

UML: Diagramma delle classi; Diagramma degli oggetti

Vincenzo Gervasi, Laura Semini
Ingegneria del Software
Dipartimento di Informatica
Università di Pisa

Riassunto lezione precedente

Outline della lezione

- Lezioni precedente:
 - Requisiti
- Questa lezione e le prossime
 - Descrizione del dominio

Classi e oggetti

- Una classe cattura
 - un concetto nel dominio del problema o della realizzazione
- Una classe descrive
 - un insieme di oggetti con caratteristiche simili
 - cioè oggetti che hanno lo stesso tipo
- Un oggetto è un'entità caratterizzata da
 - Un'identità
 - Uno stato
 - Un comportamento

Classificatori e istanze

- Le classi sono classificatori
- Gli oggetti sono istanze

- Modellare a livello dei classificatori significa vincolare i modelli a livello di istanza

Classificatori e istanze (continua)

- In teoria i modelli a livello di istanza possono esistere solo in un ambiente definito dai modelli a livello dei classificatori
 - Le variabili in un linguaggio fortemente tipato possono vivere solo in un ambiente in cui il tipo è definito
- In pratica UML è più flessibile
 - Si possono introdurre oggetti senza definirne le classi
 - UML tollera le inconsistenze

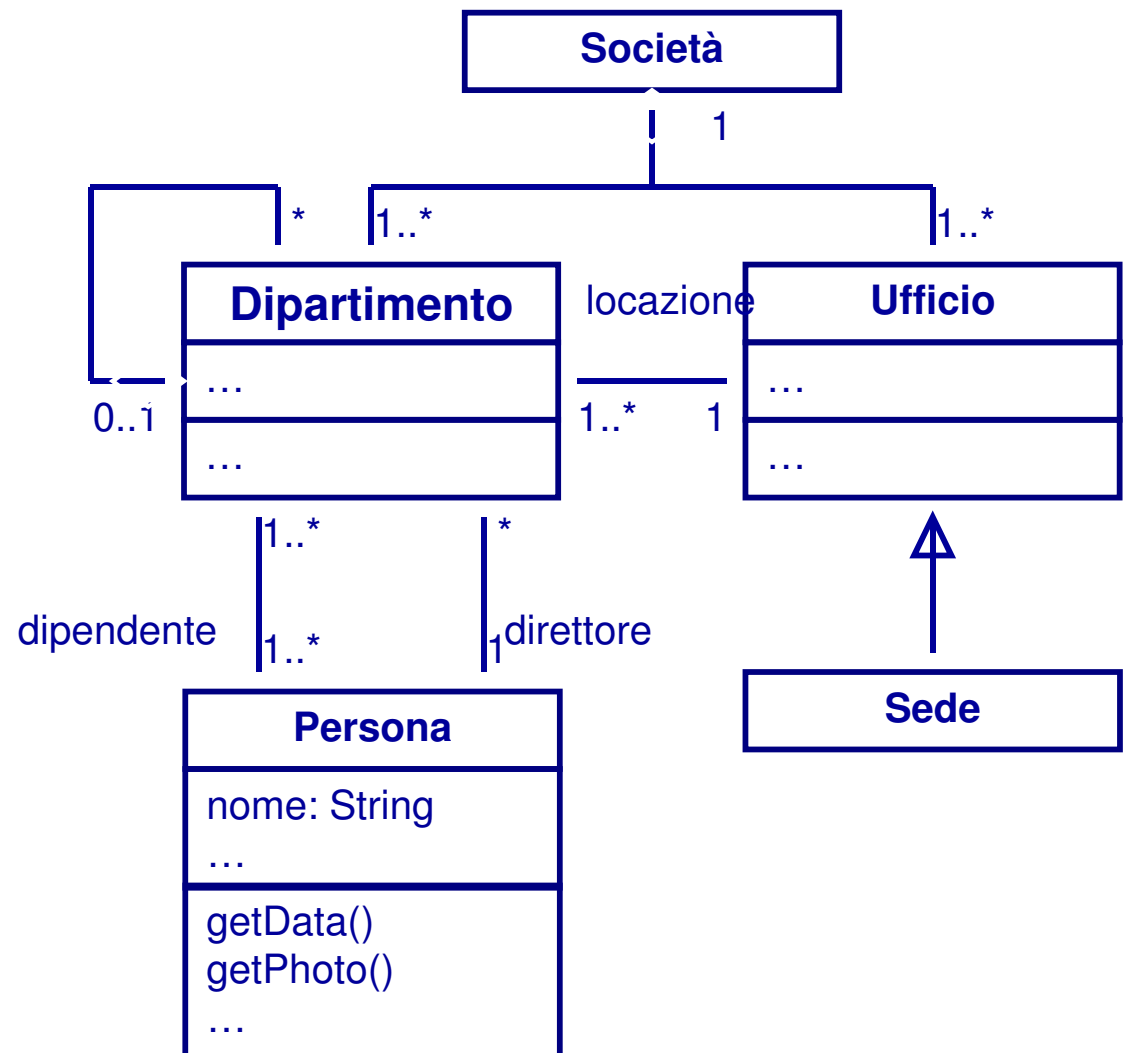
Classi UML vs entità (DB)

- DB: le classi sono intese come collezioni. Sottointeso che
 - ci sono più istanze.
 - ci sono operazioni per visitare tutte le istanze.
- Da cui, per esempio, nome singolare vs nome plurale.
- La differenza è significativa più in prospettiva di progettazione che di descrizione del dominio.
 - In progettazione OO e quindi in UML uso "ListaDiQcosa" come aggregato di "Qcosa"
- Un esempio dal dominio:



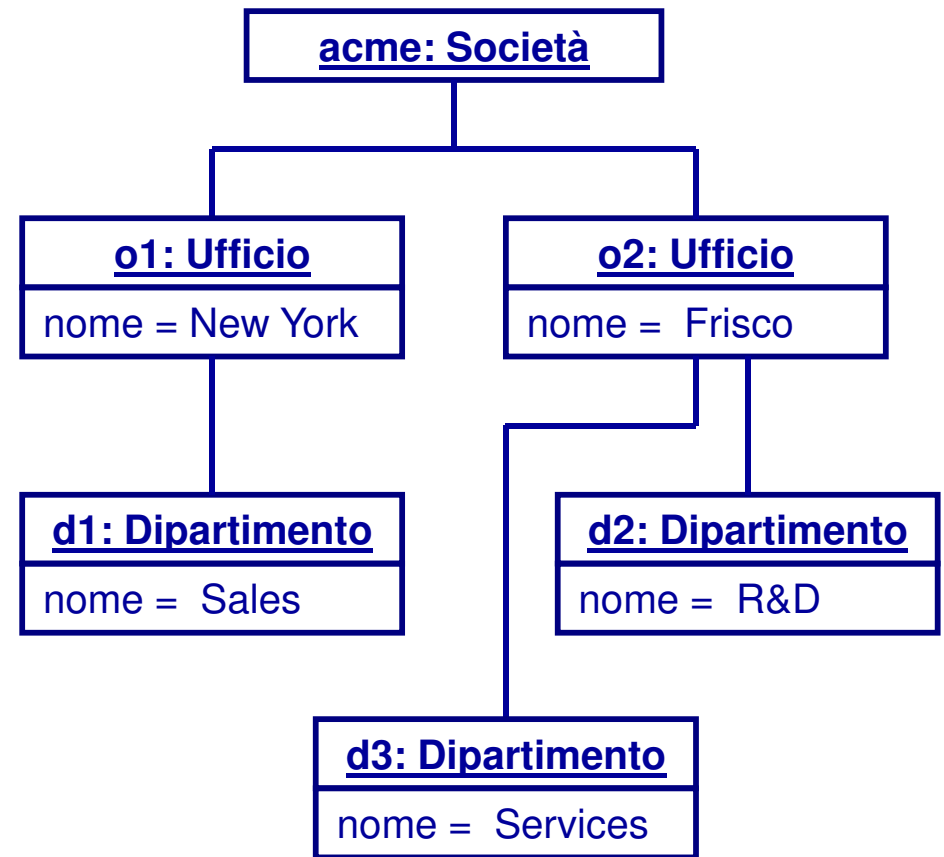
Esempio: classi

- Una società è formata da dipartimenti e uffici
- Un dipartimento ha un direttore e più dipendenti
- Un dipartimento è situato in un ufficio
- Esiste una struttura gerarchica dei dipartimenti
- Le sedi sono uffici

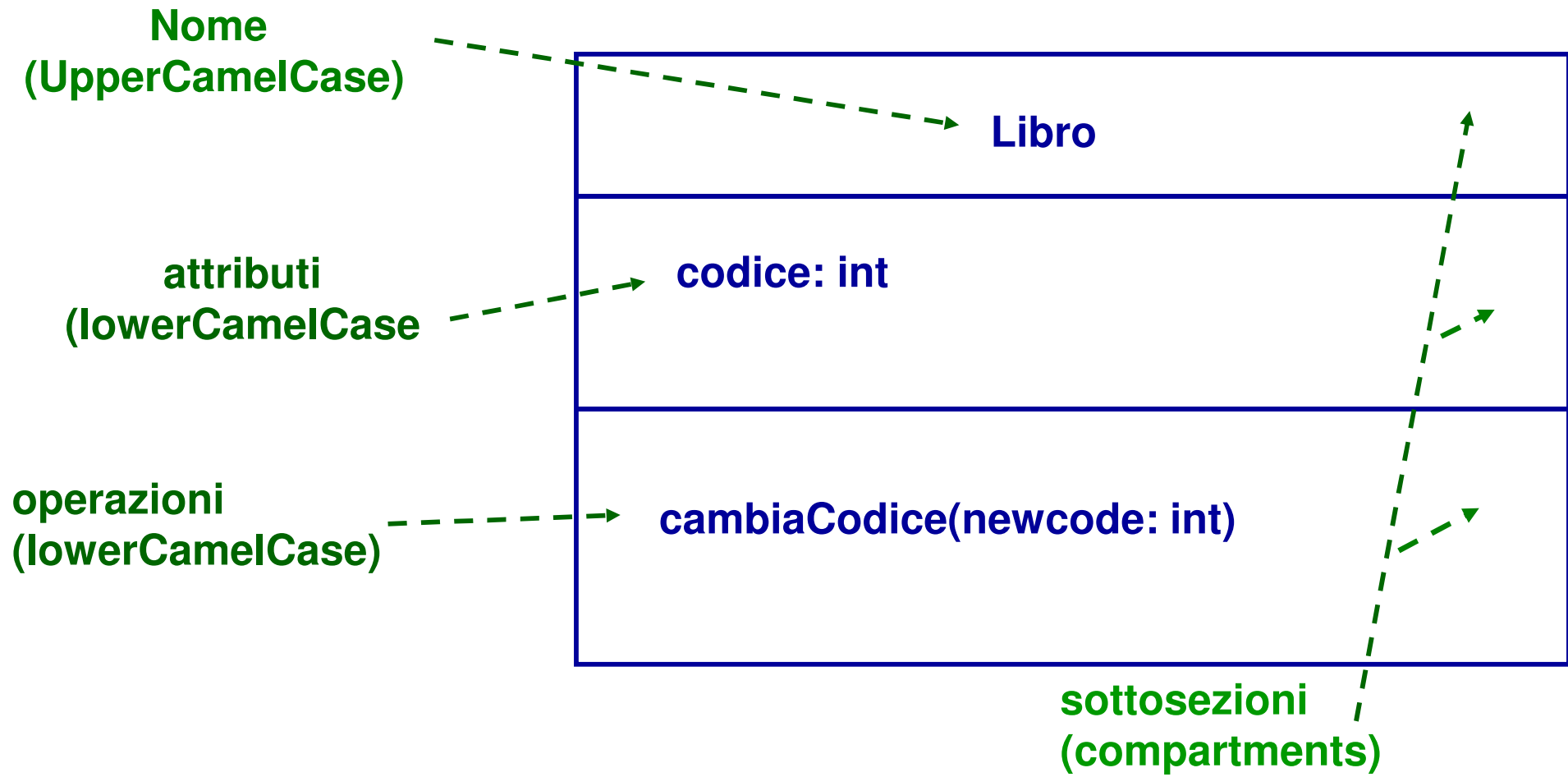


Esempio: oggetti

- Una società reale, ACME
- Ha due Uffici
- Il dipartimento vendite è a New York
- I dipartimenti Ricerca&Sviluppo e Servizi sono a San Francisco



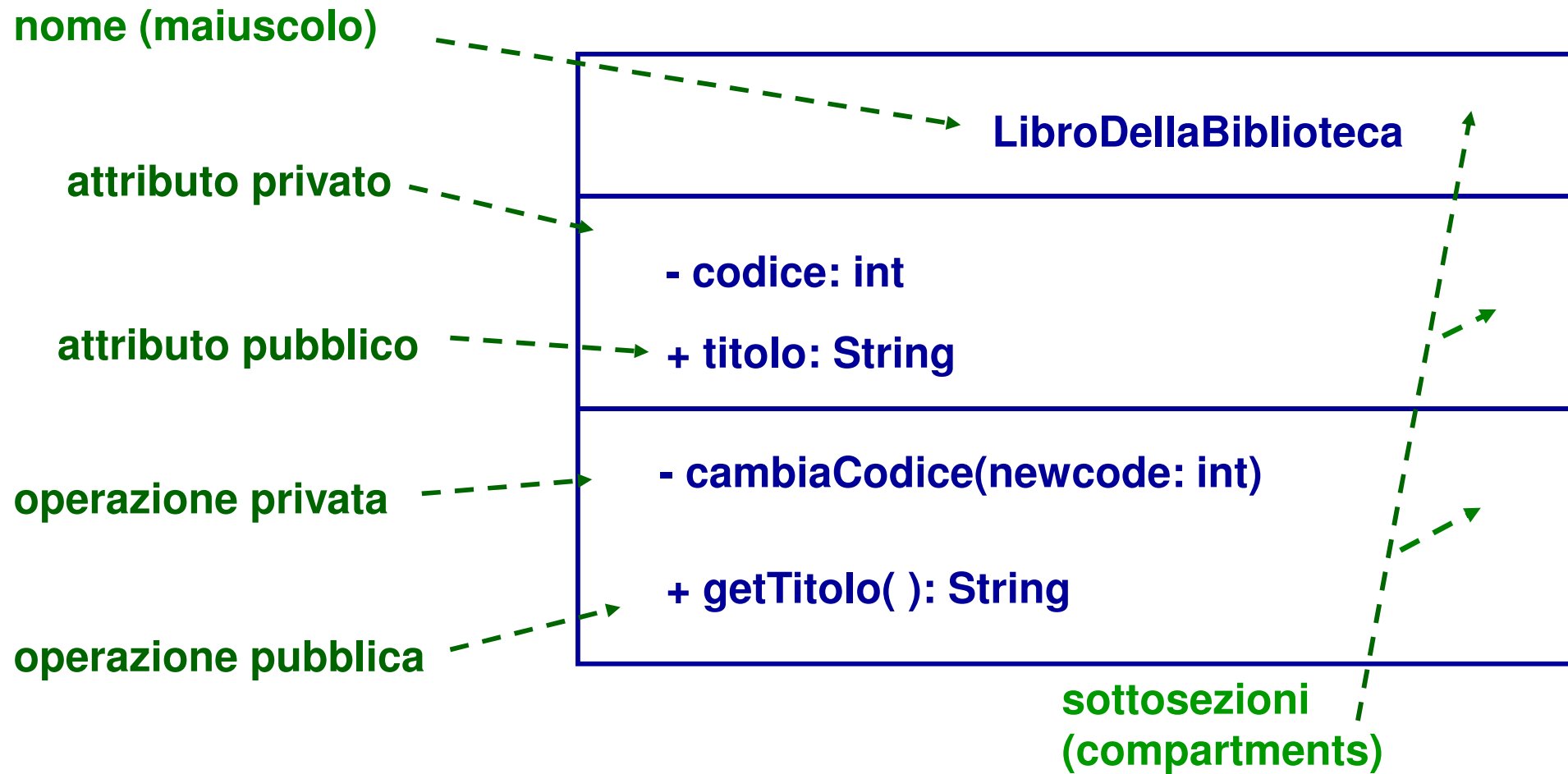
Sintassi della classe



Se non interessano attrib. e operazioni

Libro

Sintassi della classe



Semantica

- Un oggetto è un'entità caratterizzata da
 - Un'identità, uno stato, un comportamento
- L'identità si vede a livello di istanza
- Gli attributi definiscono lo stato dell'oggetto
- Le operazioni definiscono il suo comportamento

Spazio dei nomi

- Un classificatore è uno spazio di nomi
 - Gli elementi contenuti hanno un nome unico.
- Per esempio un Package
 - è un costrutto di strutturazione
 - può contenere package innestati e altri elementi (classi, diagrammi...).

