06. Classi e oggetti in UML

IS 2024-2025

Laura Semini, Jacopo Soldani
Corso di Laurea in Informatica
Dipartimento di Informatica, Università of Pisa

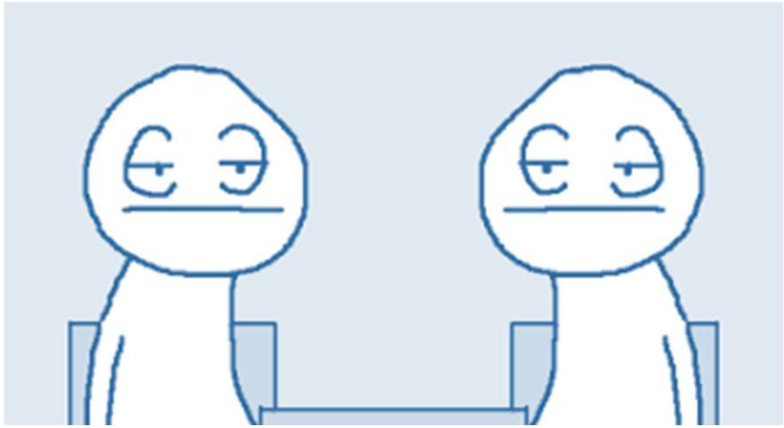
CLASSI E OGGETTI (REMINDER)

Un **oggetto** è un'entità caratterizzata da

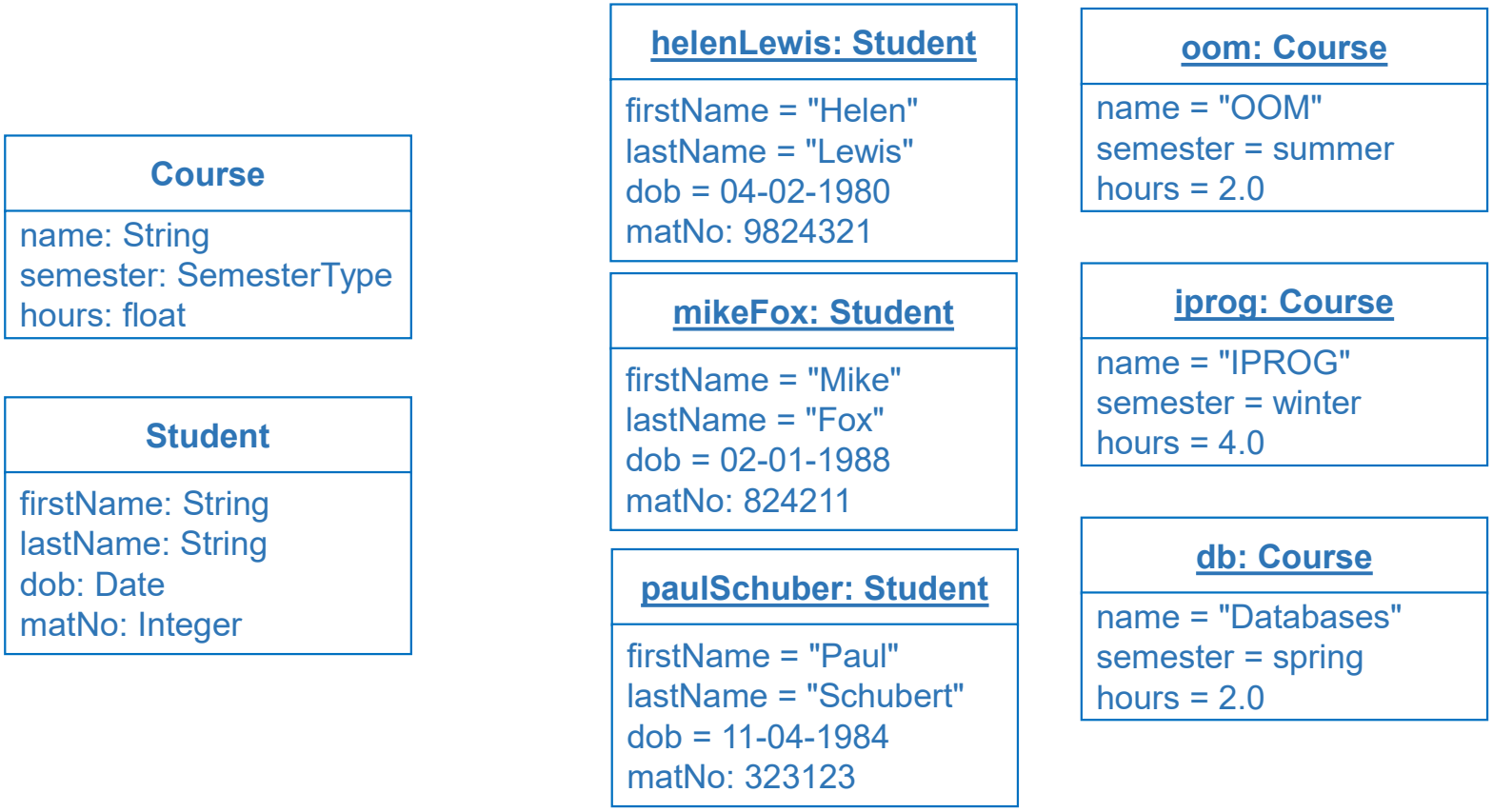
- identità,
- stato, e // (I valori de)gli attributi definiscono lo stato dell'oggetto
- comportamento // Le operazioni definiscono il suo comportamento

Una **classe** descrive

- un insieme di oggetti con caratteristiche simili
- cioè oggetti che hanno lo stesso tipo



CLASSI E OGGETTI, IN UML



classi **oggetti/istanze**

DIAGRAMMA DELLE CLASSI

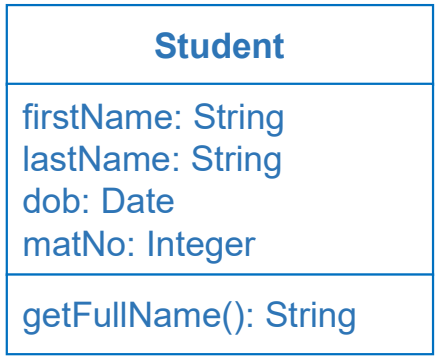
Una classe cattura un **concetto** nel dominio del problema o della realizzazione

Il diagramma delle classi descrive

- Il **tipo** degli oggetti che fanno parte di un sistema software o del suo dominio
- Le **relazioni** statiche tra essi

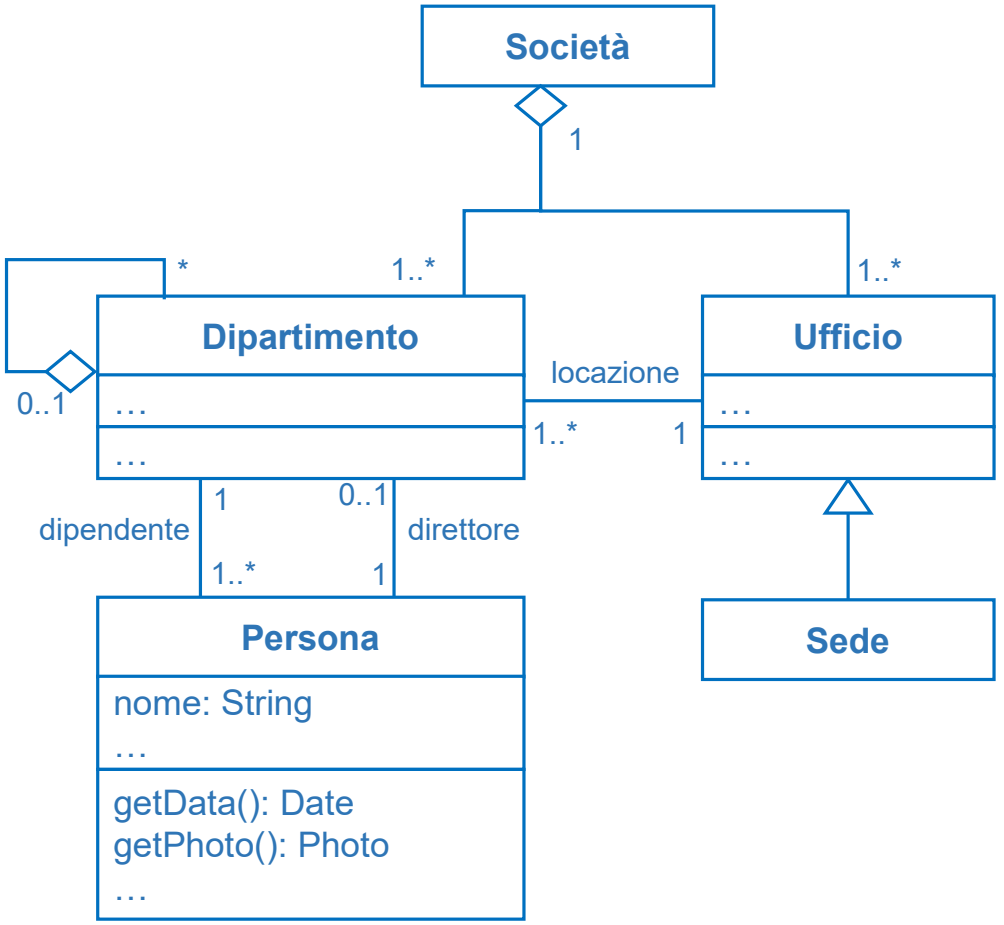
NB: Elementi e relazioni tra elementi **non cambiano** nel tempo

