinazioni : DonnaStudente, UomoInsegnante …

Interfaccia

Supponendo di avere una classe Veicolo e di voler modellare EssereUmano come Veicolo (fattorino), ha senso derivarlo da Veicolo?

No, introdurremmo troppi vincoli un essere umano ha molte prorieta' che non c'entrano con veicolo

Per stabilire relazioni tra classi disparate si usa l’elemento <<interface>>

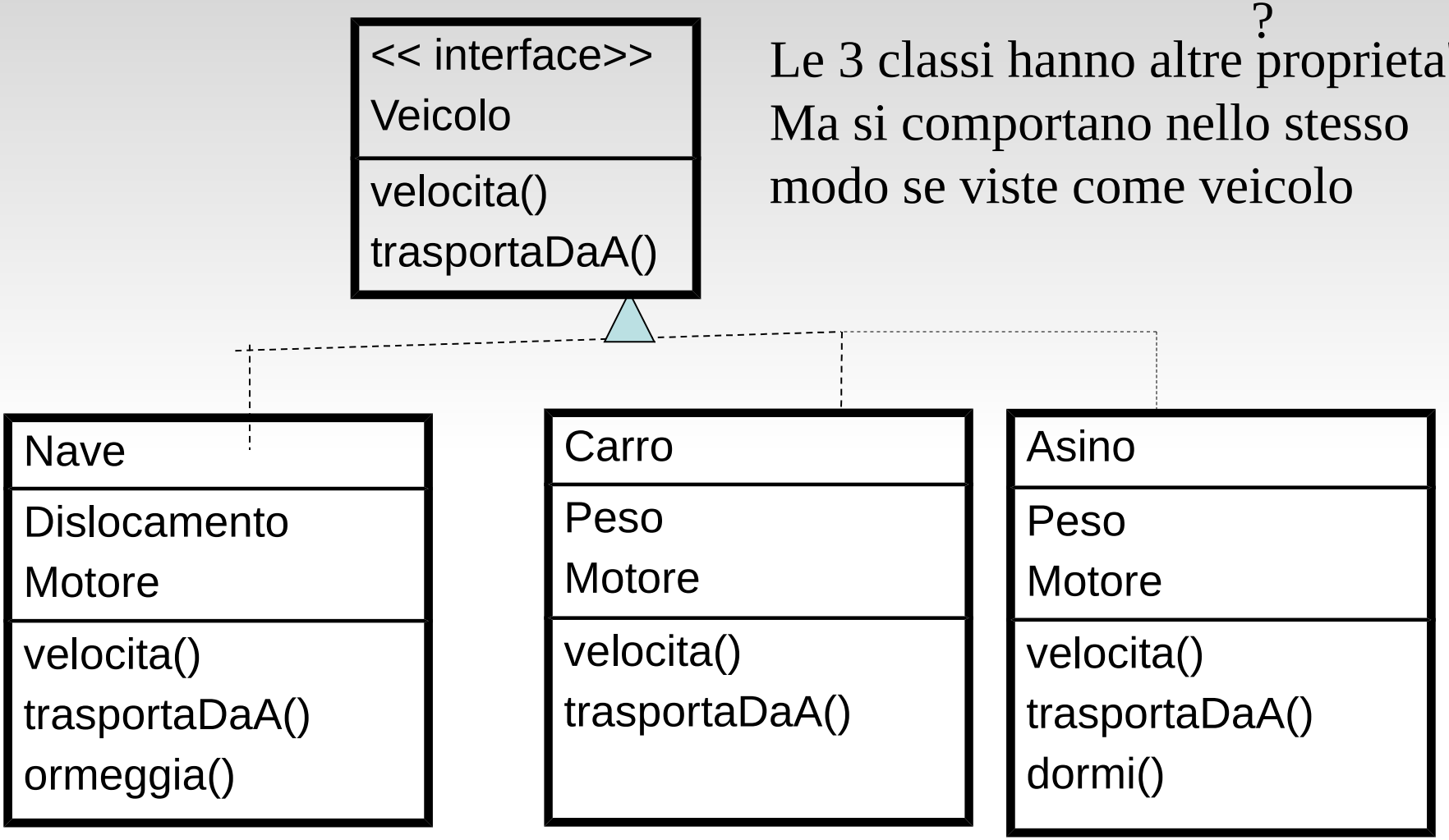
Interfaccia

Che cos'e' per voi una interfaccia?