Visibilità

- Un elemento è visibile all'esterno dello spazio di nomi che lo contiene, in accordo con il suo tipo di visibilità
 - + public: tutti
 - # protected: i discendenti
 - private: solo nell'elemento stesso
 - ~ package: nello stesso package

Sintassi attributi

- `visibilità nome: tipo [molteplicità]= valoreIniziale {proprietà}`

obbligatorio

molteplicità:
array di valori

colore: Saturazione [3]

nome: String [0..1] 0 per permettere valore null

[1] può essere omesso

proprietà

{ordered}

{>=0}

Visibilità di attributi e operazioni

- In modo simile alla visibilità della classe
 - + public: accessibile ad ogni elemento che può vedere e usare la classe
 - # protected: accessibile ad ogni elemento discendente
 - - private: solo le operazioni della classe possono vedere e usare l'elemento in questione
 - ~ package: accessibile solo agli elementi dichiarati nello stesso package

Esempi

numero di tipo intero

$n: \text{Integer}$

numero intero positivo

$n: \text{Integer}\{>= 0\}$

contatore positivo, inizialmente a 0

$n: \text{Integer} = 0 \{>= 0\}$

Esempi

Punti del quadrante positivo, pubblico

+ puntiQuadPos : Integer [2] {>= 0}

sequenza ordinata di 10 interi compresi tra 3 e 33,
privato

- seq: Integer[10] {>= 3, <= 33, ordered}

Sintassi operazioni

visibilità nome (listaParametri) : tipoRitorno {proprietà}

obbligatori

può essere vuota

listaParametri

default

direzione nome: tipo = default

in, out, inout

valore assegnato al parametro in
assenza di argomento

Esempi

Metodo pubblico che restituisce la somma di 2 interi

+ `sum (a: Integer, b: Integer) : Integer`

sum con 10 valore di default del secondo parametro

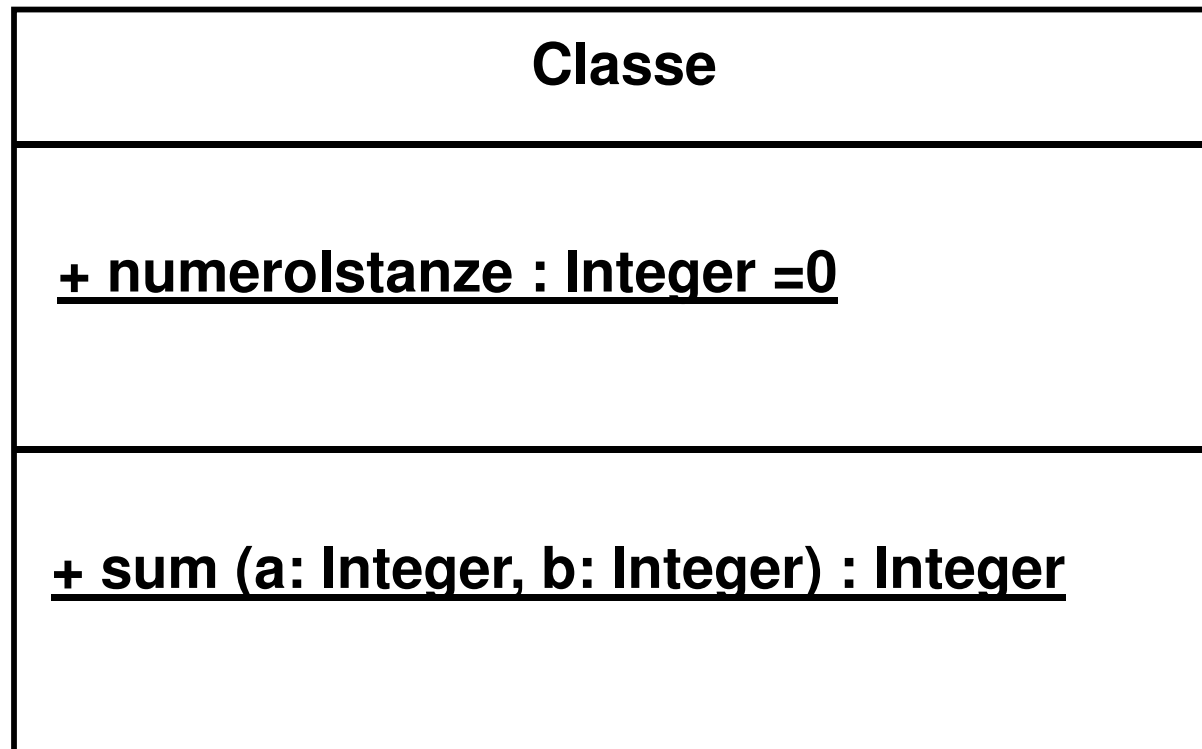
+ `sum (a: Integer, b: Integer =10) : Integer`

Metodo privato che restituisce un oggetto di tipo Gra

- `gra () : Gra`

attributi e operazioni con ambito di classe (statici)

- sottolineati



Esempio

class attribute

instance attribute

class operation

instance operation

Job	
	<u>maxCount: Integer = 0</u>
	jobID: Integer
	<u>create () { jobID = maxCount++ }</u>
	schedule ()

Diagrammi degli oggetti

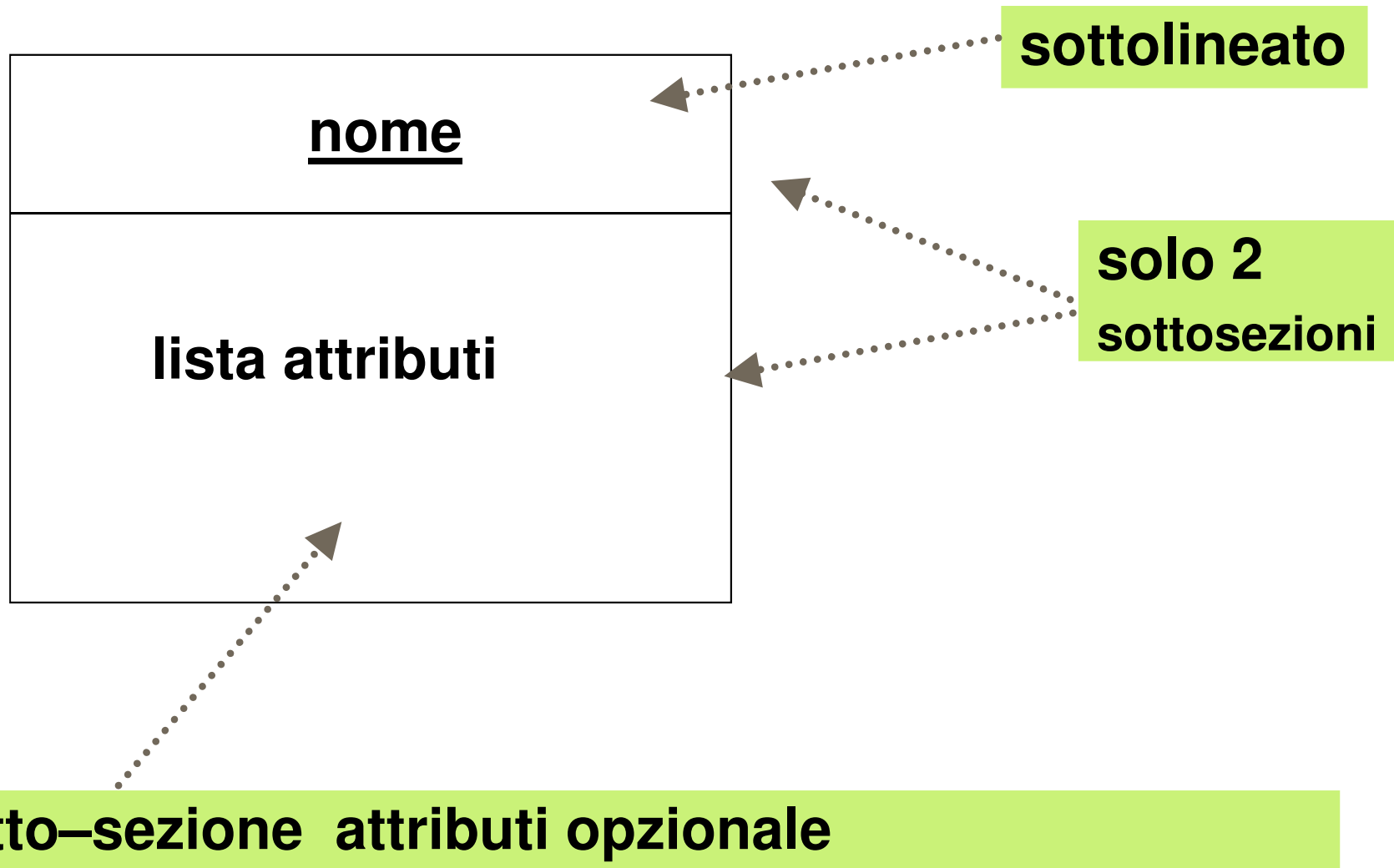


Diagramma degli oggetti: nome e tipo

nomeoggetto: Nomeclasse

nomeoggetto

: Nomeclasse

nomeoggetto: Nomeclasse, Nomeclasse

minou: Gatto

minou

: Gatto

minou: Gatto, Cantante

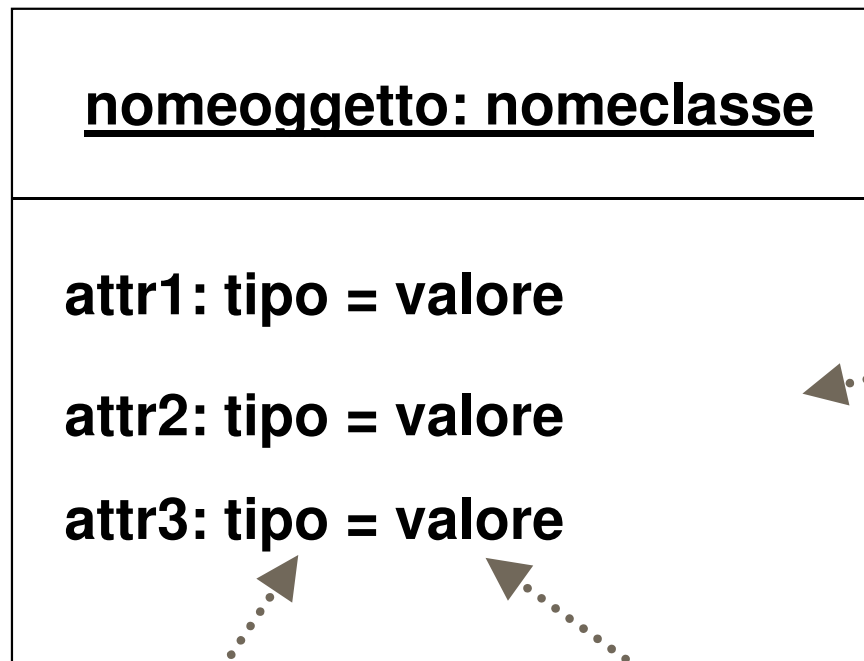
eral formulation of UML, an object may have one or more direct classes—**multiple classification**. The object behaves as if it belonged to an implicit class that was a child of each of the direct classes—effectively, **multiple inheritance** without the need to actually declare the new class.

Ereditarietà multipla

Multiple inheritance

If a classifier has more than one parent, it inherits from each one (Figure 4-8). Its features (attributes, operations, and signals) are the union of those of its parents. If the same class appears as an ancestor by more than one path, it nevertheless contributes only one copy of each of its members. If a feature with the same signature is declared by two classes that do not inherit it from a common ancestor (independent declarations), then the declarations **conflict** and the model is **ill formed**. UML does not provide a **conflict** resolution rule for this situation because experience has shown that the designer should explicitly resolve it. Some languages, such as Eiffel, permit conflicts to be explicitly resolved by the programmer, which is much safer than implicit conflict resolution rules, which frequently lead to surprises for the developer.

Diagrammi degli oggetti: attributi



singoli attributi

opzionali

Il valore può essere omissso

Il tipo è ridondante e può essere omissso

Classi e Oggetti

Punto
x : Real y: Real colore: Saturazione [3]

<u>p1: Punto</u>
x =3,14 y= 2,78

<u>p2: Punto</u>
x =1 y= 2

Individuare le classi di analisi

- Cosa sono le classi di analisi
- Come sono fatte
- Tecniche per individuarle
 - Nome-verbo
 - CRC

Cosa sono le classi di analisi

- Corrispondono a concetti concreti del dominio:
 - Per esempio i concetti descritti nel glossario
 - Normalmente, ciascuna classe di analisi sarà raffinata in una o più classi di progettazione.
- Evitare di introdurre delle classi di progettazione

Classi di analisi: caratteristiche

- Astrazione di uno specifico elemento del dominio
- Numero ridotto di responsabilità (funzionalità)
- Evitare le classi “onnipotenti”
 - Attenzione quando si chiamano “sistema”, “controllore”,
- Evitare funzioni travestite da classi
- Evitare gerarchie di ereditarietà profonde (≥ 3)
- Coesione e disaccoppiamento
 - Tenere responsabilità simili in una classe
 - Limitare interdipendenze tra classi

Classi di analisi: livello di dettaglio

- Operazioni e attributi solo quando veramente utili
 - Le classi di analisi dovrebbero contenere attributi e operazioni ad “alto livello”
 - Limitare la specifica di tipi, valori, etc.
 - Non inventare mai niente!