I diagrammi delle classi mostrano anche le **proprietà** e le **operazioni** di una classe



ESEMPIO

- 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



CLASSI, IN UML

Una classe è rappresentata con un rettangolo organizzato in tre sottosezioni (o *compartment*)

- **nome** (maiuscolo e sempre al singolare) // Libro
- **attributi** (tipizzati) // codice e titolo
- **operazioni** (tipizzate) // cambiaCodice e getTitolo

Attributi e operazioni sono specificati con **modificatori di visibilità**

- **+** → pubblico
- **–** → privato

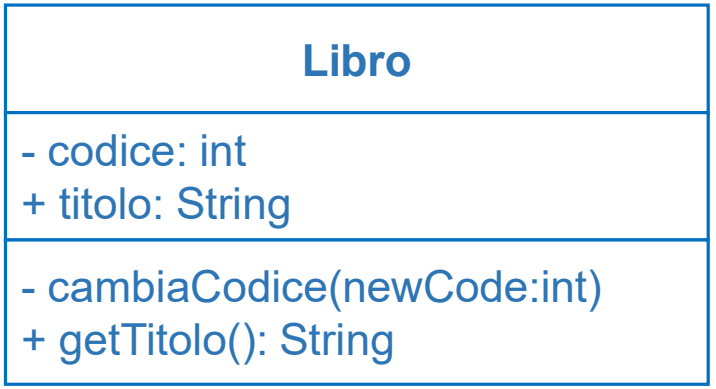


DIAGRAMMA DELLE CLASSI: USI

Il diagramma delle classi può essere usato per

descrivere il **dominio**

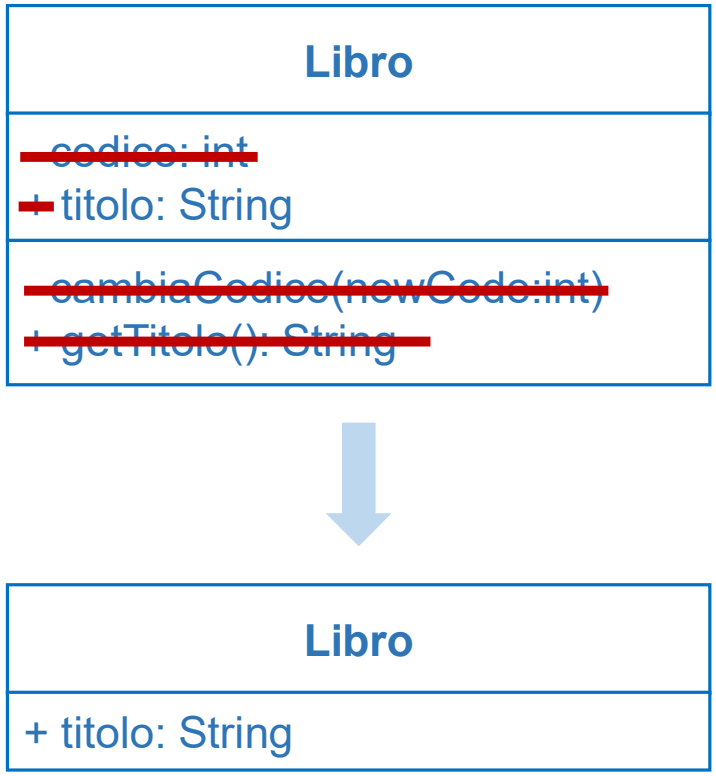
progettazione di dettaglio

LIVELLO DI ASTRAZIONE

 vs 

Quando si usa il diagramma delle classi per **descrivere il dominio**

- Solo attributi utili a caratterizzare l'elemento del dominio
  - nessun dettaglio implementativo
  - e.g., mai ID
- Le **operazioni** tipicamente si **omettono**
  - eccessivo livello di dettaglio
  - e.g., mai setter/getter
- I modificatori di **visibilità** si **omettono**



ATTRIBUTI: SINTASSI

chiamato di default nel libro,
ma si intende **iniziale**

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

obbligatorio
(gli altri opzionali)

per **array** di valori

```
colore: Integer [3] {ordered}        // modello RGB

secondoNome: String [0..1]           // zero per permettere valore null

nome: String                         // la molteplicità [1] può essere omessa
```

vincoli su valori ammissibili

```
{>0, <10}  // numero in (0,10)

{ordered}  // liste ordinate invece che insiemi o multi-insiemi

{unique}   // senza ripetizioni, come negli insiemi
```

ordered e **unique** sensati **solo** per attributi con **molteplicità** di valori

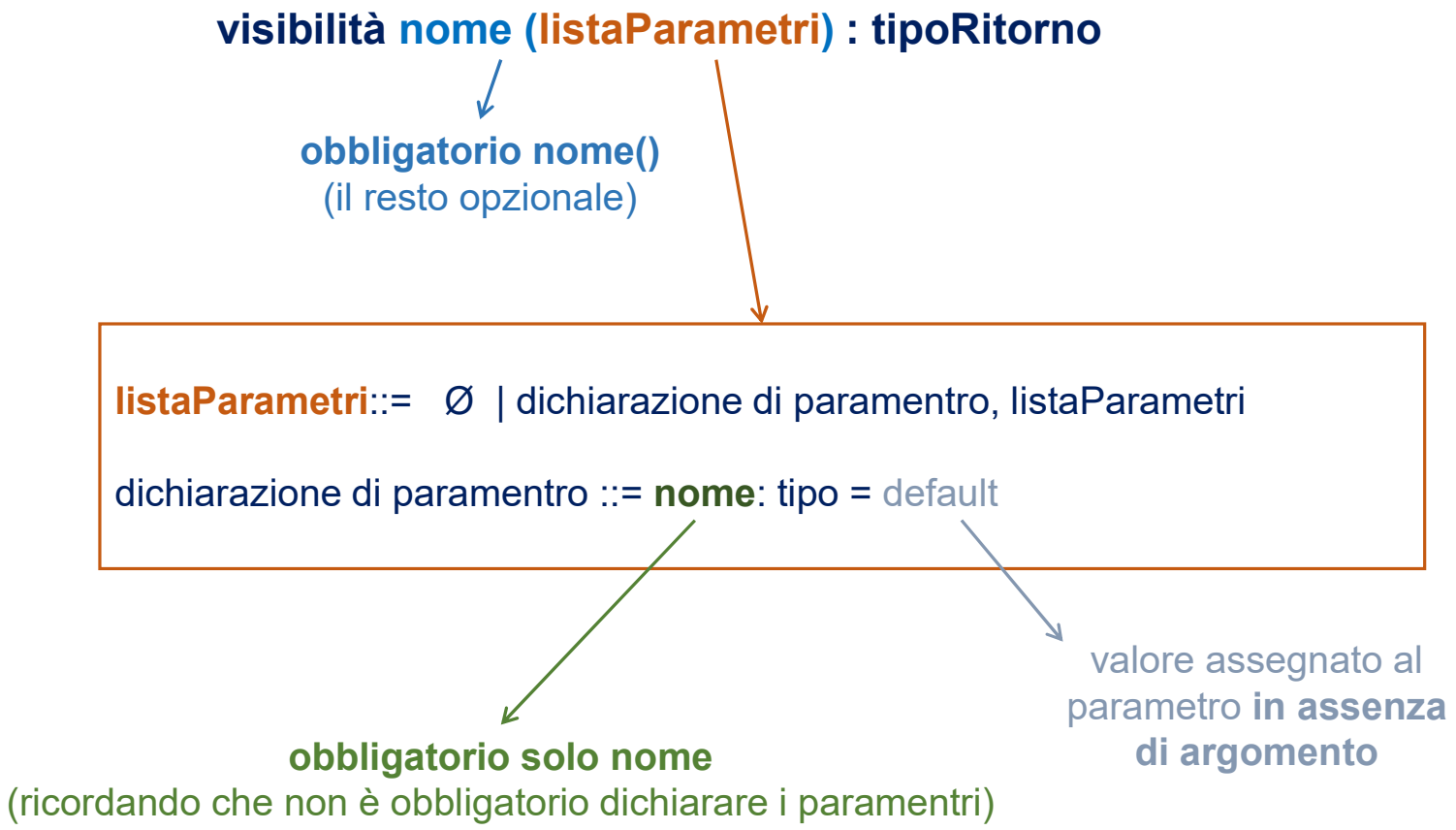
ESEMPI

- n: char // carattere, tipo predefinito
- n: String // stringa, tipo predefinito
- g: Gra // con tipo Gra definito nel modello
- n: Integer =1 {>= 0} // numero intero non negativo, inizialmente = 1
- p : Integer [2] {>0, ordered} // punto del quadrante positivo
- nome: String [1..2] {ordered, unique} // almeno un nome, opzionalmente un secondo nome, ma diverso dal primo

MODIFICATORI DI VISIBILITÀ

- **+** (**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

OPERAZIONI: SINTASSI



ESEMPI

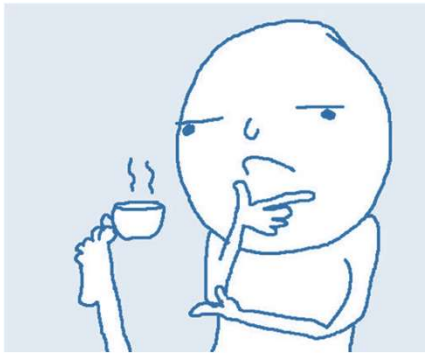
- + sum(a: Integer, b: Integer = 10): Integer

  // metodo pubblico che, dati due interi restituisce un intero
  // 10 valore di default del secondo parametro

- - gra(): Gra

  // metodo privato che restituisce un oggetto di tipo Gra

CLASSI O ATTRIBUTI?