In UML

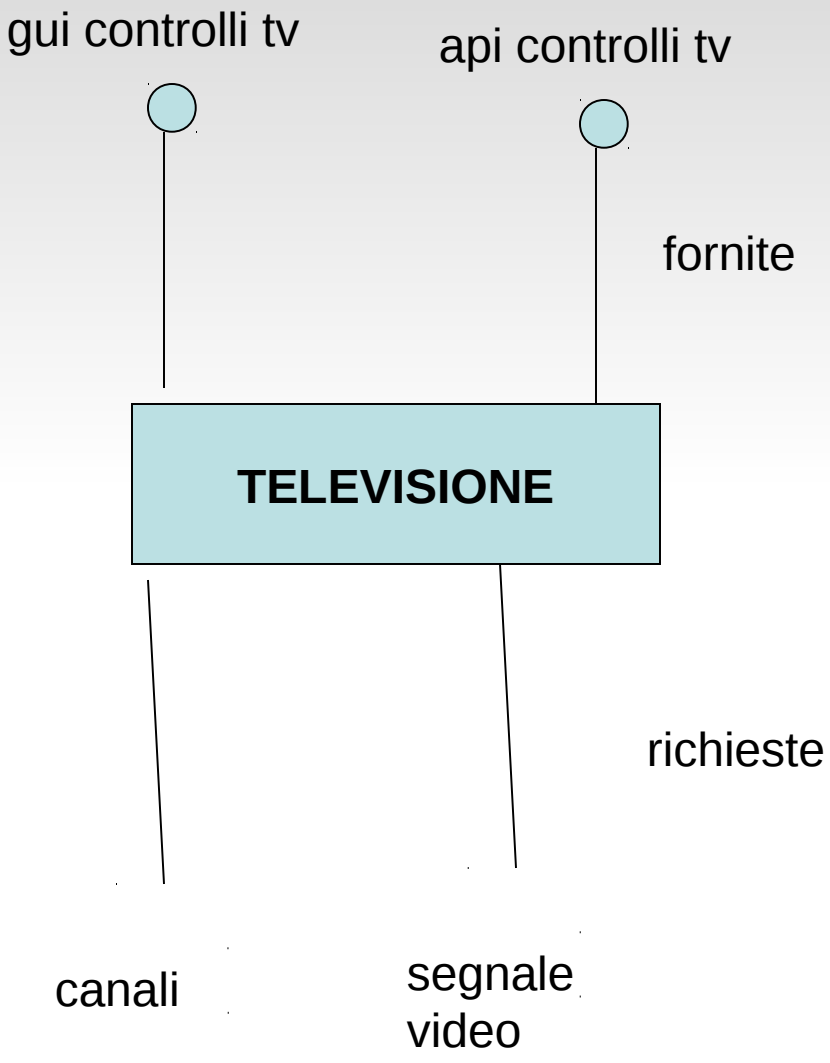
- Un elenco di operazioni
- Costituita solo da **Firme** di metodi (evt costanti)
- due classi che implementano la stessa interfaccia si comportano in modo identico per quanto concerne essa
- due classi arbitrarie possono implementare una stessa interfaccia senza che vi sia alcuna altra relazione tra esse
- Ovviamente entrambe le classi avranno anche altre proprieta'/operazioni, ma non ci interessano.

Interfaccia



Le 3 classi hanno altre proprieta' ?
Ma si comportano nello stesso modo se viste come veicolo