Identificazione delle classi

- Problema classico delle prime fasi di sviluppo
- Approccio data driven
 - Si identificano tutti i dati del sistema e si dividono in classi (ad esempio mediante identificazione dei sostantivi)
- Approccio responsibility driven
 - Si identificano le responsabilità e si dividono in classi (ad esempio mediante CRC Cards)

Analisi nome-verbo

- Sostantivi → classi o attributi
- Verbi → responsabilità o operazioni
- Passi:
 - Individuazione delle classi
 - Assegnazione di attributi e responsabilità alle classi
 - Individuazione di relazioni tra le classi

Analisi nome-verbo

- Problemi ricorrenti :
 - Tagliare le classi inutili
 - Trattare i casi di sinonimia
 - Individuare le classi nascoste cioè le classi implicite del dominio del problema che possono anche non essere mai menzionate esplicitamente
 - In un sistema di prenotazione di una compagnia di viaggi si potrebbe parlare di prenotazione, richiesta, ma tralasciare il termine ordine

Tagliare le classi inutili

- Elenco dei sostantivi
- Eliminazione dei sostantivi riconosciuti come
 - Sinonimi
 - Eventi o operazioni
 - Metalinguaggio (sistema)
 - Inutili (estranei al sistema)
 - Attributi (titolo del libro....)

Individuazione classi: chiavi magnetiche

Per motivi di sicurezza, un'organizzazione ha deciso di realizzare un sistema secondo il quale a ogni dipendente è assegnata una chiave magnetica per accedere (aprire) determinate stanze. I diritti di accesso dipenderanno in generale dalla posizione e dalle responsabilità del dipendente. Quindi sono necessarie operazioni per modificare i diritti di accesso posseduti da una chiave se il suo proprietario cambia ruolo nell'organizzazione.

Organizzazione

Dipendente

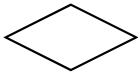
Chiave

Stanza

Ruolo

DirittiDiAccesso

Relazioni

- Una relazione rappresenta un legame
 - tra due o più oggetti
 - normalmente istanze di classi diverse
- Dal punto di vista matematico
 - una relazione binaria tra A e B è un sottoinsieme del prodotto cartesiano $A \times B$
- UML fornisce notazioni grafiche per
 - prodotti cartesiani, a livello di classificatori
 - coppie, a livello di istanza
 - anche triple, quadruple, etc (associazioni n-arie) 

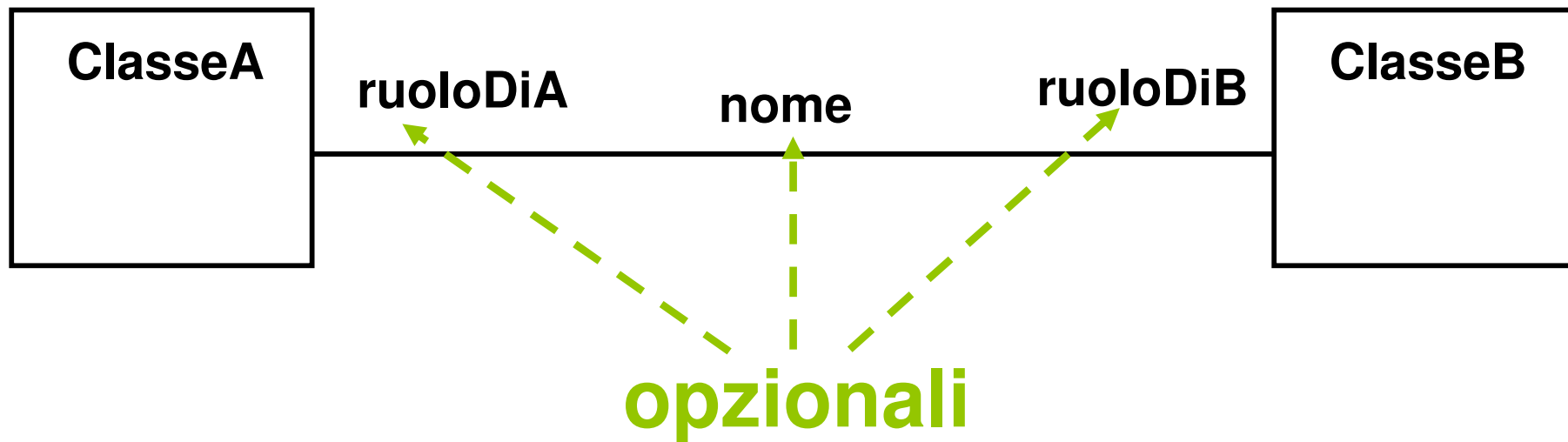
Relazioni

- Tra elementi di un modello

Tra Classi	Tra oggetti
Associazione	Collegamento
Aggregazione	
Composizione	
Generalizzazione	

- Dipendenza (uso, realizzazione, istanza)
- Flusso, estensione, realizzazione **Non ora**

Associazione (binaria): sintassi



- Almeno uno tra nome o ruoli, raramente entrambi.
- Il ruolo serve a parlar meglio dell'oggetto associato, soprattutto se gli oggetti sono della stessa classe.

Associazione (binaria): esempio

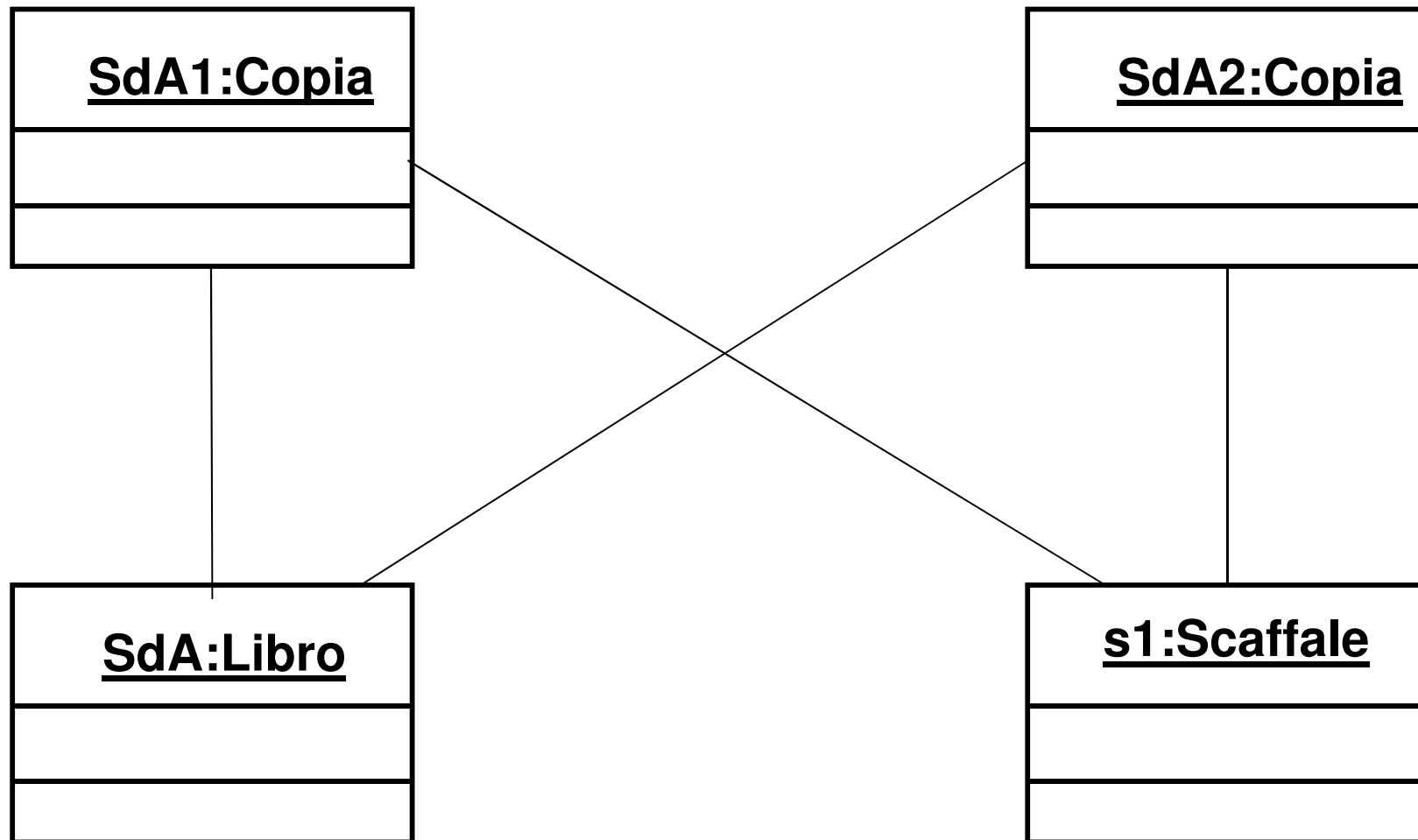


- se i è un Impiegato, e c una Chiave
- e c è stata consegnata ad i ,
- si può dire che i è il proprietario di c e che c è la chiave dell'ufficio di i

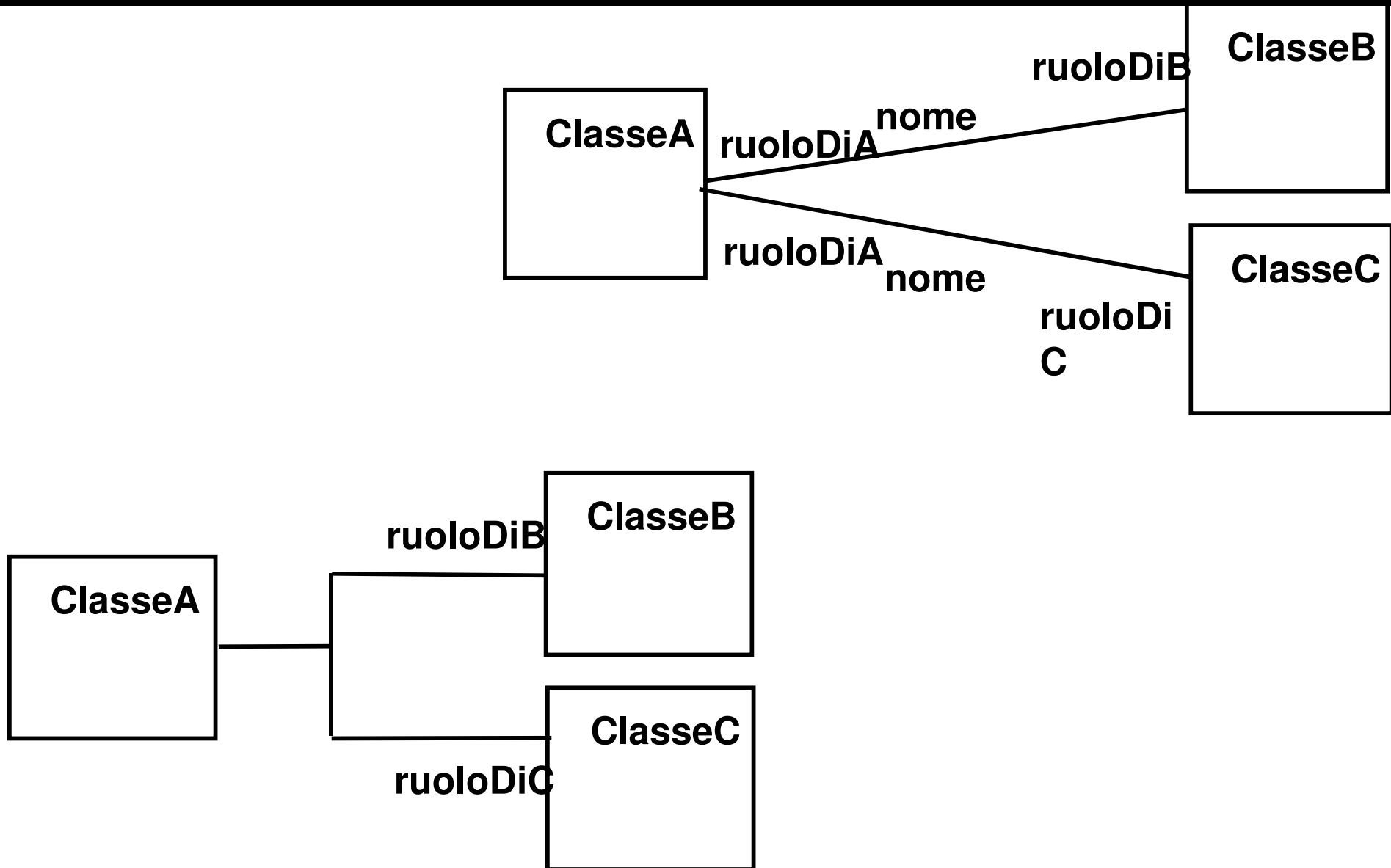
Associazione (binaria): nome e ruoli

- nome e ruoli: lowerCamelcase
- nome: normalmente un verbo
- ruolo: normalmente un sostantivo
- Formalmente opzionali,
 - è utile ci sia o il nome dell'associazione o l'indicazione dei ruoli
 - inutile entrambi

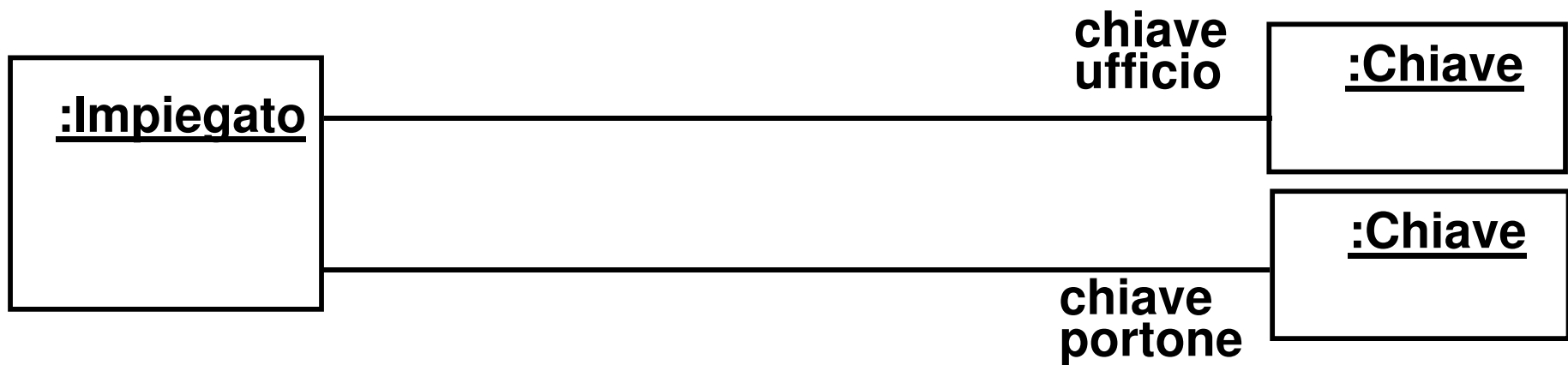
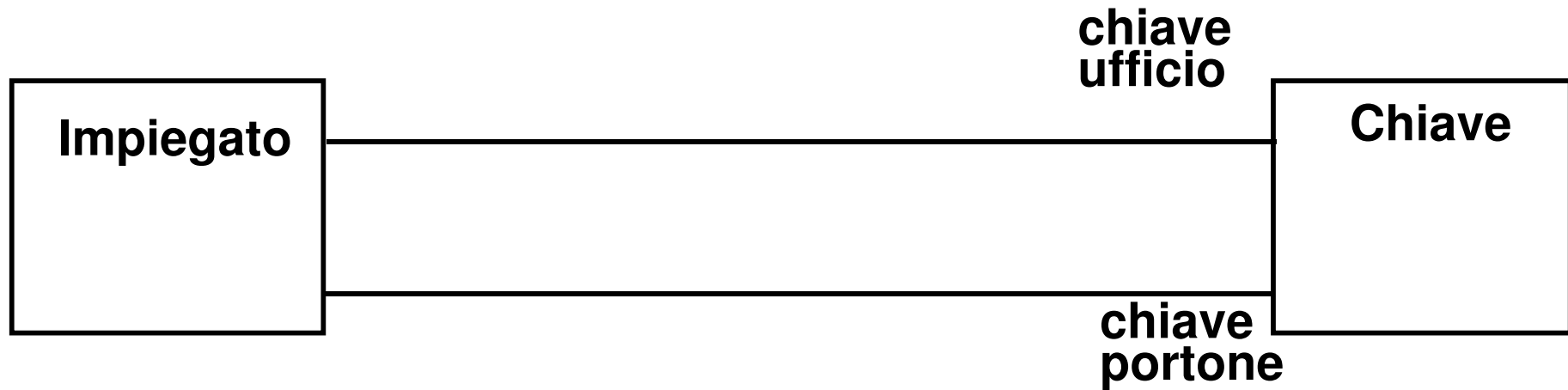
Associazione a livello istanza: collegamento