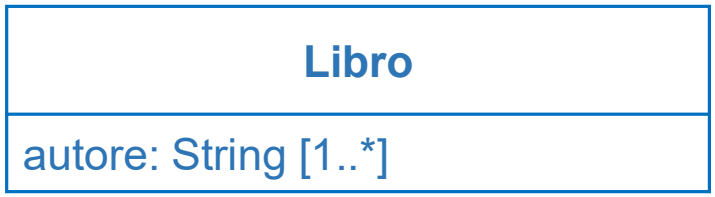


FAQ: Dato un concetto, meglio modellarlo come attributo o come classe?

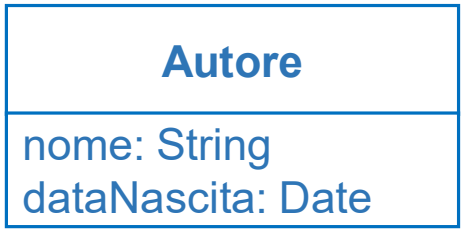
A: **Dipende** ☺

Esempio: Un libro ha uno o più autori

- autore come **attributo** di Libro: possiamo specificare solo il nome

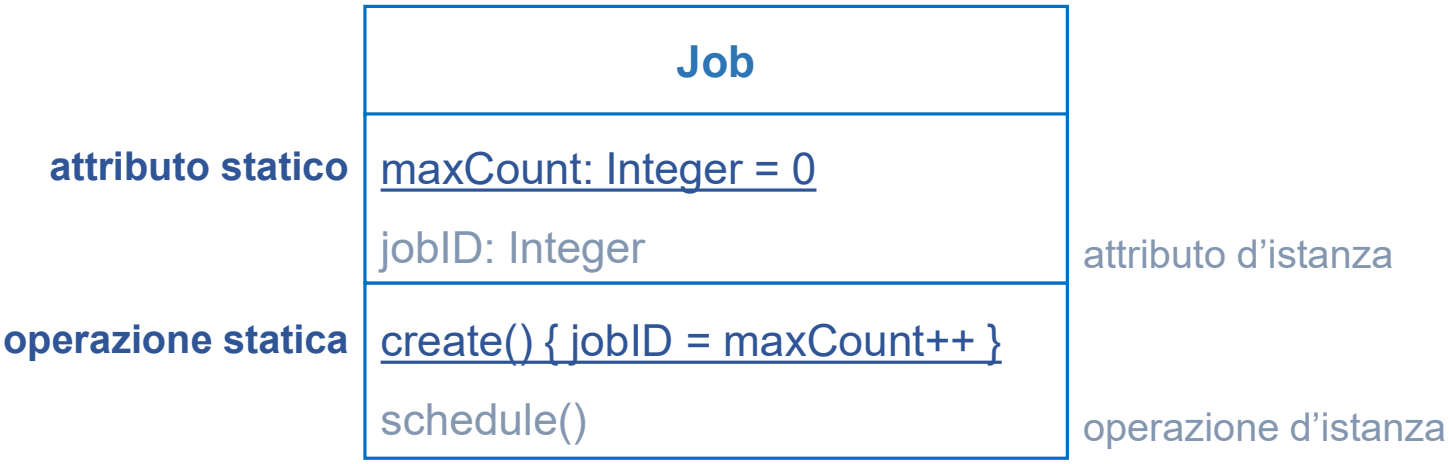


- autore come **classe**: può avere attributi propri e eventualmente operazioni



CAMPI STATICI

Attributi e operazioni con ambito di classe (**statici**) sono **sottolineati**



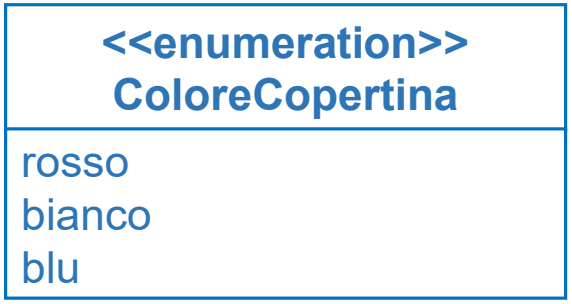
ENUMERAZIONI

Le **enumerazioni** sono usate per specificare un **insieme di valori prefissati**

- **lista completa** di tutti i valori che gli attributi di un determinato tipo possono assumere

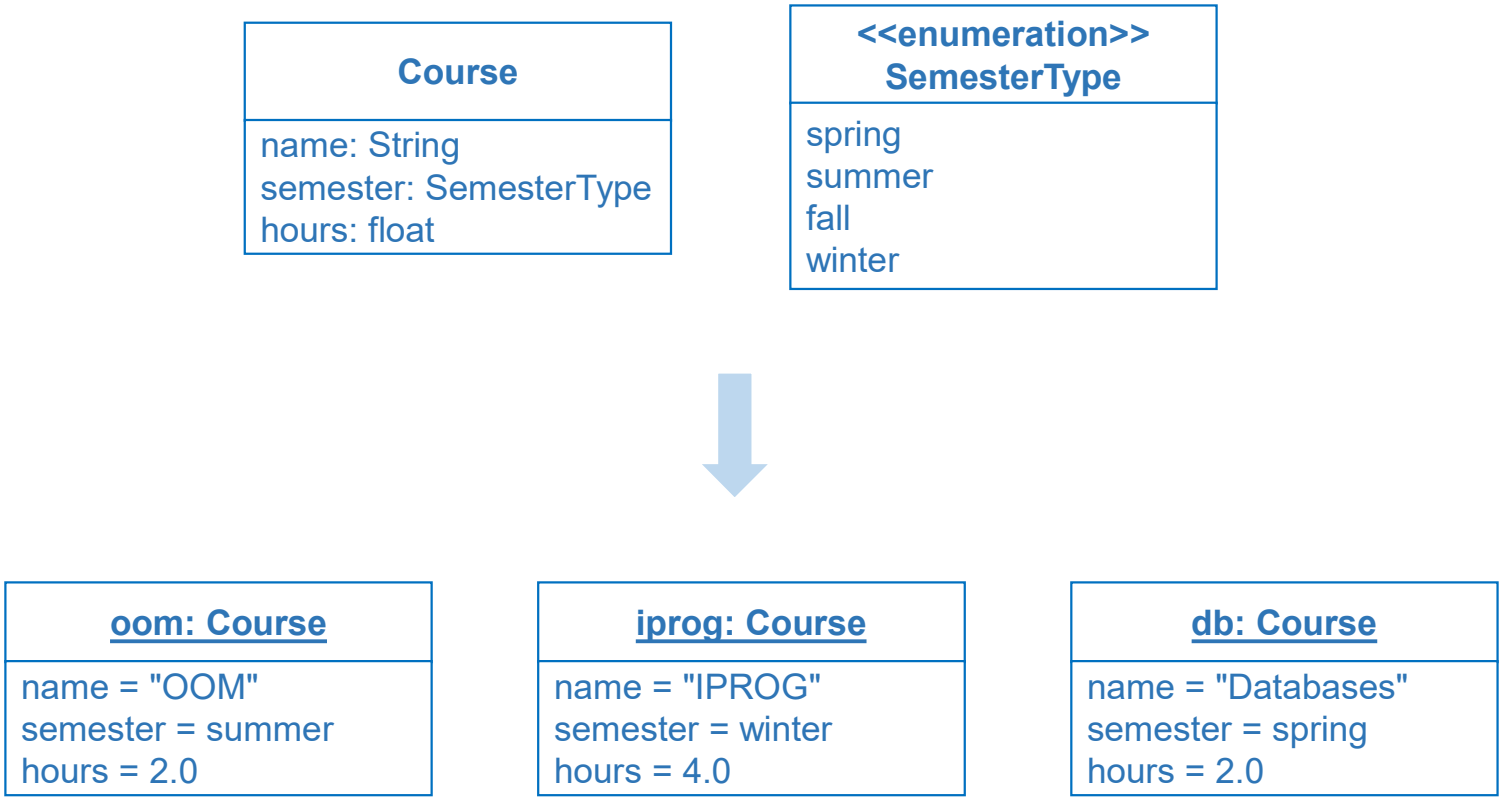
In UML sono rappresentate da **classi**

- etichettate dallo stereotipo **<<enumeration>>**
- con un nome (il tipo) e l'insieme di valori che gli attributi di quel tipo possono assumere



la copertina di un libro può essere
o tutta rossa, o tutta bianca, o tutta blu

ESEMPIO


Course
name: String
semester: SemesterType
hours: float
<<enumeration>>
SemesterType
spring
summer
fall
winter
oom: Course
name = "OOM"
semester = summer
hours = 2.0
iprog: Course
name = "IPROG"
semester = winter
hours = 4.0
db: Course
name = "Databases"
semester = spring
hours = 2.0

RELAZIONI

Una **relazione** rappresenta un legame tra due o più **oggetti**

- (tipicamente) istanze di classi diverse

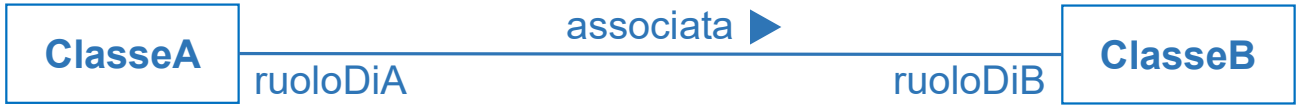
Vedremo

Relazioni tra classificatori	Relazioni tra oggetti
Associazione Aggregazione Composizione	Collegamento Aggregazione Composizione
Generalizzazione	(non definita)
Realizzazione	(non definita)
Dipendenza (d'uso, d'istanza, ecc.)	