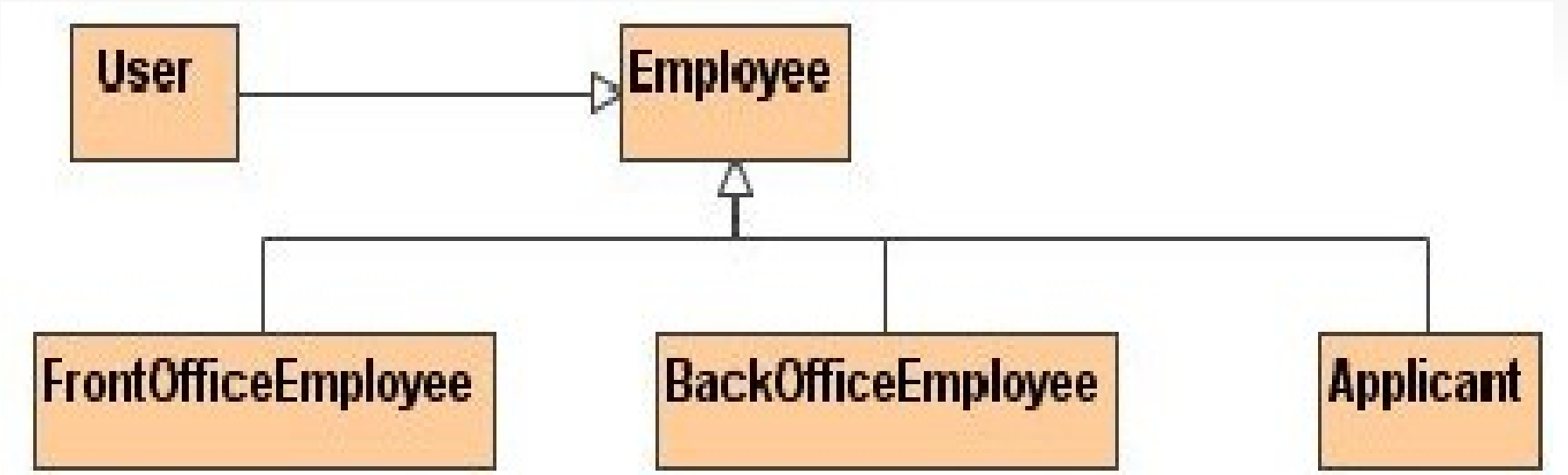
notazione estesa di classe



Televisione
<<interfacce fornite>> gui controlli tv api controlli tv <<interfacce richieste>> canali segnale video

esercizio 2

volendo creare una applicazione di e-government, e'il seguente un modello corretto?

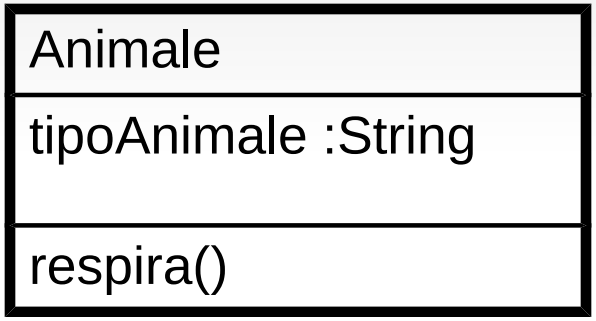


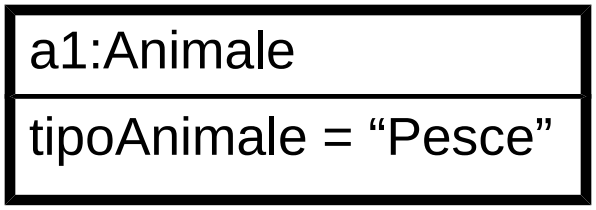
Discriminatore esplicito

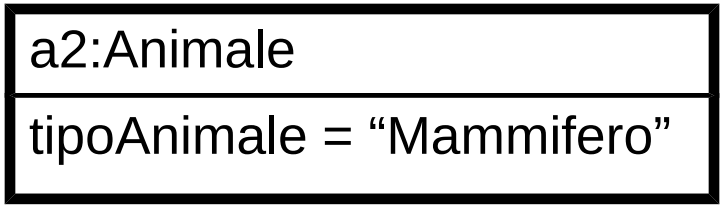
Discriminatore e' l'attributo (implicito) il cui valore determina l'appartenenza ad una o all'altra classe.

Se diventasse esplicito, non avremmo piu' bisogno delle varie classi ma dovremmo tenerne conto nel comportamento.

pro/contro ?