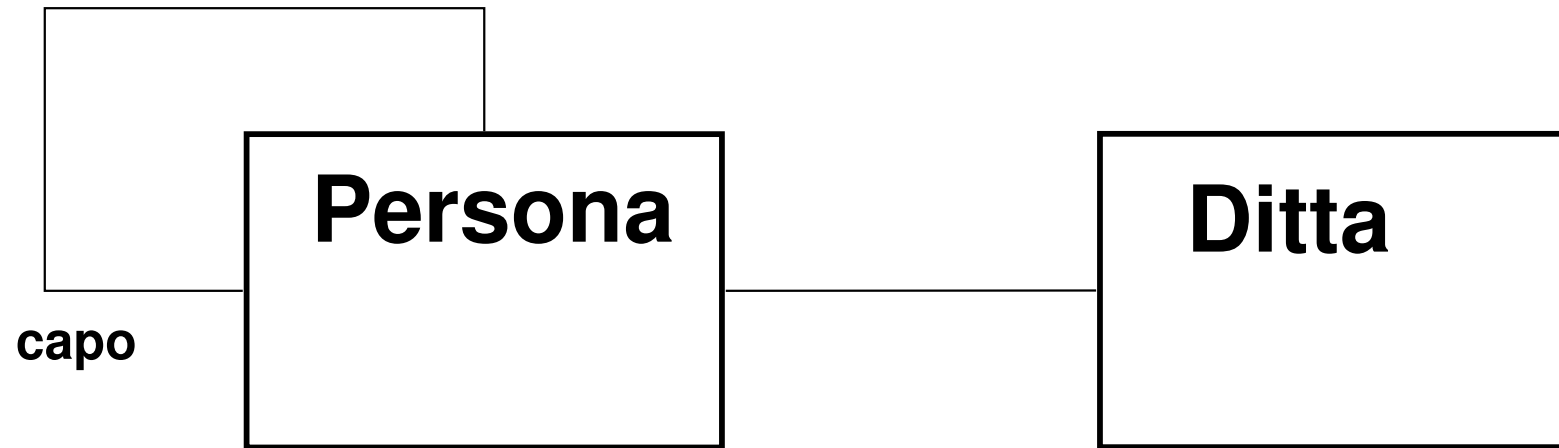
2 associazioni: stile



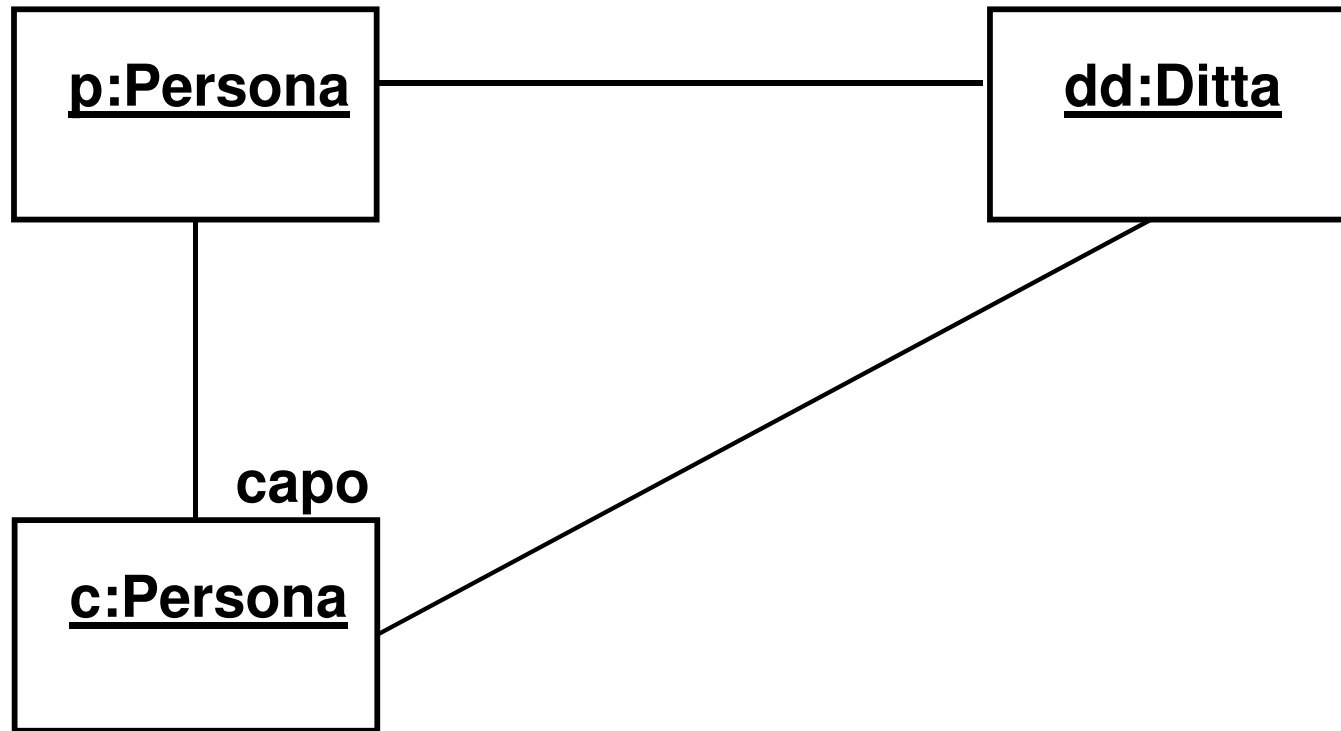
Associazione: caso particolare ruoli importanti



Associazione riflessiva ruoli importanti



A livello di oggetti



Associazioni: direzione



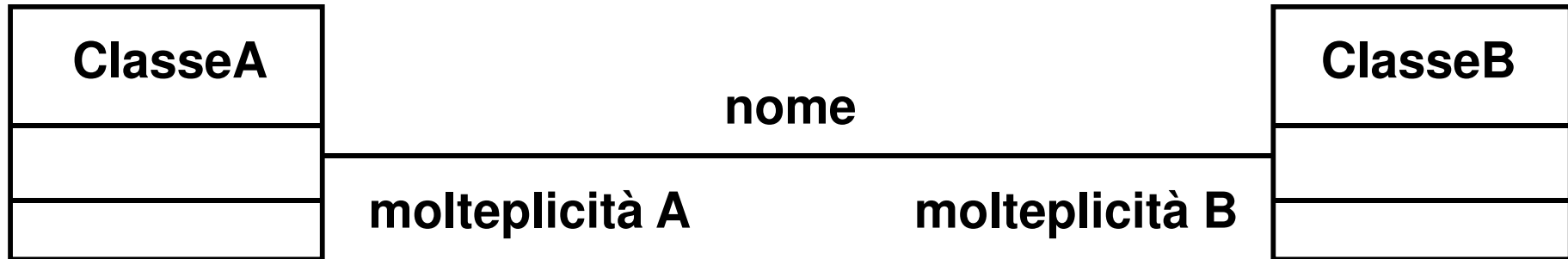
Potrebbe essere letto:

(ogni) telefonino possiede un italiano [Altan]

Uso una freccia di *direzione di lettura* per disambiguare



Associazioni: vincoli di molteplicità



**Numero di oggetti coinvolti
nell'associazione
in un dato istante**

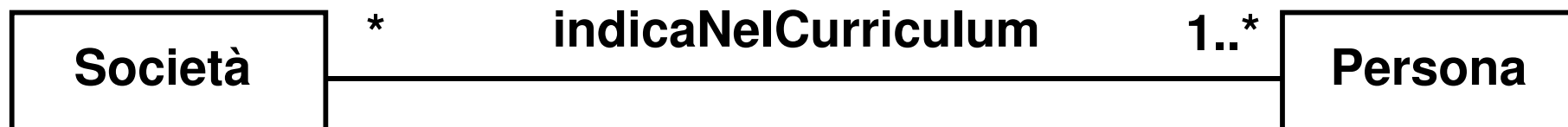
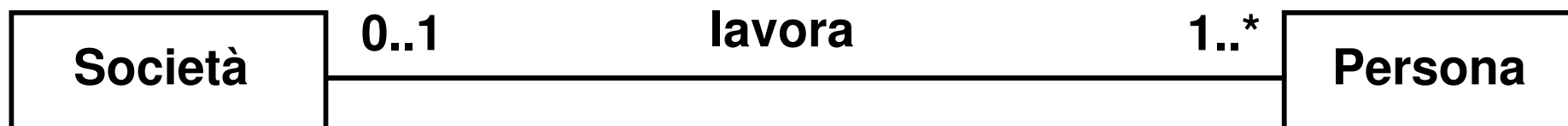
Molteplicità, un esempio



- Un oggetto Società può essere in relazione con molti oggetti Lavoratore
- Un oggetto Lavoratore può essere in relazione con un solo oggetto Società

in un dato istante

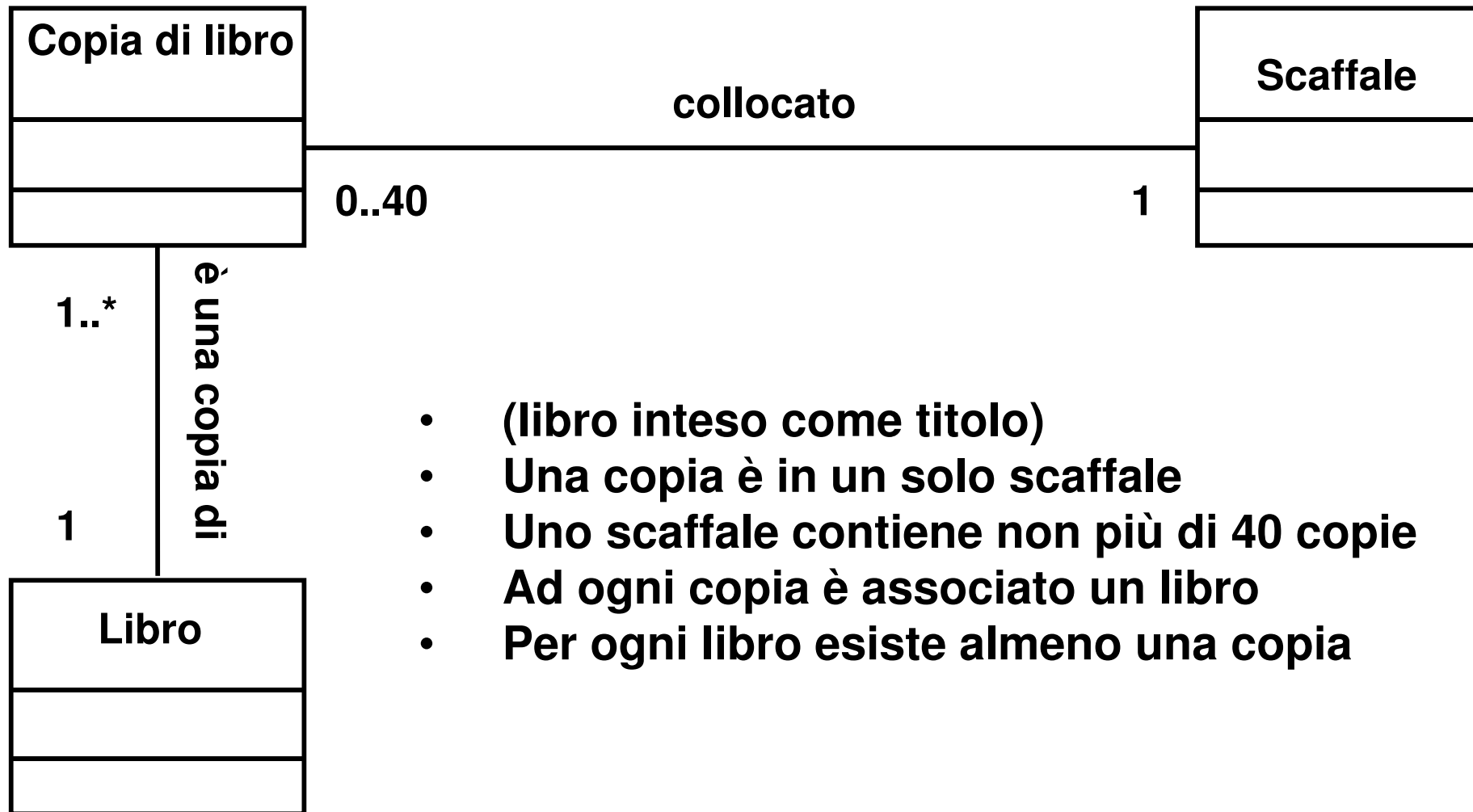
Attenzione al nome dell'associazione



Molteplicità: sintassi

- minimo..massimo
 - 0..2 nessun oggetto, un oggetto o 2 oggetti
 - 1..* almeno un oggetto
 - * equivalente a 0..*
 - 1 un solo oggetto
- ARLOW: cancellare ultima riga tabella 9.1 (UML1)