ASSOCIAZIONE

Un’associazione si rappresenta con una **linea retta** caratterizzata da

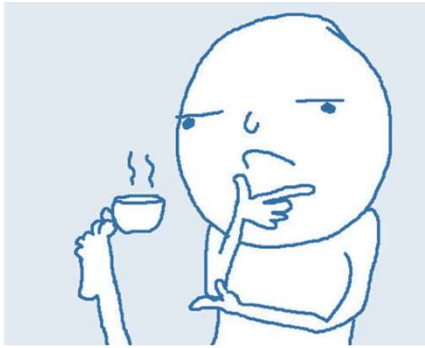
- **nome** (normalmente, un verbo) // associata
- **ruoli** (normalmente, sostantivi) // ruoloDiA, ruoloDiB
- **verso** di lettura //▶ (opzionale e raro, meglio un nome chiaro)



NB: **Almeno uno** tra nome o ruoli, raramente entrambi

NB: Verso dell’associazione (diverso dal verso di lettura) è specificato solo quando si documenta il codice (e non il dominio)

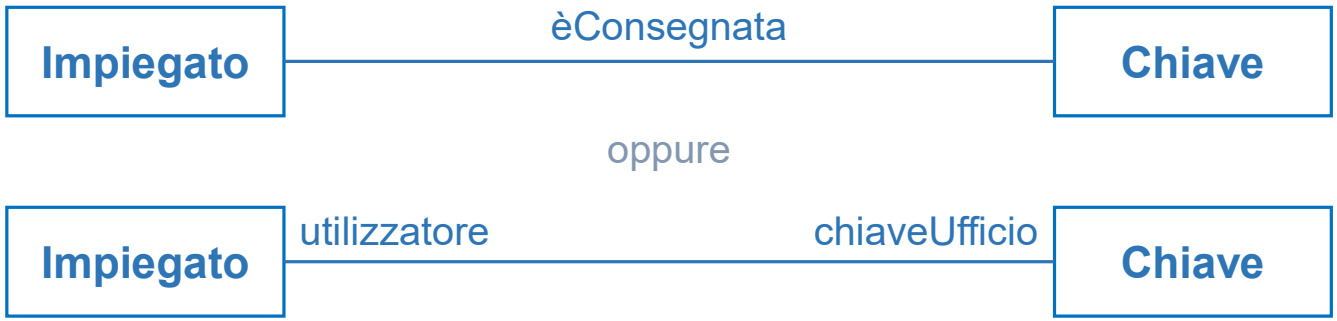
ASSOCIAZIONI O ATTRIBUTI?



La descrizione del dominio usa associazioni

che nel codice diventano attributi

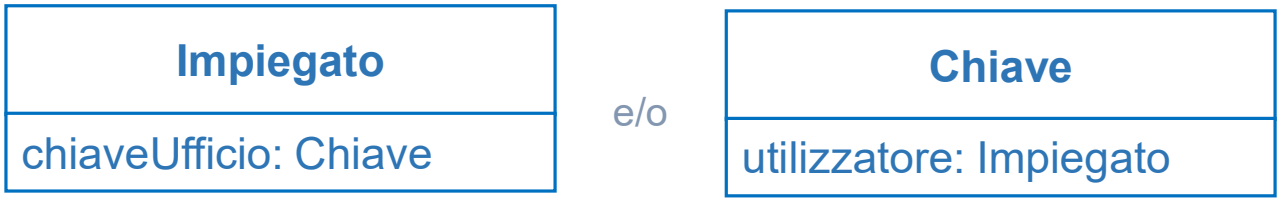
ESEMPIO



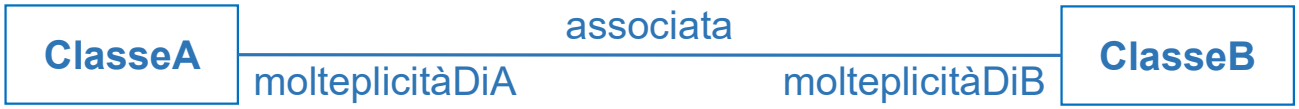
Si esplicitano i ruoli degli oggetti nella relazione

- c'è la chiave dell'ufficio di un impiegato
- tale impiegato ne è l'utilizzatore

Quando si trasforma il modello in codice:

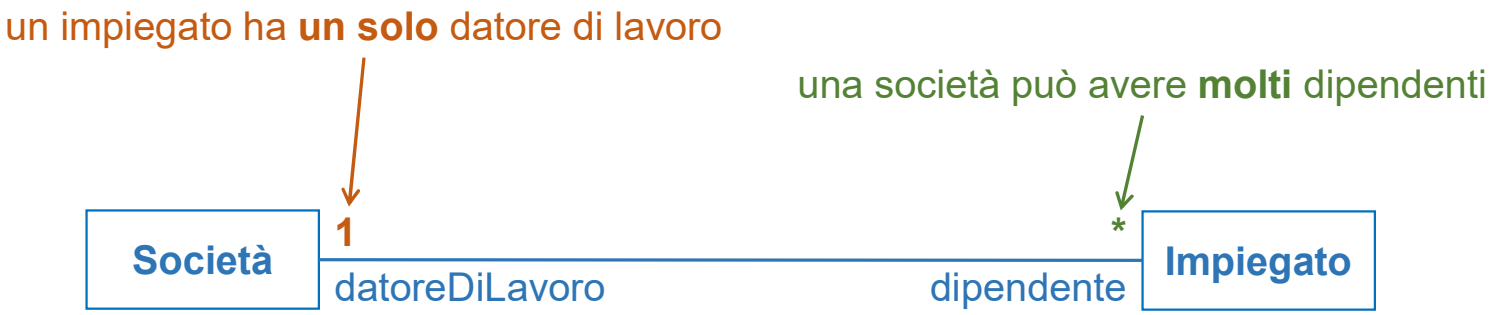


VINCOLI DI MOLTEPLICITÀ



La molteplicità indica il **numero di oggetti coinvolti** nell'associazione **in un dato istante**

ESEMPIO



Relazione 1:*

- Un oggetto Società può essere in relazione con molti oggetti Lavoratore
- Un oggetto Lavoratore può essere in relazione con un solo oggetto Società
  (reminder: in un dato istante di tempo)

VINCOLI DI MOLTEPLICITÀ (CONT.)

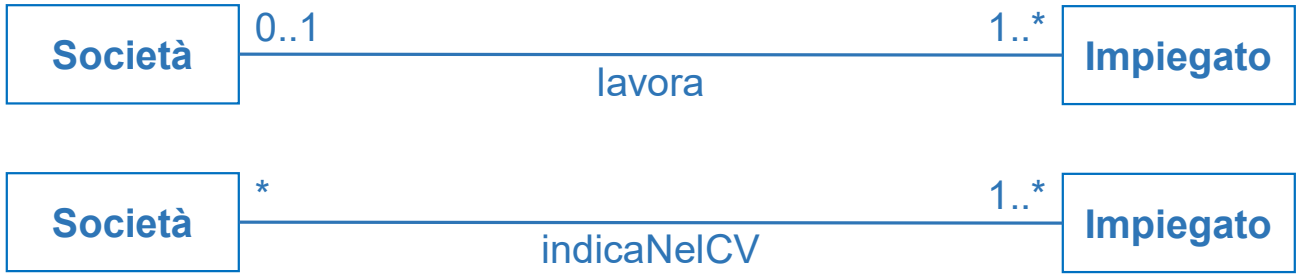
Le molteplicità si possono definire

- Con un **numero positivo** // il valore 1 è il default e si può omettere
- Con il simbolo di **indefinito** (*) // alias per un qualunque numero positivo o uguale a zero
- Indicando gli estremi inferiore e superiore di un intervallo // e.g., 2..4 per le ruote di un veicolo
  - l'estremo **inferiore** può essere **zero o un numero positivo**
  - l'estremo **superiore** un **numero positivo o indefinito** (*)