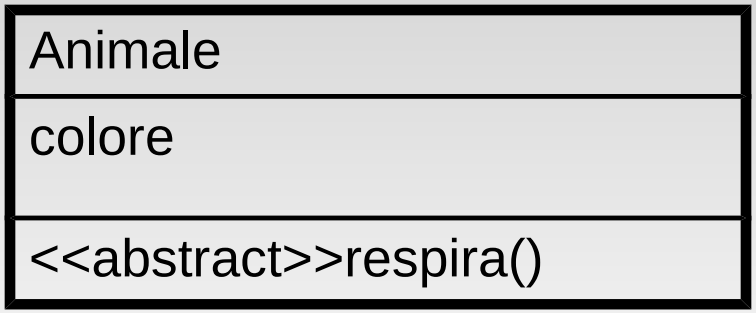


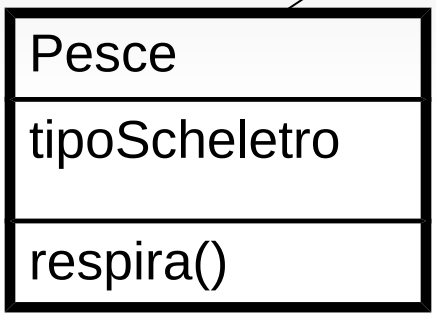





```
void respira() {
if (tipoAnimale = "Pesce") {
            //usa le branchie }
if (tipoAnimale = "Mammifero") {
            //usa I polmoni }
……
}
```

Metodo astratto



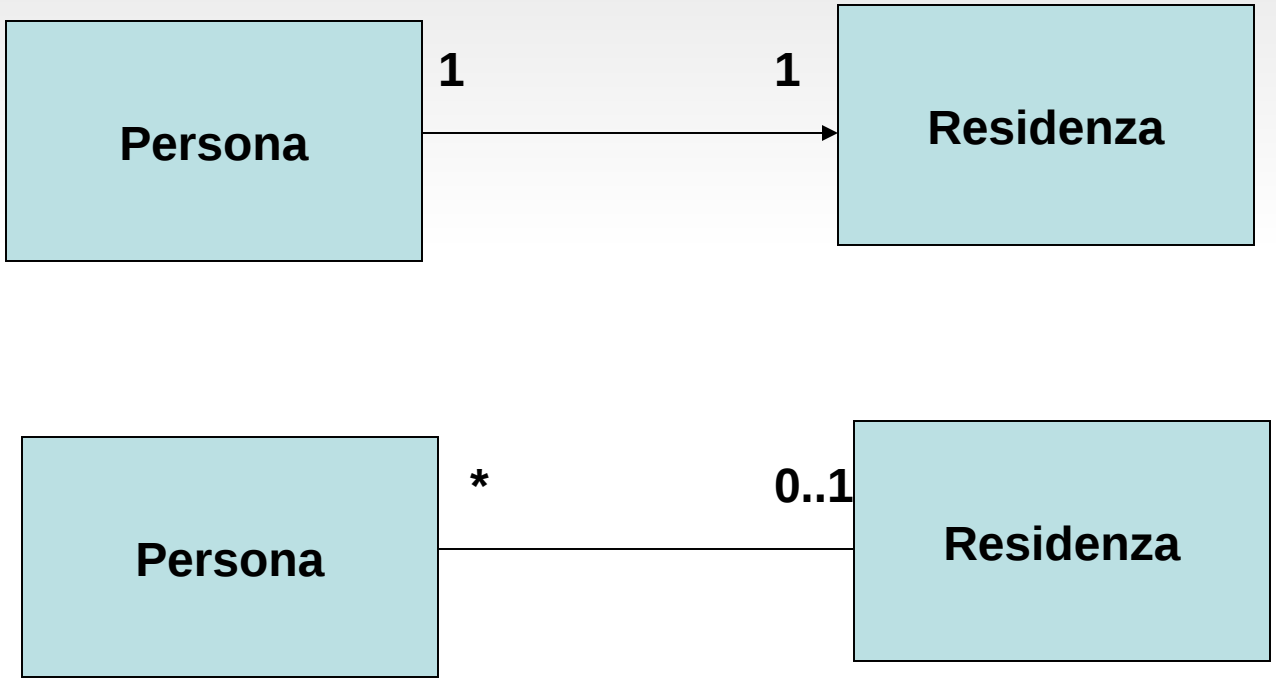



```
void respira() {
//usa le branchie
}
```

```
void respira() {
//usa i polmoni
}
```

esercizio

differenza tra questi due modelli:



esercizio

modellare tassonomie (classificare) tre tipi di frutta e tre di verdura da vari aspetti: botanico, coltivazione , cottura

esercizi

modellare:

un automobile puo’ essere fornita in configurazione standard, con un insieme di allestimenti, accessori di serie, oppure personalizzata: in tal caso alcuni accessori possono essere sostituiti, altri aggiunti, ed ognuno ha un costo di sostituzione o aggiunta