Molteplicità: esempio

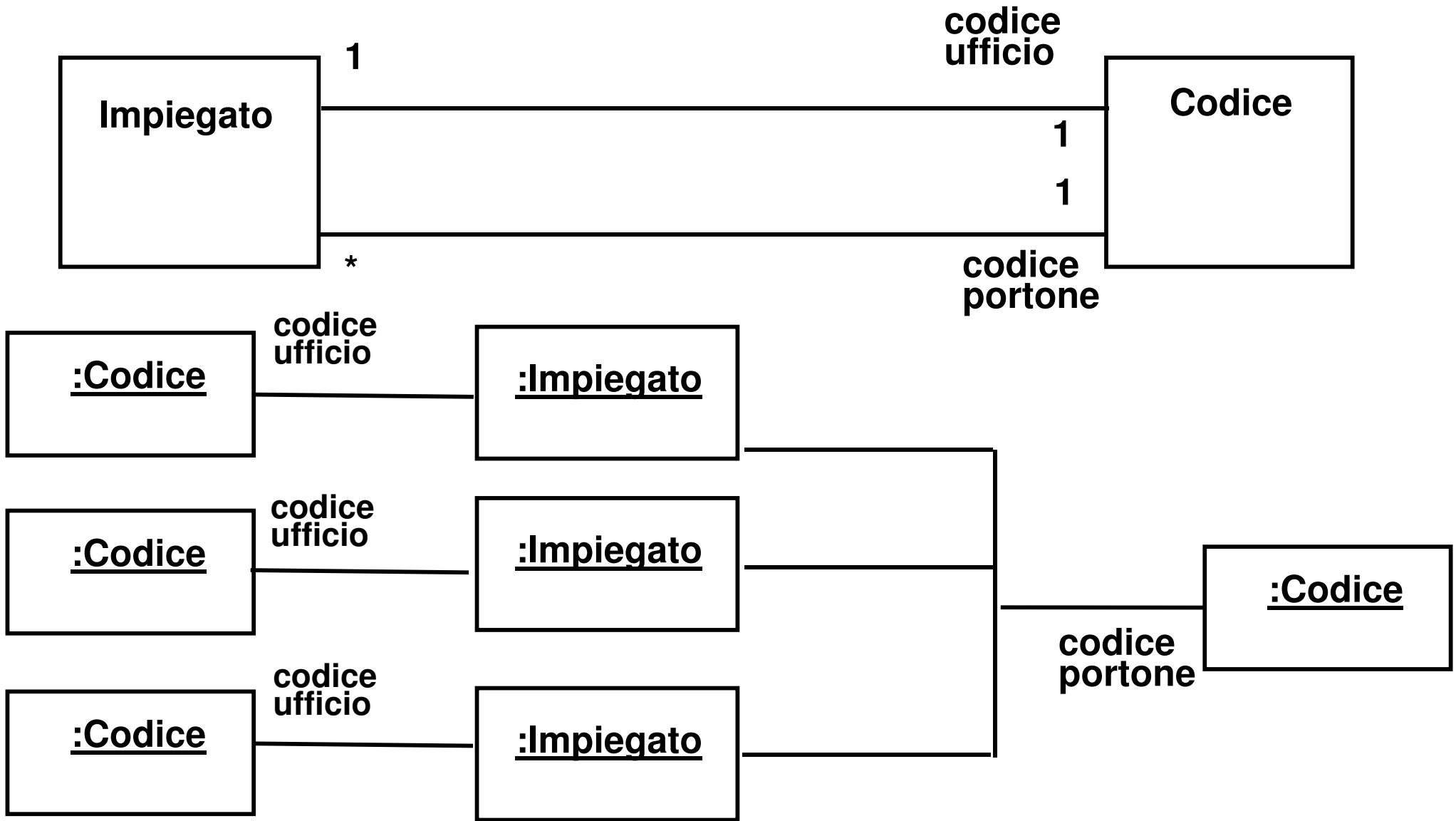


- (libro inteso come titolo)
- Una copia è in un solo scaffale
- Uno scaffale contiene non più di 40 copie
- Ad ogni copia è associato un libro
- Per ogni libro esiste almeno una copia

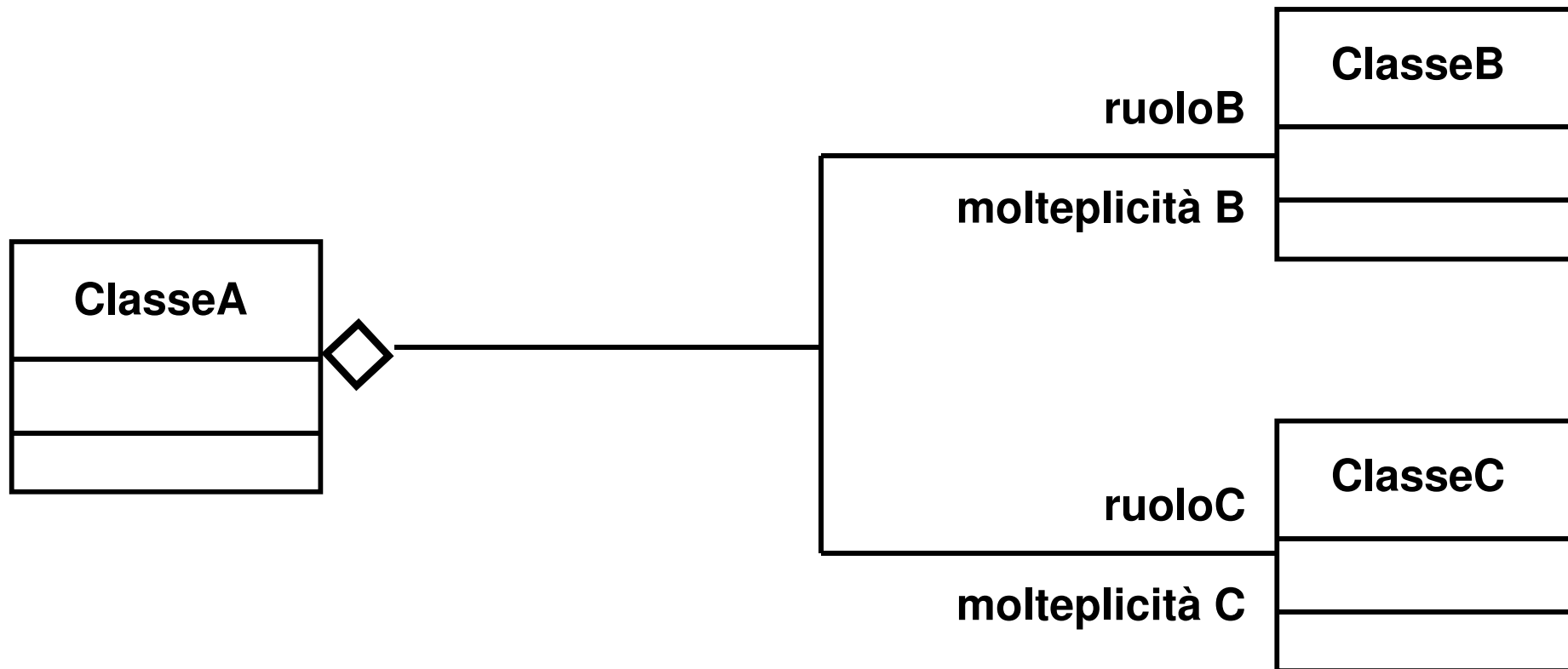
Sulla molteplicità, finezze.

In an n -ary association, the multiplicity is defined with respect to the other $n-1$ ends. For example, given a ternary association among classes (A, B, C), then the multiplicity of the C end states how many C objects may appear in association with a particular pair of A and B objects. If the multiplicity of this association is (many, many, one), then for each possible (A, B) pair, there is a unique value of C. For a given (B, C) pair, there may be many A values, however, and many values of A, B, and C may participate in the association.

Molteplicità: esempio

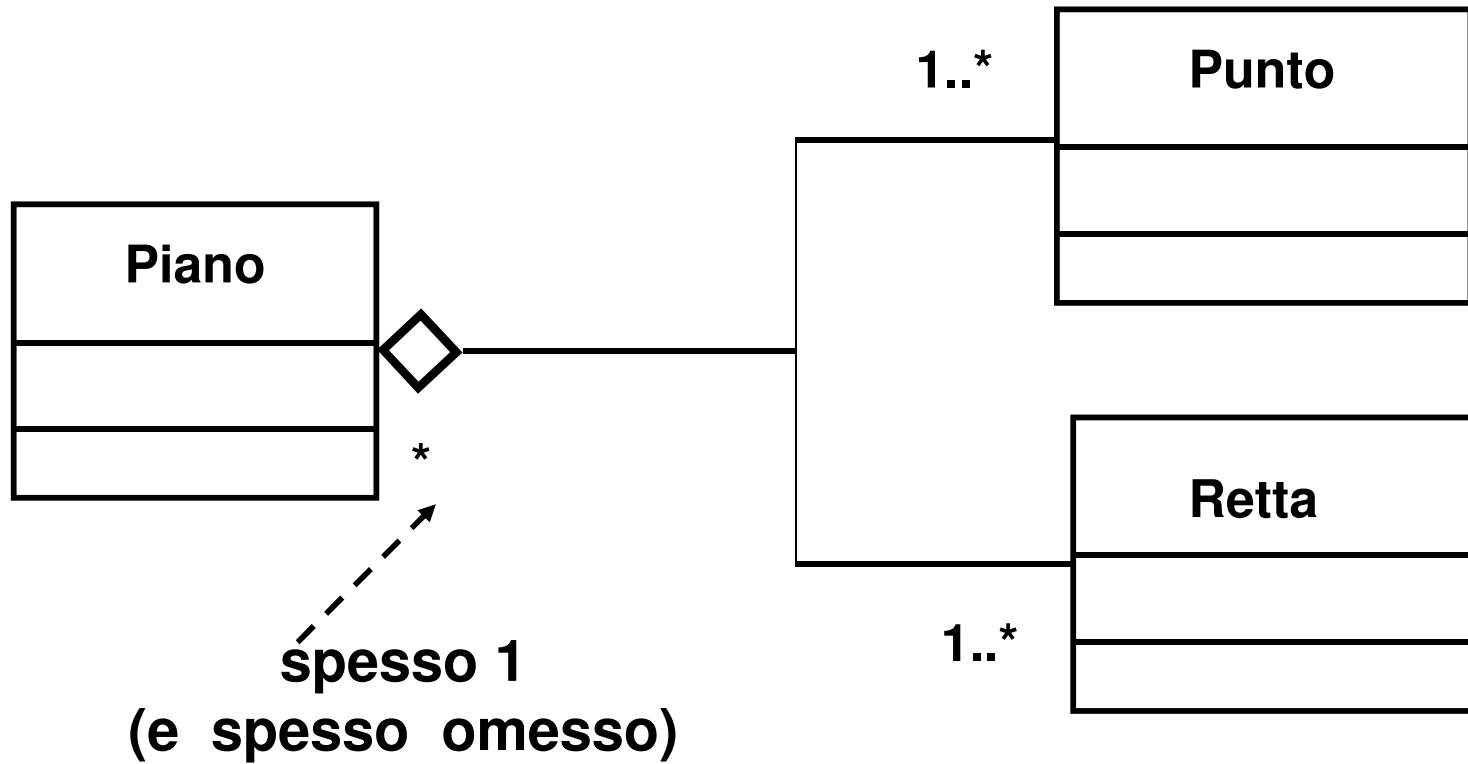


Sintassi dell'aggregazione

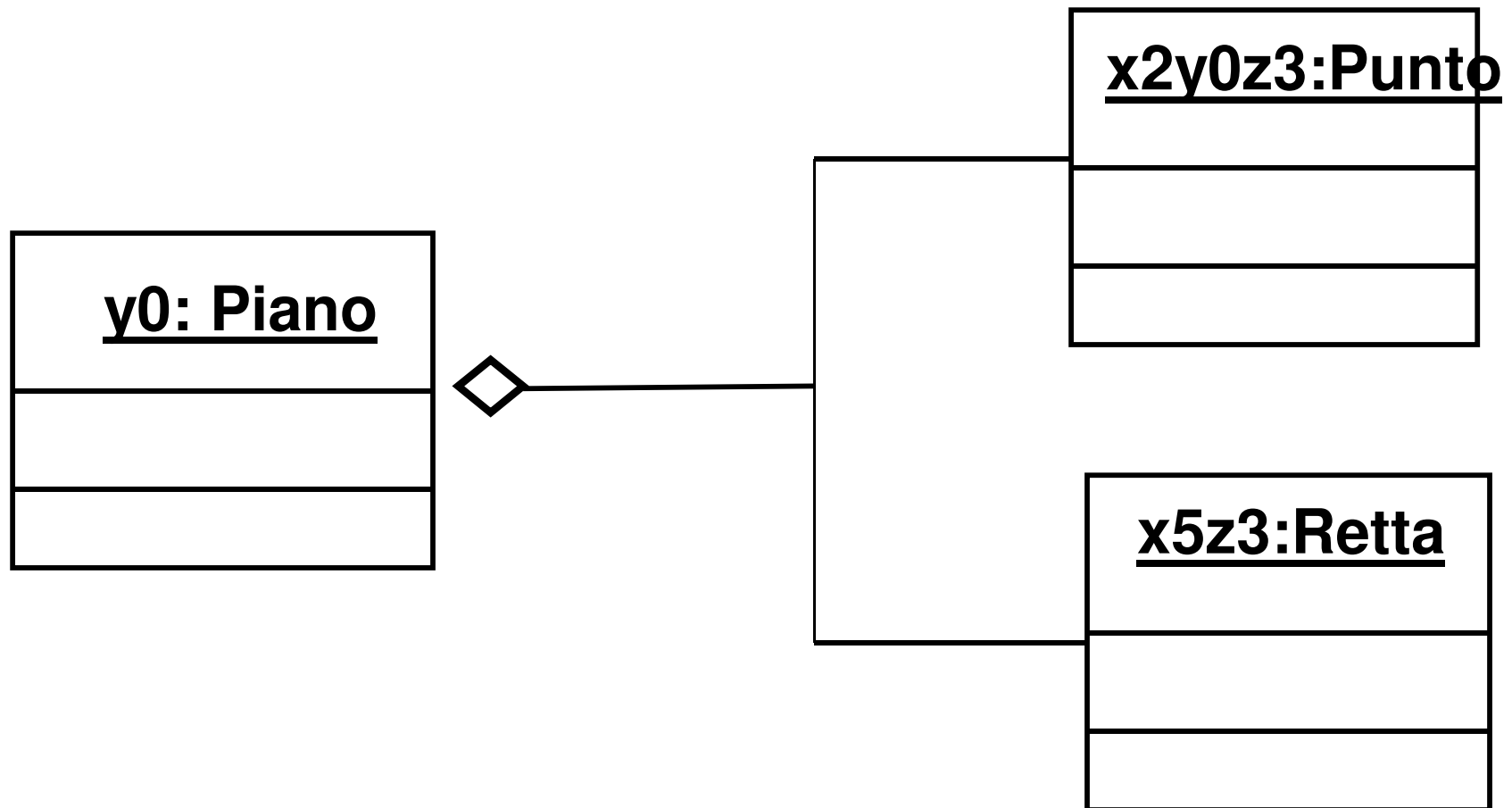


Rappresenta una relazione parte (B e C) – tutto (A)