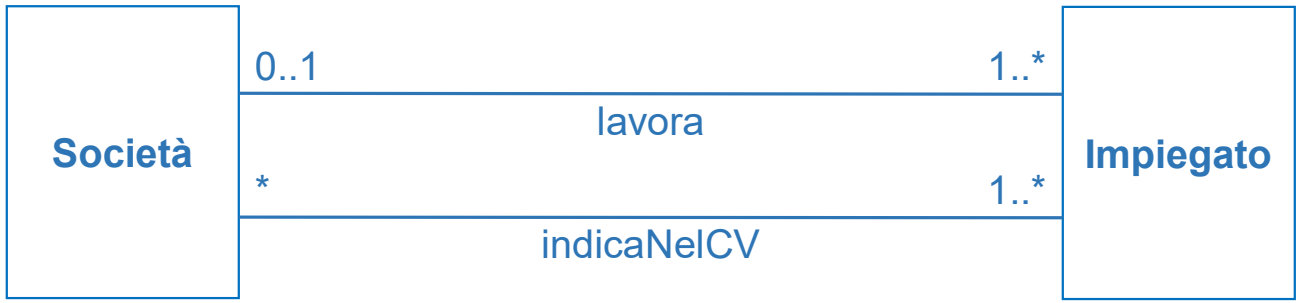
NB: La molteplicità n..n equivale a n, mentre 0..* equivale a *

LEGAME TRA MOLTEPLICITÀ E NOMI

La molteplicità è legata al nome dell'associazione

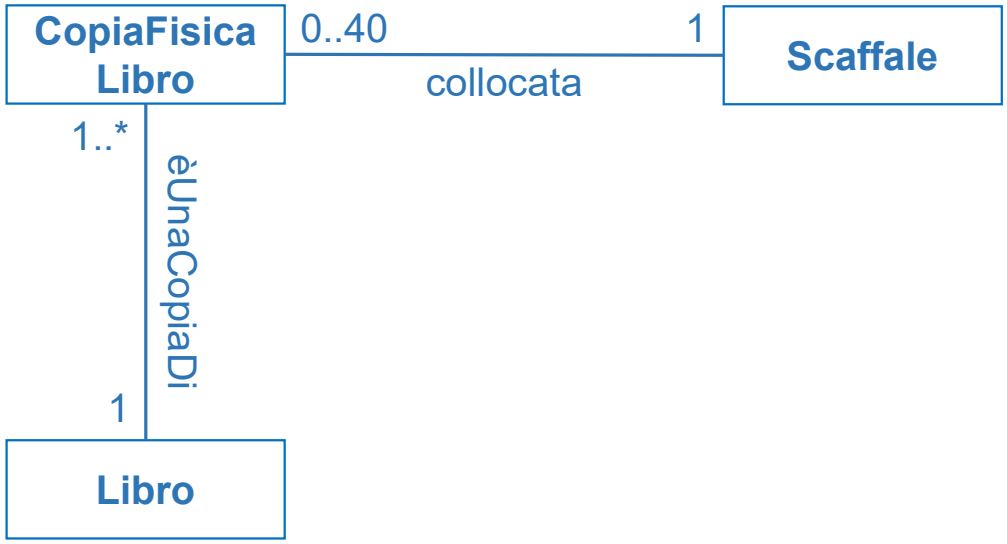


Ci possono essere anche più associazioni tra due classi



ESEMPIO

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



ASSOCIAZIONI RIFLESSIVE

Un'associazione riflessiva mette in relazione un'**entità con se stessa**

- fondamentale indicare il **ruolo**!

AGGREGAZIONE E COMPOSIZIONE

Aggregazione e **composizione** sono tipi particolari di associazione

- specificano che un oggetto di una classe è **una parte** di un oggetto di un'altra classe

  TIP: considerarle quando il nome di un'associazione sarebbe *fa parte di*, *appartiene*, ecc. (o, dualmente, *è composto da*, *possiede*, ecc.)

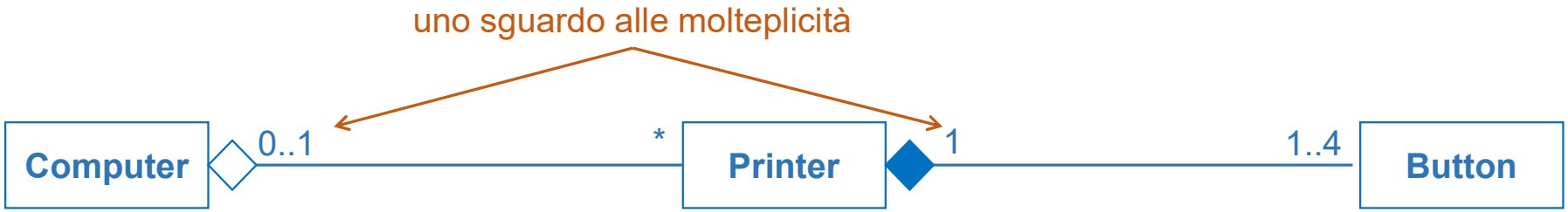
◇ Aggregazione → relazione **poco forte**

classi «parte» hanno significato **anche senza** la classe «tutto»

◆ Composizione → relazione **forte**

classi «parte» hanno significato **solo se legate** alla classe «tutto»

AGGREGAZIONE VS COMPOSIZIONE: ESEMPIO



aggregazione

Una stampante può essere collegata (nel tempo) a computer diversi

Una stampante esiste anche senza computer

Se il computer viene distrutto, la stampante continua a esistere

NB: L'aggregazione non ha un nome

composizione

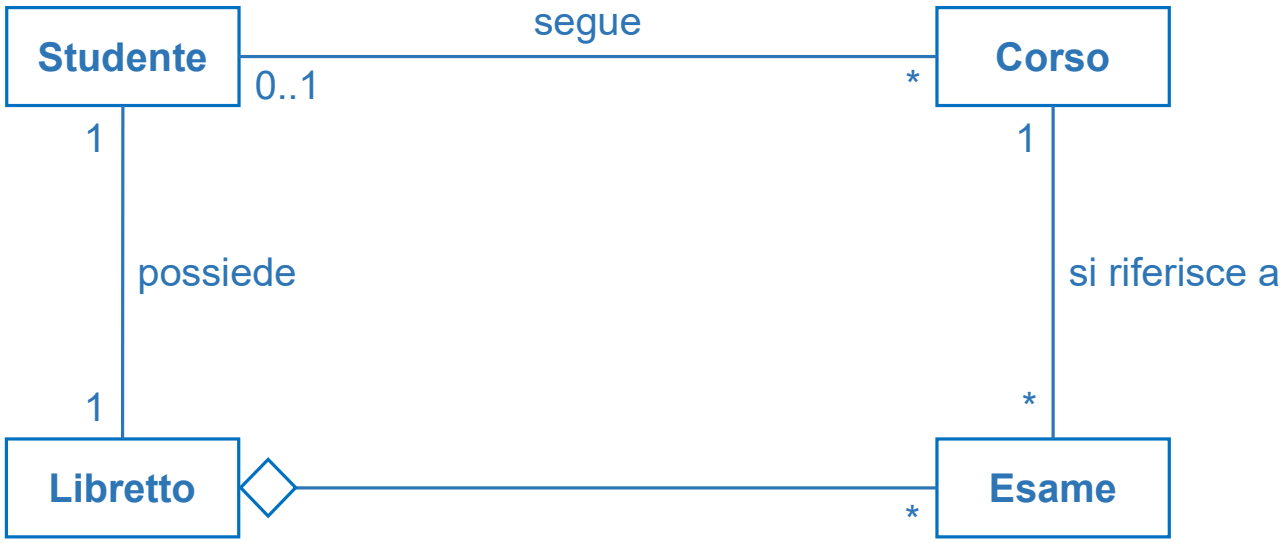
Un pulsante appartiene ad una sola stampante

Un pulsante non esiste senza la sua stampante

Se la stampante viene distrutta, vengono distrutti anche i suoi pulsanti

NB: La composizione non ha un nome

ESEMPIO


Studente
segue
Corso
0..1
*
1
1
possiede
si riferisce a
1
*
Libretto
Esame
*

GENERALIZZAZIONE

Relazione tra un elemento generico ***G*** e uno più specializzato ***S***

- ***S*** è **consistente** con ***G*** ma **contiene più informazione**
- Principio di **sostituzione** della Liskov: ***S*** può essere usato al posto di ***G***
- Intuitivamente: ***S*** *è un tipo di* ***G***

GENERALIZZAZIONE ED EREDITARIETÀ

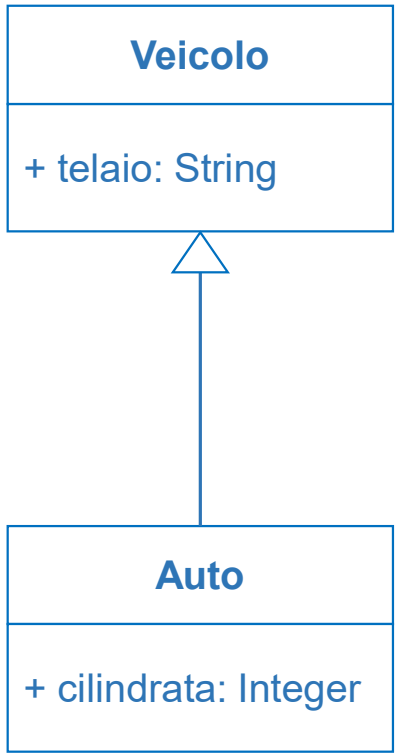
Una superclasse **generalizza** le sue sottoclassi