Esempio di aggregazione

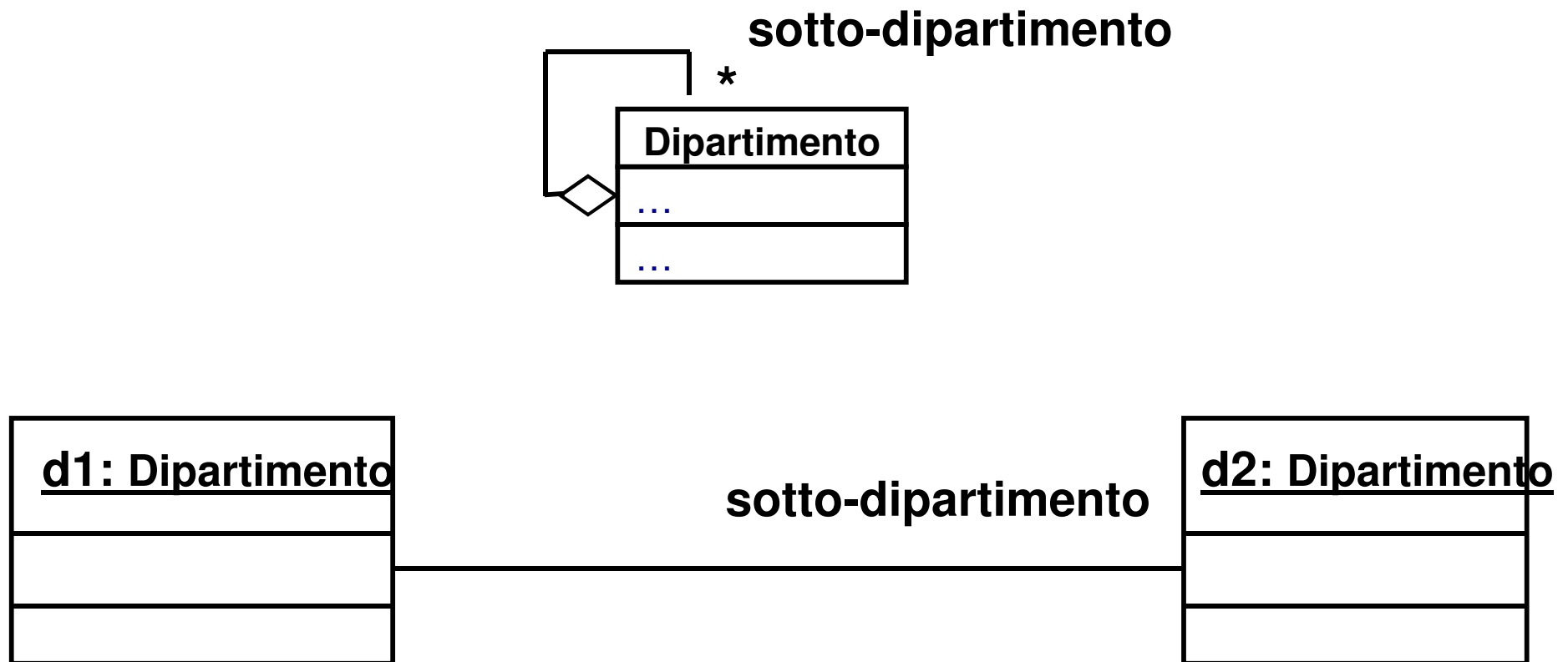


Aggregazione a livello di istanza



Ruoli e aggregazione: un caso speciale

- Nel diagramma delle classi

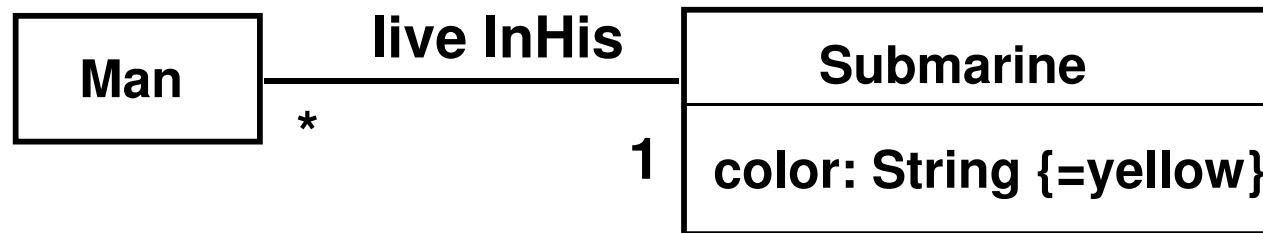


Esempio

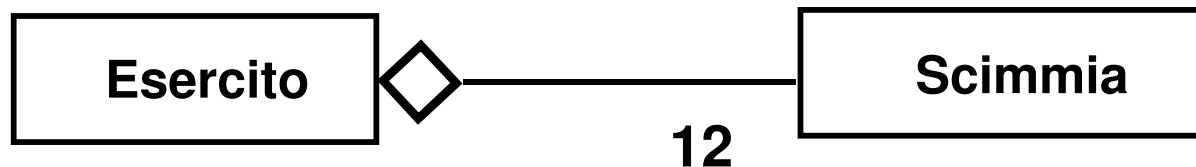
- Descrivere un albero, formato da radice e sottoalberi

Esempi

- We all live in our yellow submarine

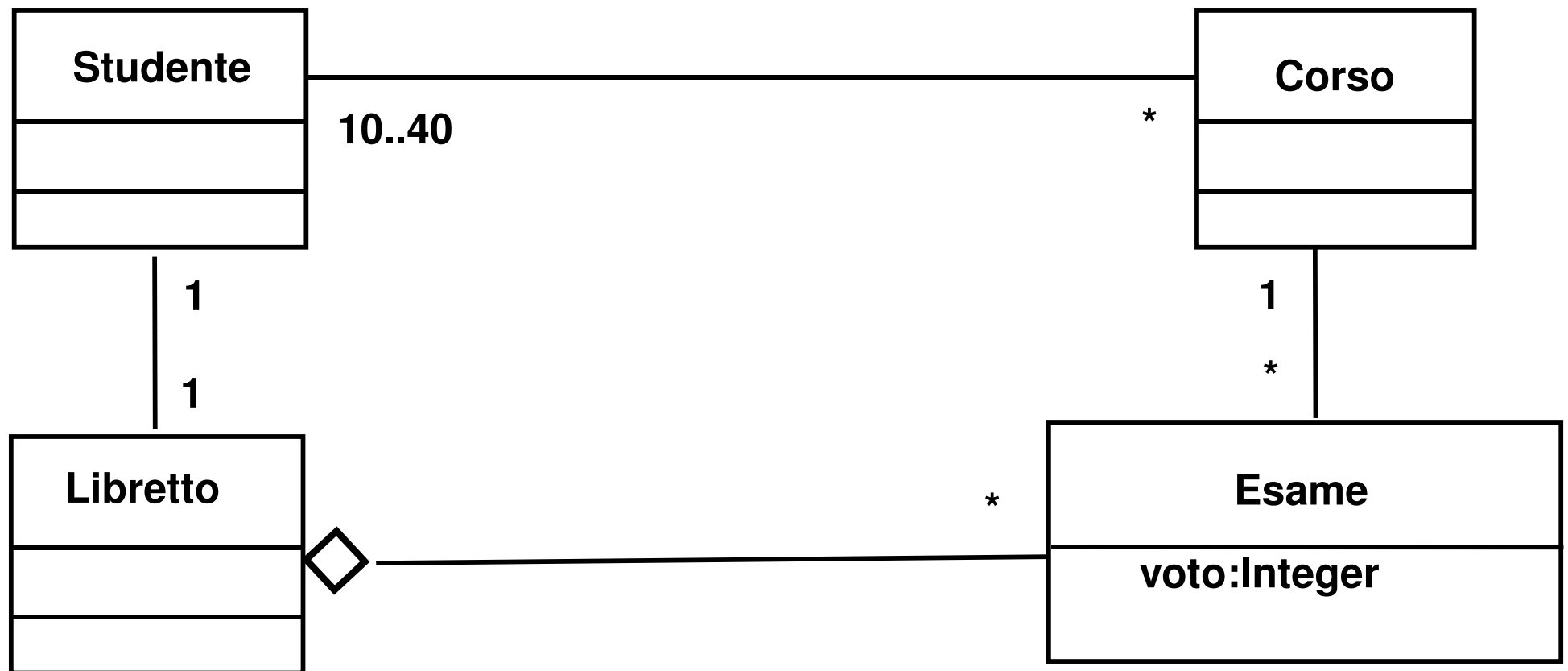


- L'esercito delle 12 scimmie

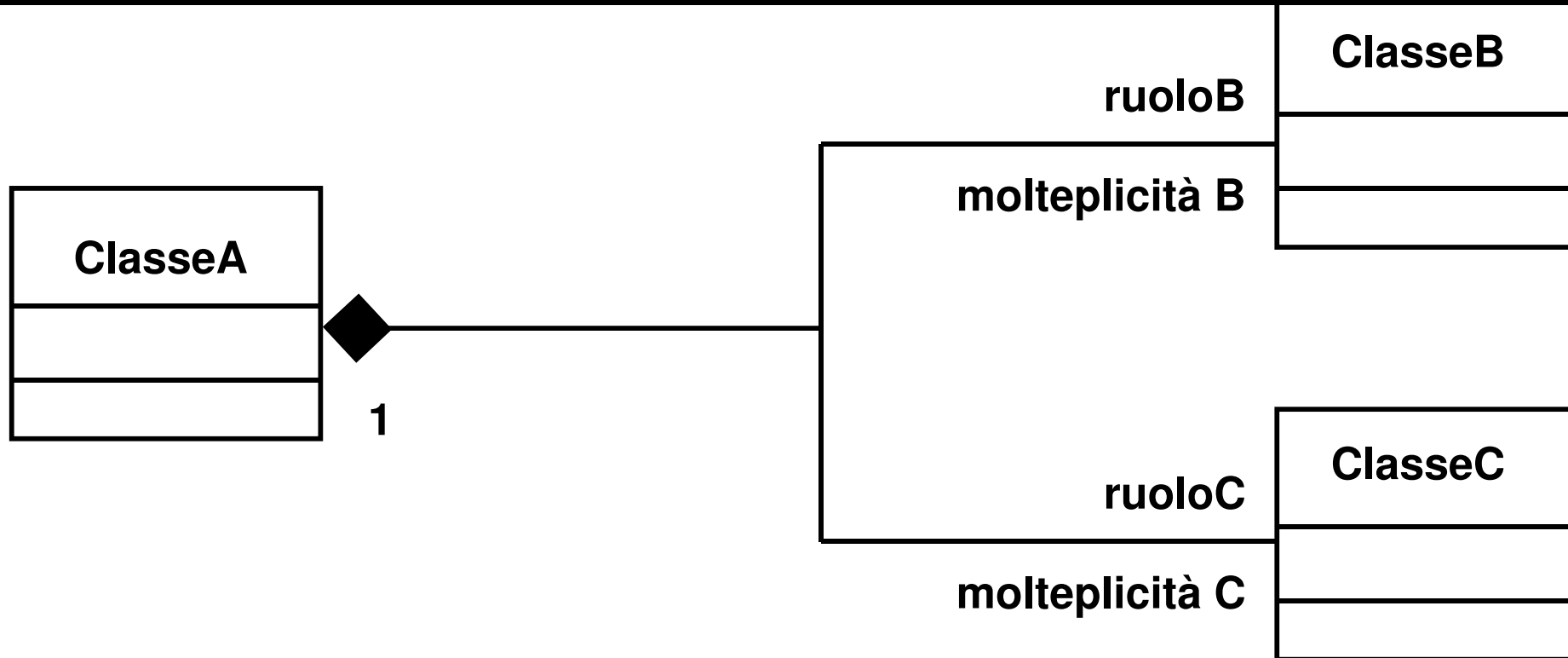


Esempio

- Studente, Libretto, Esame, Corso



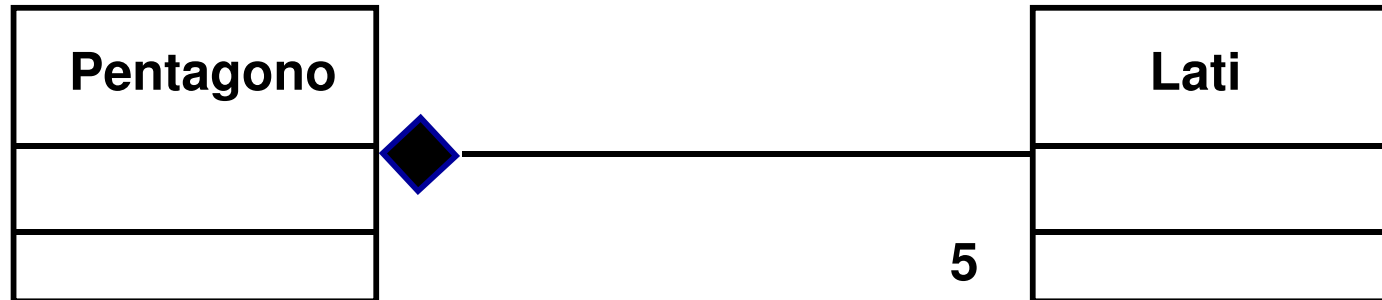
Sintassi della composizione



Una composizione è una forma di associazione più forte dell'aggregazione

le parti non hanno senso senza il tutto

Esempio di composizione



**Se cancello un poligono cancello anche i suoi lati
(mentre se cancello un piano, i punti restano in quanto
appartengono anche a altri piani)**

Prospettiva sw: il tutto crea e distrugge le parti

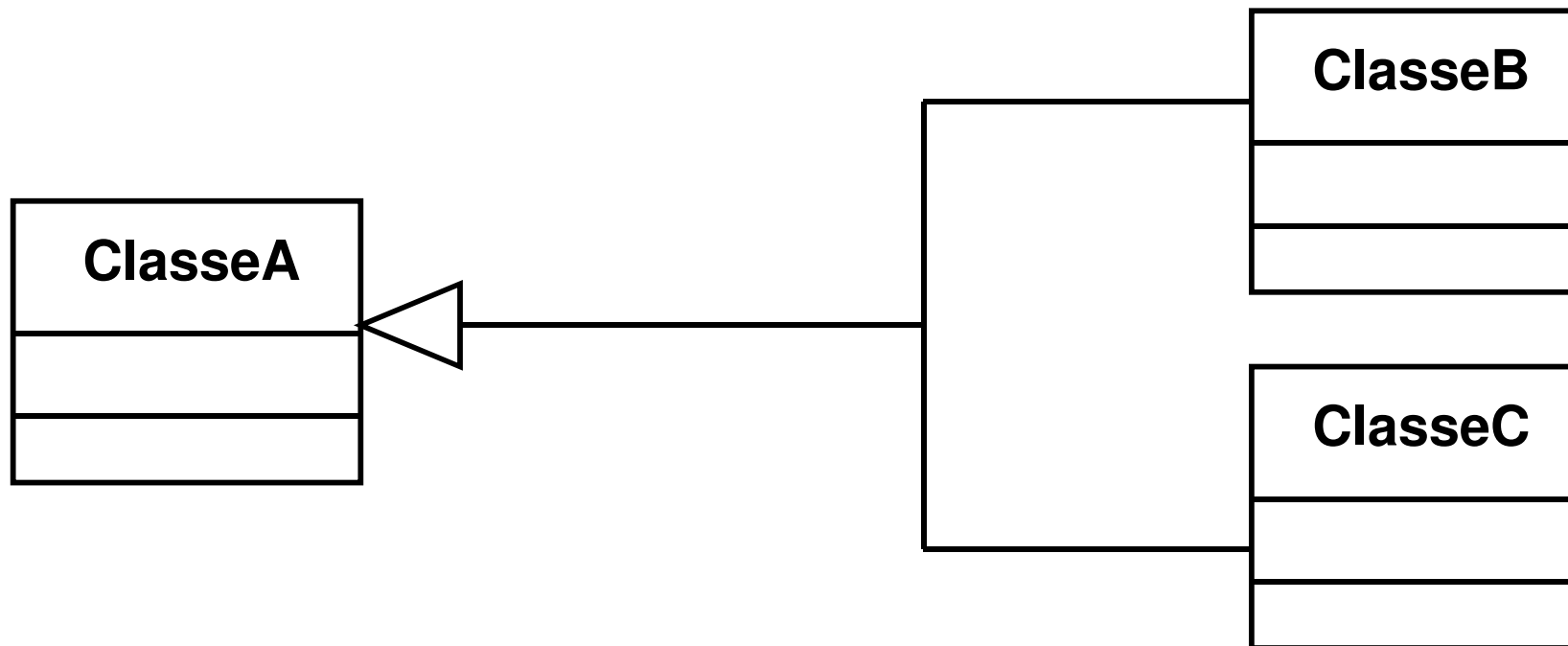
Attributo vs composizione

- semanticamente equivalenti
- mai entrambi

Note that an attribute is semantically equivalent to a composition association. (There may be some doubt about this. The specification includes a notation for composition of attributes. The use of attributes in the metamodel included in the specification, however, indicates that this is not meant to be taken seriously.) However, the intent and usage of attributes and associations are usually different. Use attributes for data types—that is, for values with no identity. Use associations for classes—that is, for values with identity. The reason is that for objects with identity, it is important to see the relationship in both directions; for data types, the data type is usually subordinate to the object and has no knowledge of it.

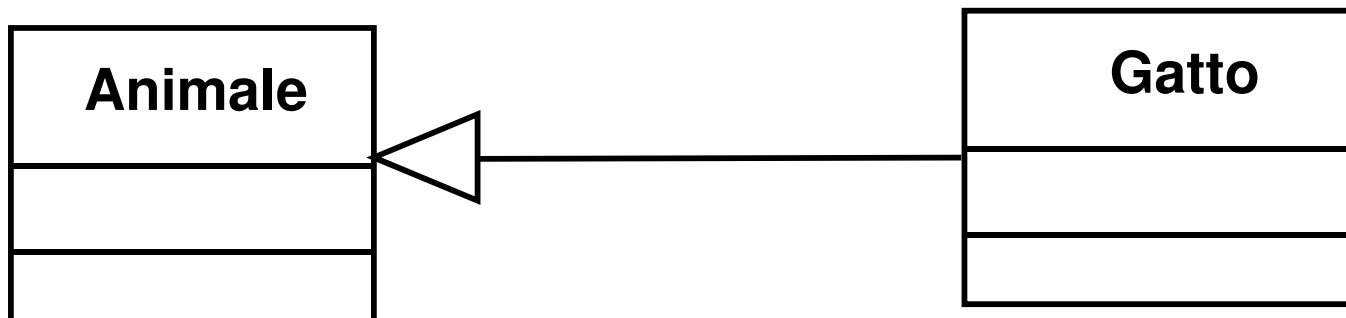
Sintassi della generalizzazione

- Generalizzazione/specializzazione, super/sotto classe
- Relazione : is-a

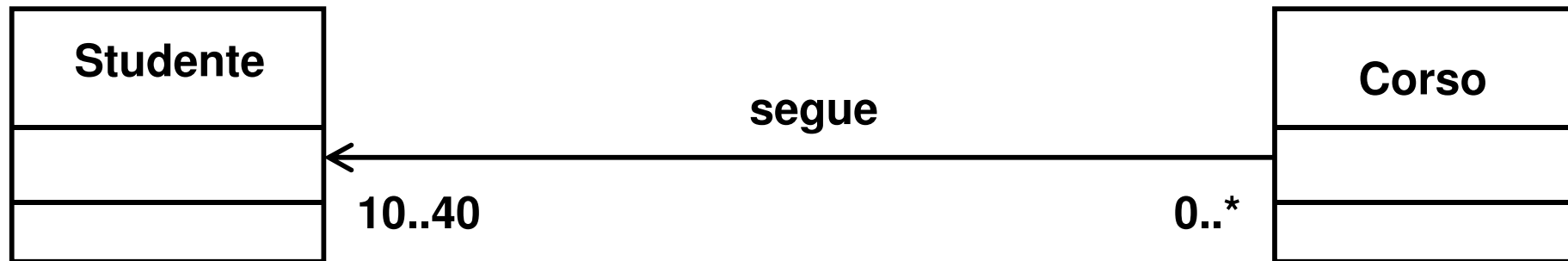


Esempio di generalizzazione

- (tutti) i gatti sono animali
- Minou è sia un Gatto che un Animale

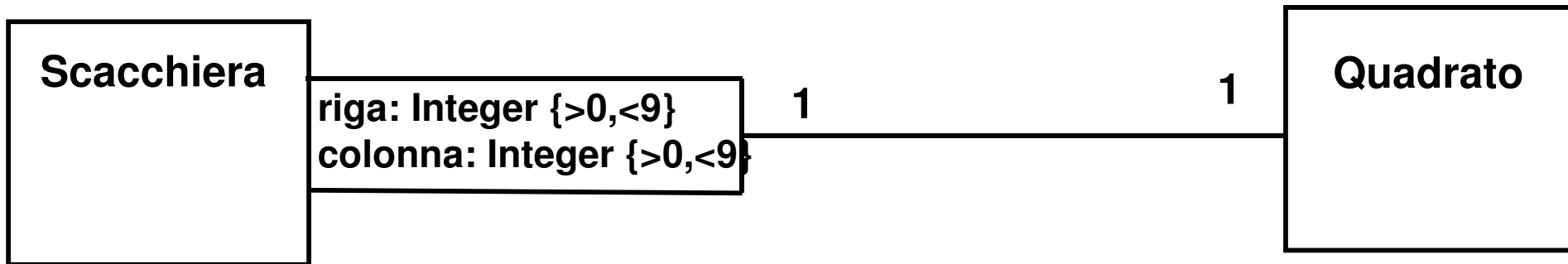


Associazione: navigabilità



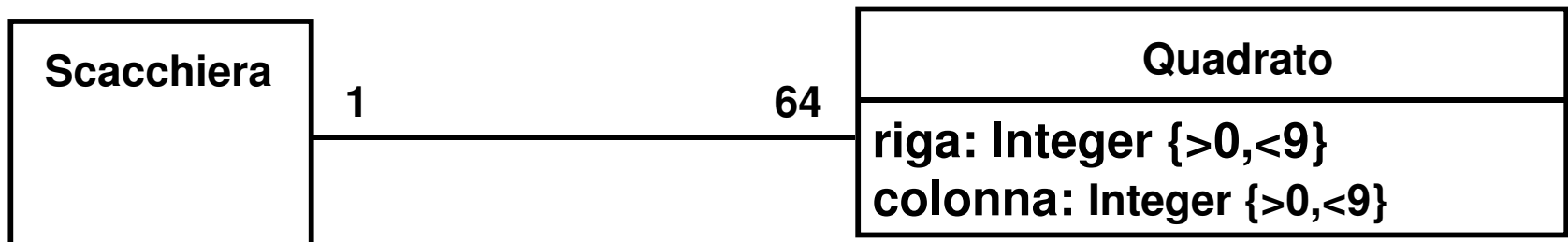
- **Ci si può spostare da un corso agli studenti: corso “conosce” lo studente.**
- **Posso riusare la classe studente senza la classe corso, ma non viceversa.**
- **La navigabilità è indipendente dal verso di lettura del nome dell’associazione** ►

Associazioni qualificate

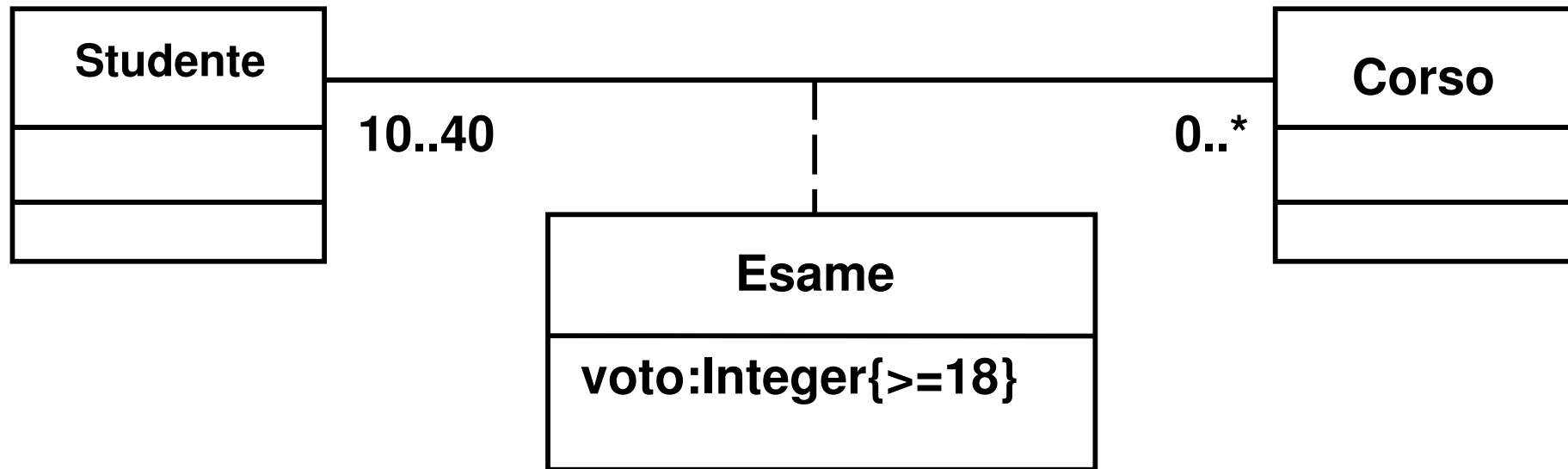


**Un qualificatore è un attributo di un'associazione:
individua un oggetto tra quelli collegati
dall'associazione**

Per meglio definire

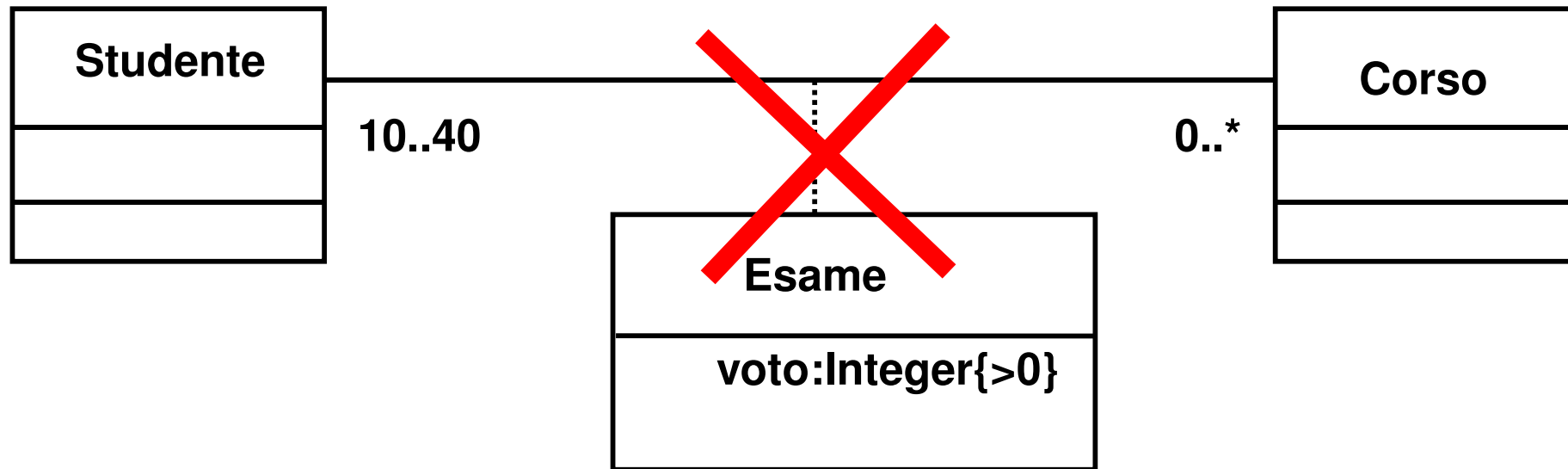


Classi associazione



- **Un'associazione può avere attributi propri, rappresentati con una classe associazione.**
- **Le istanze sono collegamenti con attributi propri.**
- **Voto non è attributo né di Corso né di Studente**

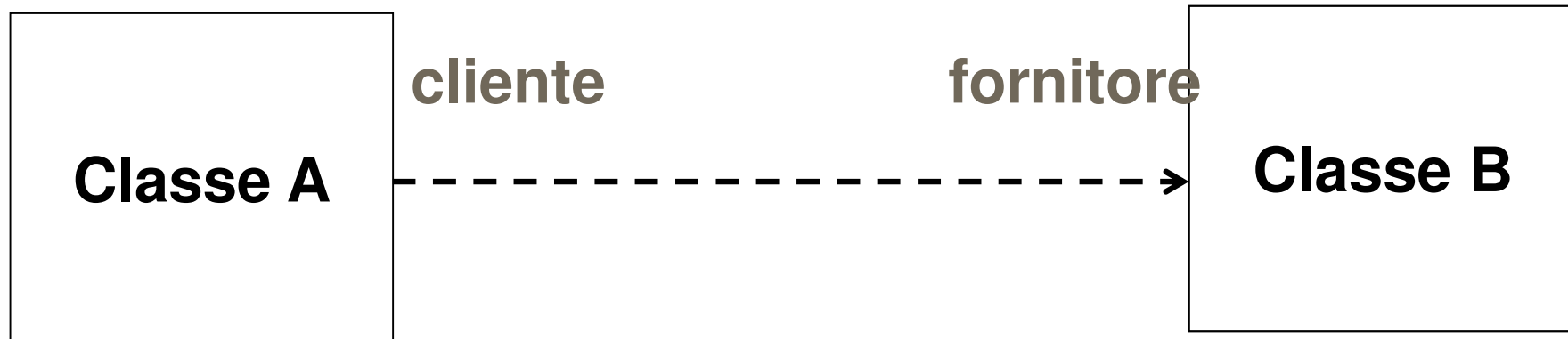
Classi associazione (2)



- **Supponiamo di tener traccia dei voti negativi**
- **Per ogni coppia di oggetti collegati tra loro può esistere un unico oggetto della classe associazione**
- **A meno di usare {bag}**

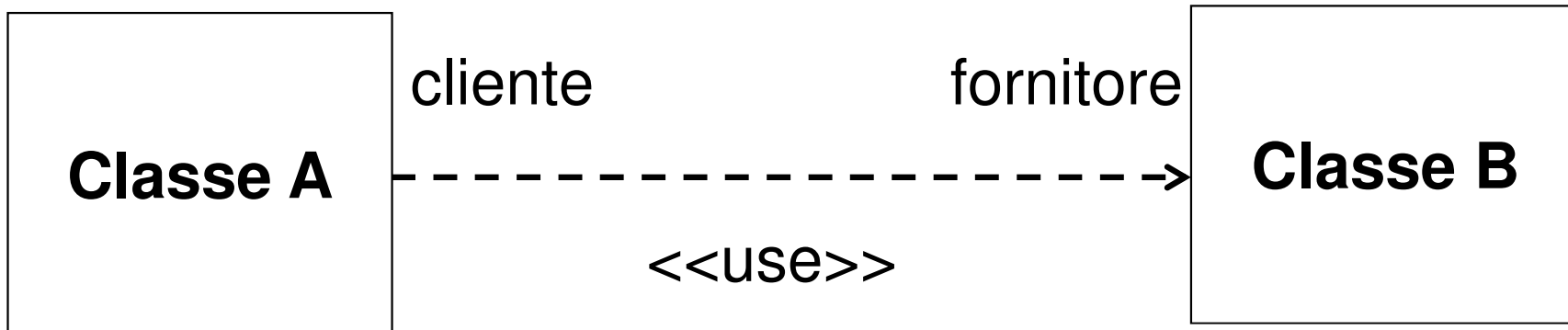
Dipendenze

- Una relazione in cui una modifica nel fornitore può influenzare il cliente
 - Il cliente dipende dal fornitore