Le sottoclassi ereditano caratteristiche dalla superclasse

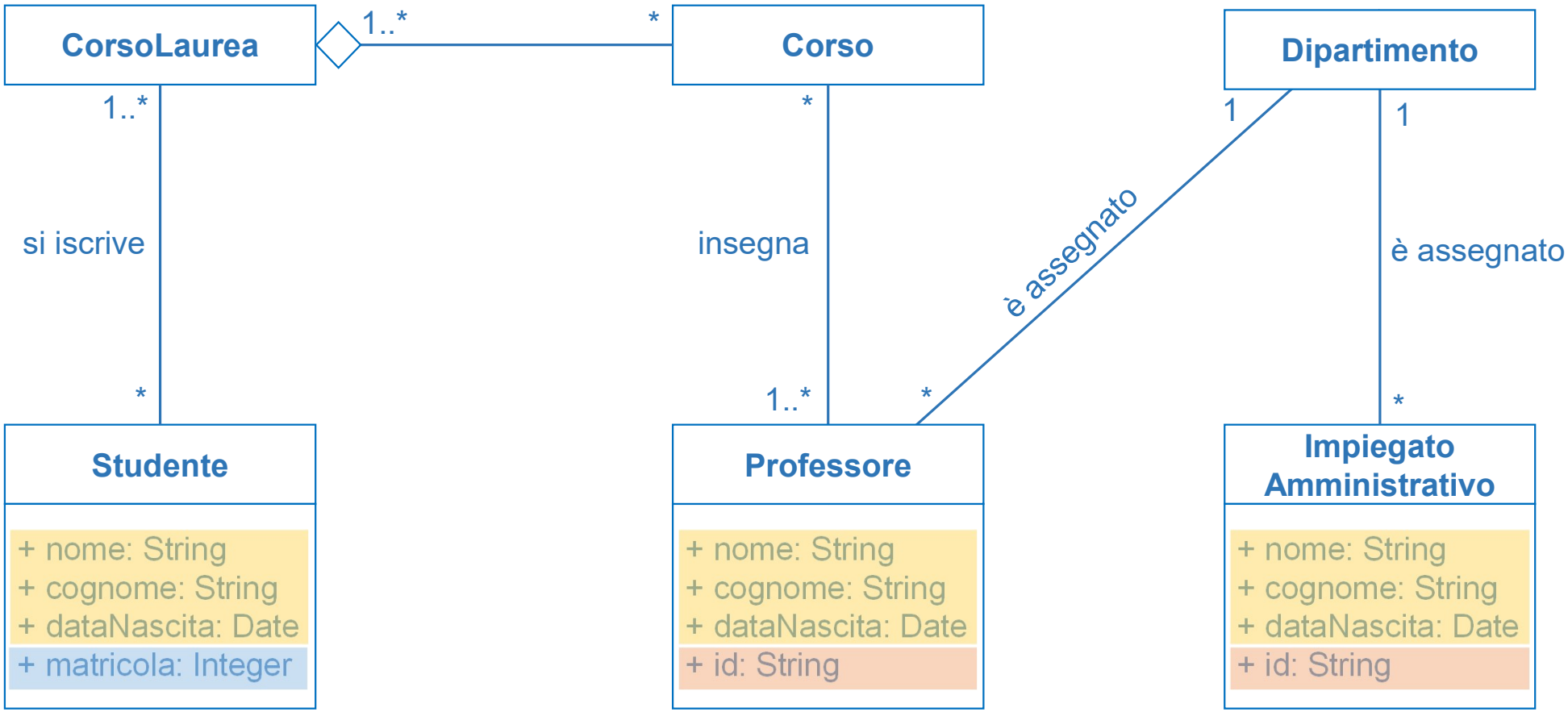
- attributi,
- operazioni,
- relazioni, e
- vincoli

Le sottoclassi può

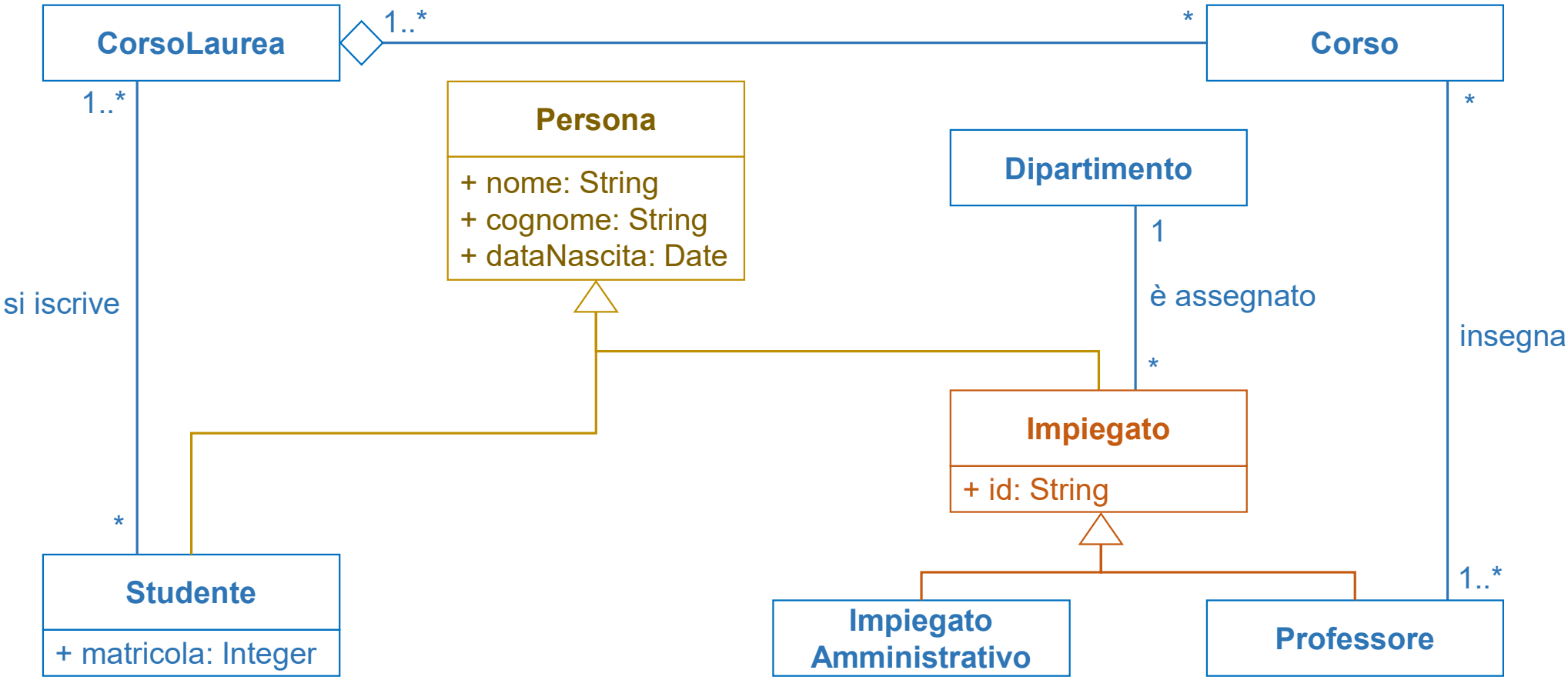
- aggiungere caratteristiche e
- ridefinire le operazioni



UN ESEMPIO DI GENERALIZZAZIONE



UN ESEMPIO DI GENERALIZZAZIONE (CONT.)



CLASSI ASTRATTE E INTERFACCE

Classi astratte

notazione compatta vs notazione estesa

Interfacce (solo comportamento e niente stato)

DIPENDENZE

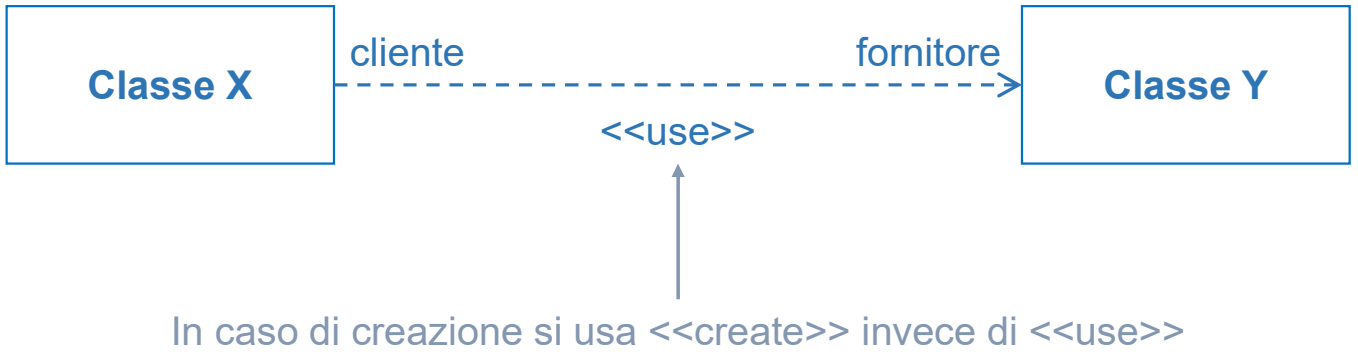
Una dipendenza è una relazione tra una classe **cliente** e una classe **fornitore**

- il cliente **dipende** dal fornitore
- una modifica nel fornitore può influenzare il cliente

ESEMPI DI DIPENDENZE

Alcune dipendenze comuni

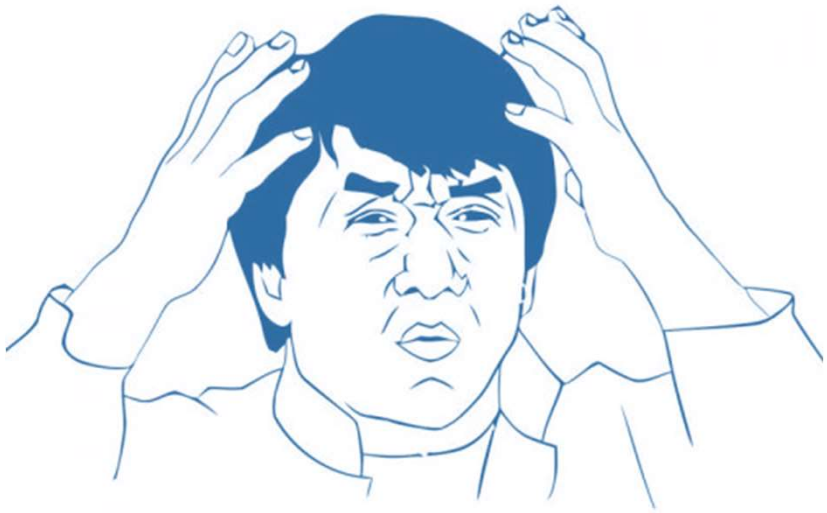
- Un parametro di un'operazione di X è di tipo Y
- Un'operazione di X restituisce un oggetto di tipo Y
- Un'operazione di X crea dinamicamente un oggetto di tipo Y



REWIND: DIAGRAMMA DELLE CLASSI

Una classe cattura un **concetto nel dominio** del problema o della realizzazione

Come si individuano le **classi del dominio**?



INDIVIDUARE LE CLASSI DI ANALISI (DEL DOMINIO)

FAQ: Cosa sono le classi di analisi?

Corrispondono a **concetti concreti** del dominio

- Per esempio, i concetti descritti nel glossario
- Una classe di analisi viene spesso raffinata in una o più classi di progettazione
- Evitare di introdurre delle classi di progettazione

FAQ: Quali tecniche usare per individuare le classi di analisi?

Non esiste un metodo automatico che le estrae da un documento in linguaggio naturale

- Ma possiamo usare alcune metodologie note ☺

CLASSI DI ANALISI: CARATTERISTICHE (E TIP)

- **Astrazione** di uno specifico elemento del dominio
- Hanno un **numero ridotto di responsabilità** (funzionalità)
- Evitare di definire classi “onnipotenti”
  Attenzione quando si chiamano “sistema”, “controllore”, ...
- Evitare funzioni travestite da classi
- Evitare gerarchie di ereditarietà profonde (>=3)

Livello di astrazione/dettaglio

Specifica di **operazioni** e **attributi**
- **solo** quando veramente **utili** e
- ad «**alto livello**» (limitare tipi, valori, ecc)

Non inventare niente, ma **limitarsi** a
- quanto scritto nel **documento** e
- **confronto** con clienti o utenti

CLASSI DI ANALISI: TECNICHE DI IDENTIFICAZIONE

Approccio **data-driven** // tipico della fase di **analisi**

- si identificano i **dati** del sistema (p.e., trovando i sostantivi) e
- si dividono in **classi**

Approccio **responsibility-driven** // soprattutto durante la **progettazione**

- si identificano le **responsabilità** e
- si dividono in **classi**

IDENTIFICAZIONE MEDIANTE ANALISI NOME-VERBO

sostantivi → classi o attributi

verbi → operazioni (responsabilità)

Passi

1. Individuazione delle classi
2. Assegnazione di attributi e responsabilità alle classi
3. Individuazione di relazioni tra le classi

Problemi ricorrenti

- Tagliare le **classi inutili** (per esempio, trattando casi di sinonimia)
- Individuare le **classi nascoste** (classi implicite del dominio del problema che possono non essere menzionate esplicitamente)

Esempio: In un sistema di gestione degli orari delle lezioni di un corso universitario, nella descrizione testuale potrebbe non essere mai nominata l'aula, che invece deve essere inserita nel modello

ESERCIZIO

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.

ESERCIZIO (PRIMA OCC. SOSTANTIVI IN GRASSETTO)

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.

ESERCIZIO: PRIMA BOZZA

DIGRESSIONE: CLASSI UML VS ENTITÀ (BASI DI DATI)

Classi UML (IS)

nome **singolare**
(«tipo» di un oggetto)

Entità (BD)

nome **plurale**
(sono intese come collezioni)

La differenza è significativa più in prospettiva di progettazione che di descrizione del dominio

«ListaDiQualcosa» per aggregare
istanze di «Qualcosa»

unica entità per rappresentare
tutte le istanze di «Qualcosa»

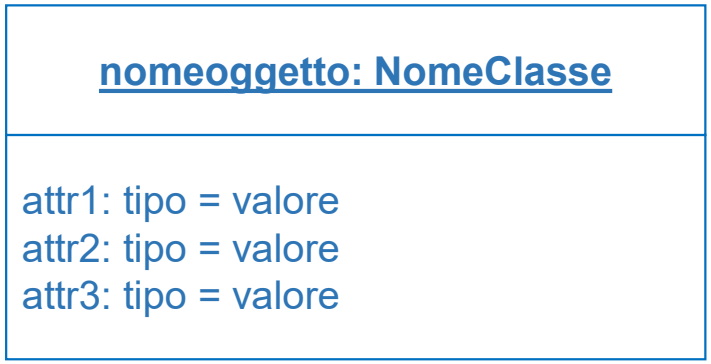
Ad esempio

E GLI OGGETTI?

DIAGRAMMA DEGLI OGGETTI

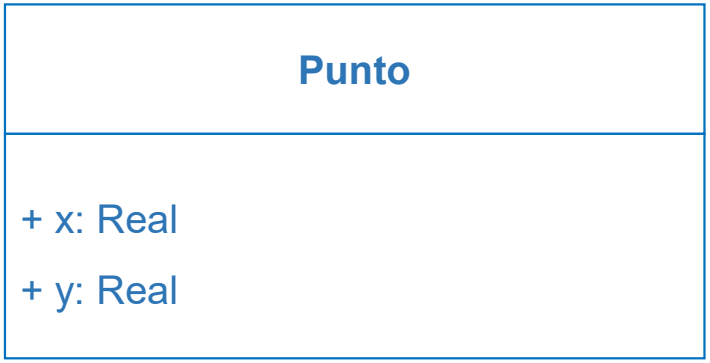
Un **oggetto** si rappresenta con due sole sezioni

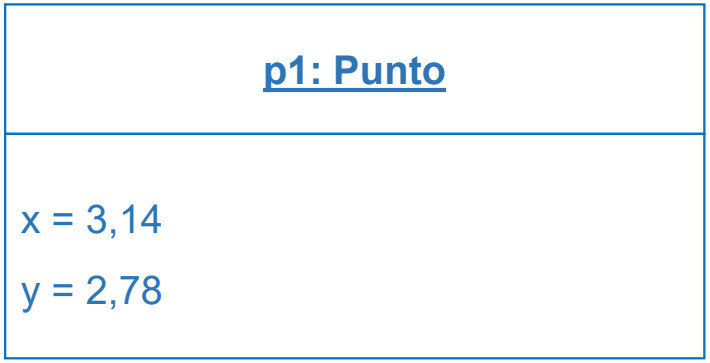
- **nome** dell'oggetto e della classe che istanzia, **sottolineati**
- lista degli **attributi** (opzionale e rarissima)
  - **nome** attributo
  - **tipo** attributo (ridondante, meglio ometterlo)
  - **valore** attributo



Il **valore** è la **parte interessante** (senza quello, inutile ripetere quanto già specificato nel diagramma delle classi)

ESEMPIO





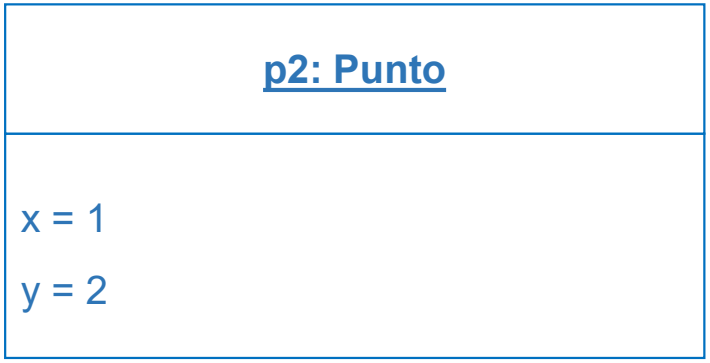


DIAGRAMMA DEGLI OGGETTI (CONT.)