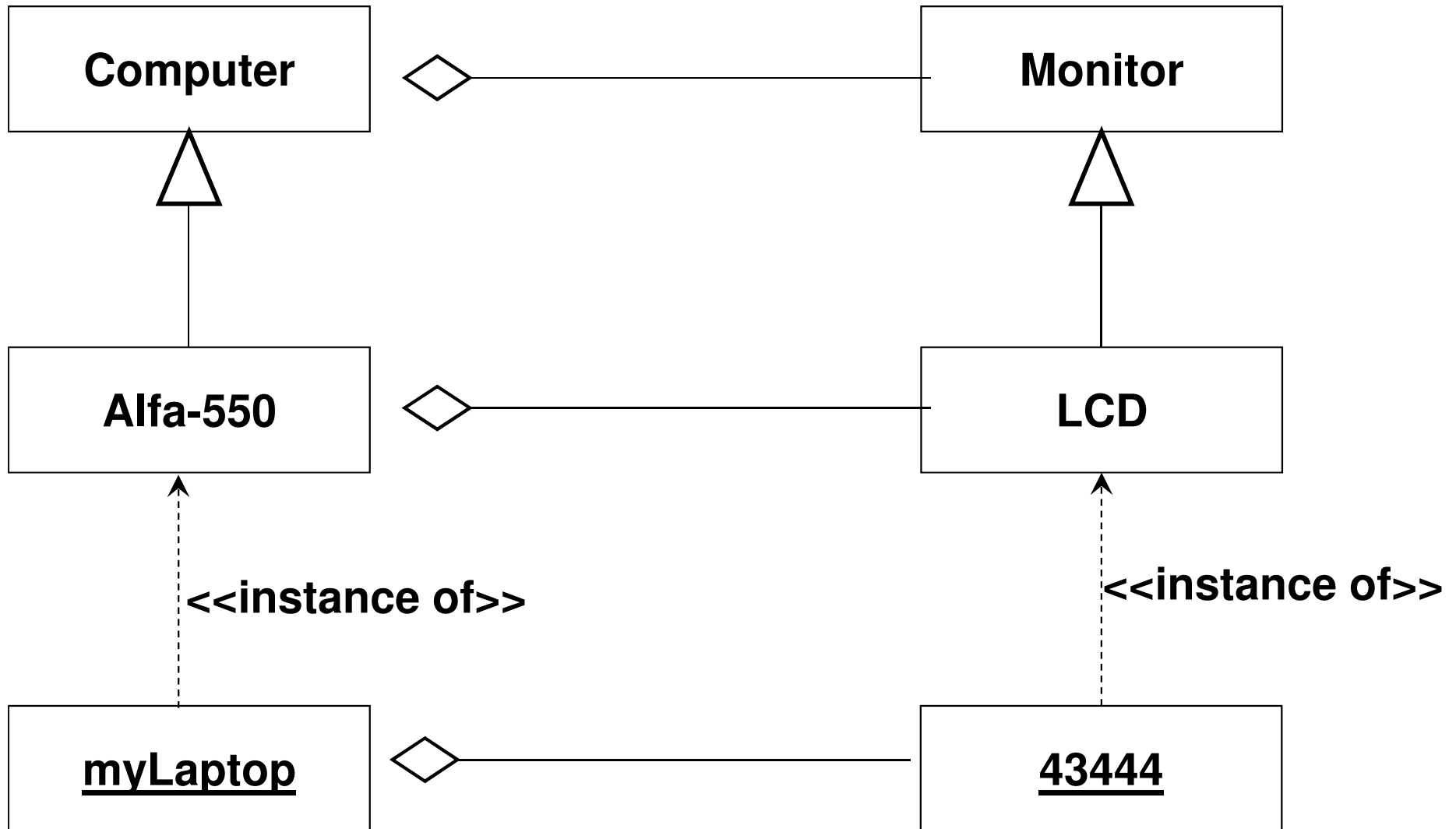


Dipendenza d'uso: esempi

- Un parametro di un'operazione di A è di tipo B
- Un'operazione di A restituisce un oggetto di tipo B
- Un'operazione di A crea dinamicamente un oggetto di tipo B

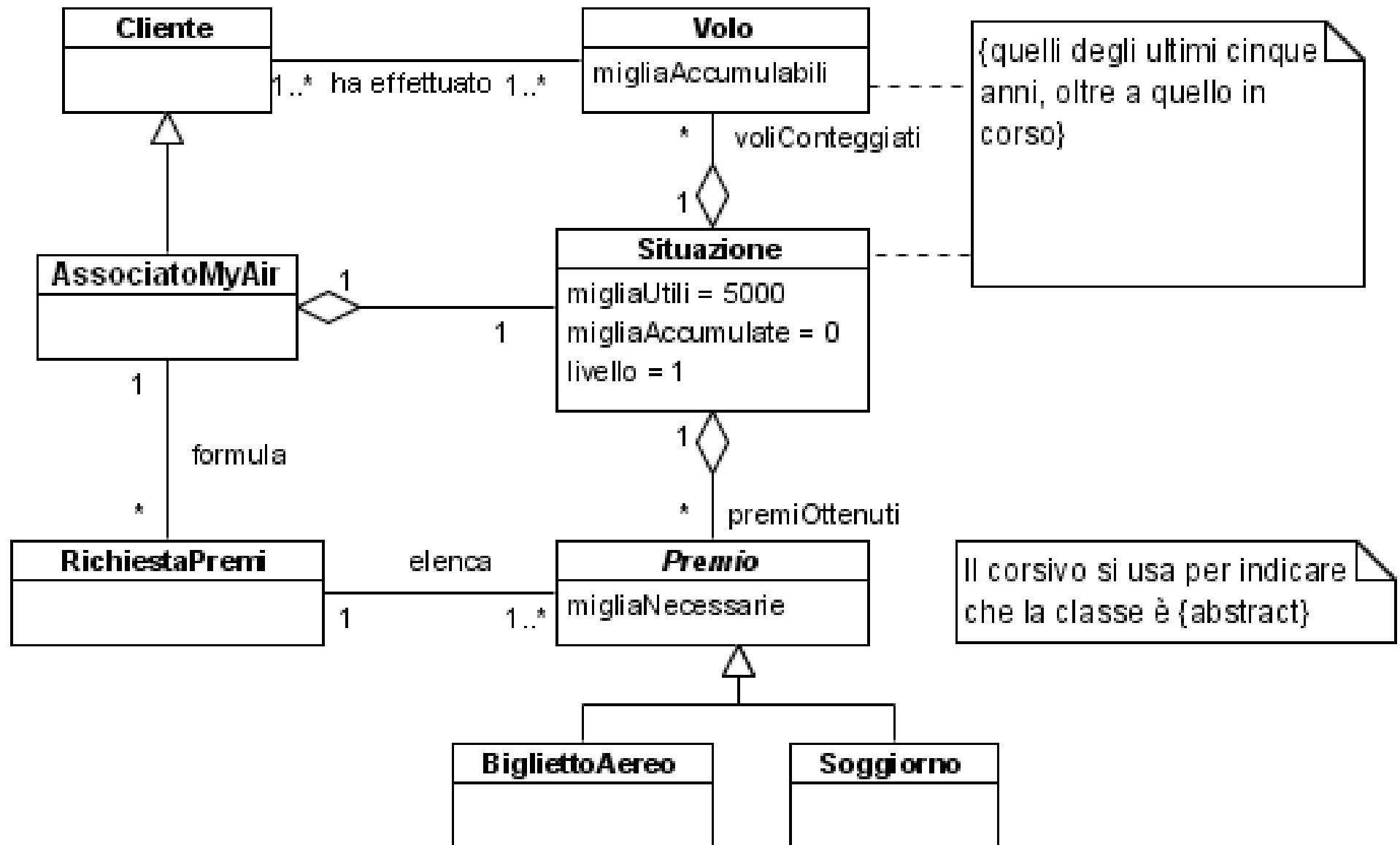


Un esempio completo



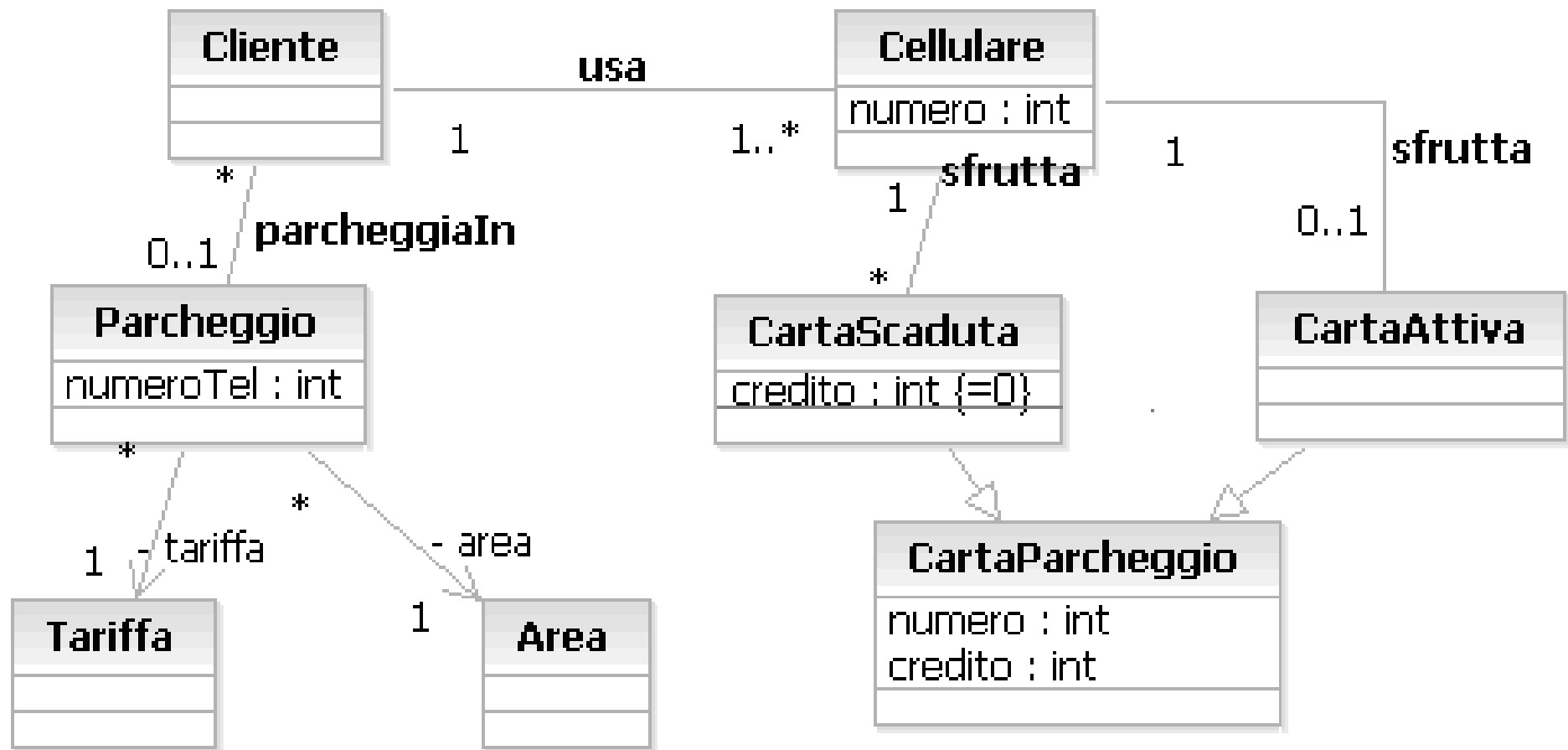
Ex: MyAir

- Iscriviti al programma e da semplice cliente diventerai un associato MyAir, guadagnando immediatamente un bonus di 5.000 miglia utili.
- Ogni volta che volerai con MyAir le miglia accumulabili del volo saranno sommate alle tue miglia utili, permettendoti di raggiungere in poco tempo le miglia necessarie per richiedere uno dei nostri premi (omaggio biglietti aereo o soggiorni in località da sogno).
- I premi riscossi danno luogo a una diminuzione immediata delle miglia utili. La situazione è aggiornata il 31 dicembre, mantenendo solo le miglia dei voli effettuati negli ultimi 5 anni.
- Inoltre se accumulerai almeno 15.000 miglia (miglia accumulate) sarai promosso dal livello standard al livello argento. Se invece accumulerai almeno 100.000 miglia entrerai a far parte del ristretto numero di associati del livello oro2.
- Tutte le condizioni si riferiscono esclusivamente alle miglia accumulate in un anno. Il passaggio da un livello all'altro è effettuato il 31 dicembre. La permanenza nel livello da un anno all'altro è soggetta al rispetto degli stessi requisiti per entrare nel livello. Il bonus iniziale non concorre al raggiungimento delle miglia richieste per cambiare o mantenere un livello.



Easy Park

- Soluzione per il pagamento del parcheggio via telefono cellulare.
- 1. Il cliente acquista una Carta Parcheggio prepagata e l'attiva indicando il proprio numero di cellulare. Durante l'attivazione, il sistema trasferisce sulla nuova carta l'eventuale credito residuo su una carta già associata al numero di telefono indicato.
- 2. Il cliente parcheggia ed espone sul cruscotto la Carta Parcheggio. Nel cartellone del Parcheggio verifica qual è il numero di telefono che identifica l'area e la tariffa. Il cliente telefona a questo numero, il cliente è identificato attraverso il proprio numero di telefono cellulare e il sistema attiva il pagamento della sosta.
- 3. Il Controllore controlla l'effettivo pagamento della sosta inserendo il numero della Carta Parcheggio in un applicativo fruibile tramite Pocket PC connesso a internet o Telefono Cellulare.
- 4. Disattivazione della sosta con chiamata via cellulare: l'utente chiama il numero associato al parcheggio, il sistema riconosce l'utente e disattiva il pagamento. Inoltre il sistema comunica vis SMS la disattivazione, la somma pagata, la durata della sosta e il residuo presente sulla Carta Parcheggio.



**Modellare anche sosta, con inizio, fine costo
(classe associazione)**

Enumerazioni

Table 4-1: *Kinds of Classifiers (continued)*

<i>Classifier</i>	<i>Function</i>	<i>Notation</i>
enumeration	A data type with predefined literal values	<code>«enumeration» Name</code>

User models may also declare **enumeration** types. An enumeration type declares a set of **enumeration literals** that may be used as values.

Enumeration types are user-definable finite sets of named elements that have a defined ordering among themselves but no other computational properties. An enumeration type has a name and a list of enumeration constants. The enumeration type `Boolean` is predefined with the enumeration literals `false` and `true`.

enumeration

A **data type** whose **instances** form a list of named literal values. Usually, both the enumeration name and its literal values are declared.

Syllabus

- Arlow. Capitolo 7 e 8, tranne 8.2 e 8.4.3. Cap 9