Un **collegamento** istanzia un'associazione nel diagramma delle classi

- Collega due o più oggetti
- Non ha nome
- Se utile, si possono indicare i ruoli
- La molteplicità è sempre 1:1
  (la molteplicità di un'associazione dice quanti collegamenti saranno istanziabili)

ESEMPIO

Le associazioni istanziate si **deducono** dal tipo degli oggetti

Diagramma delle **classi**

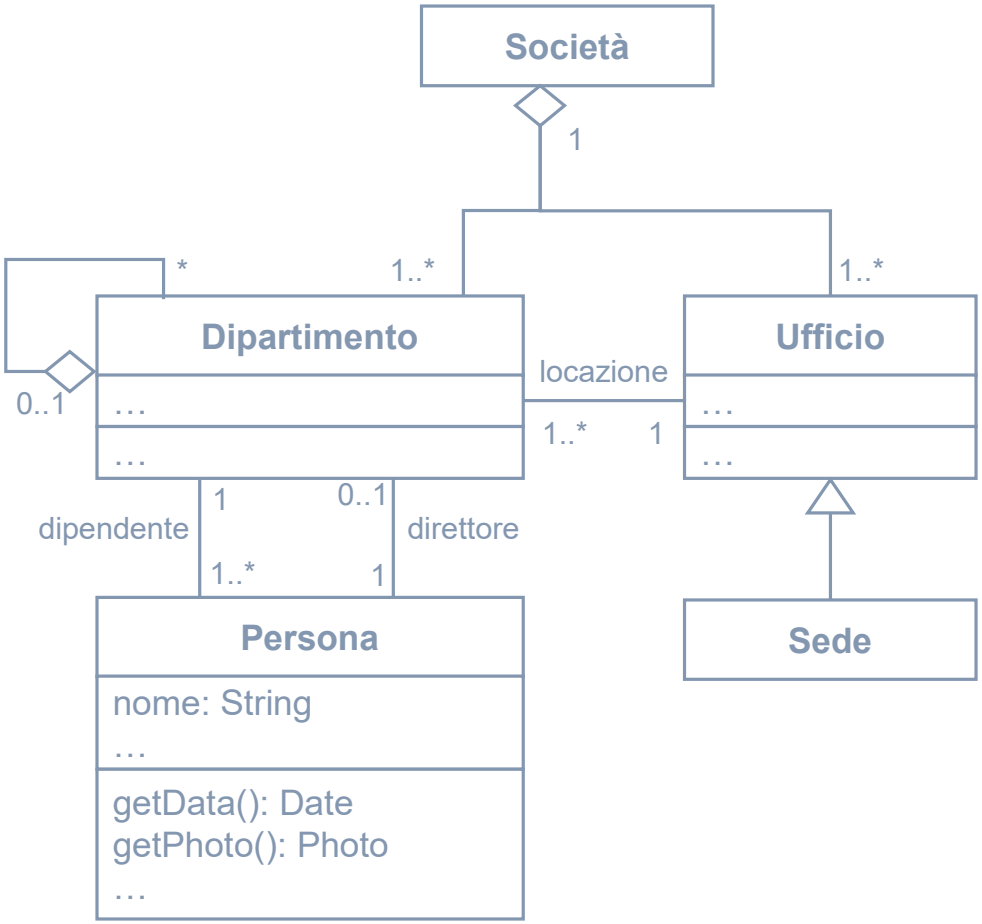
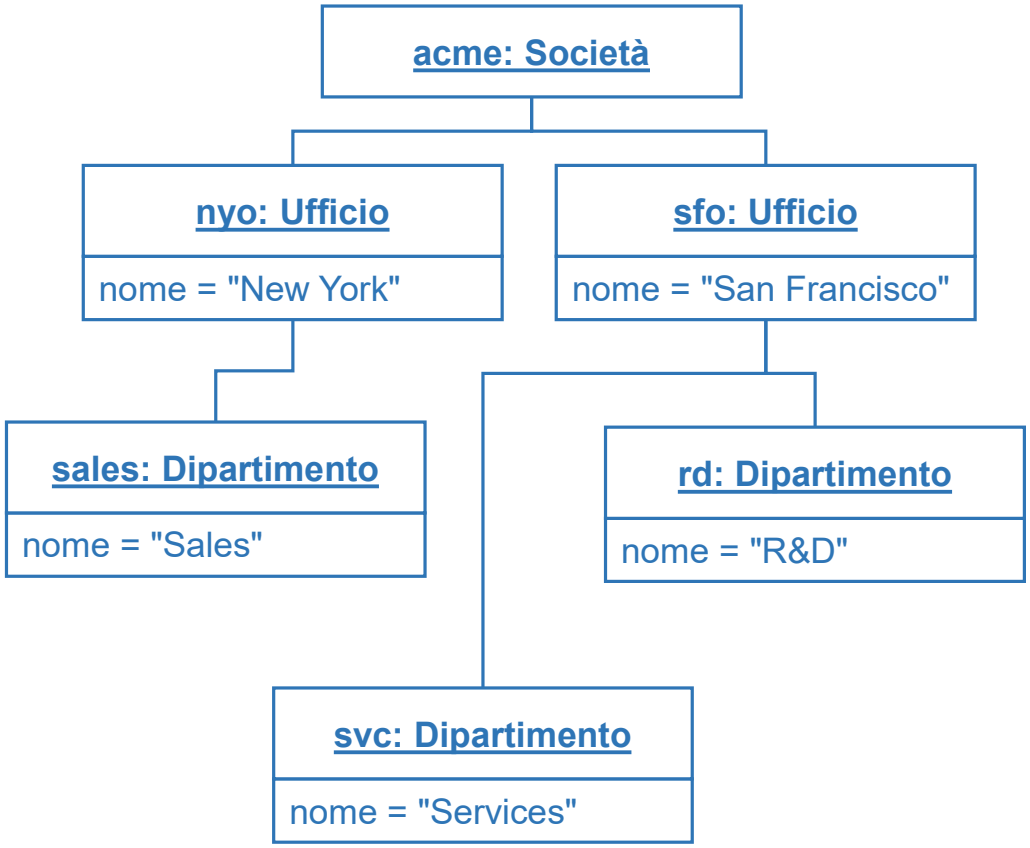
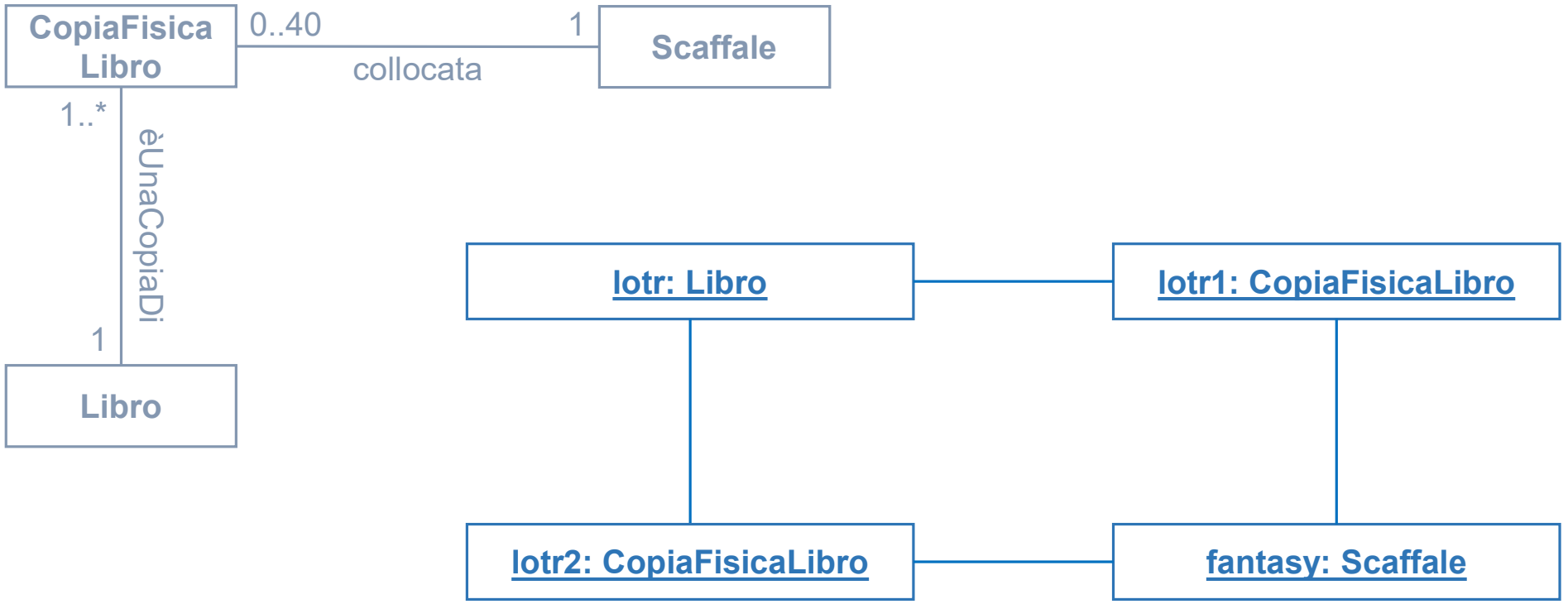


Diagramma degli **oggetti**



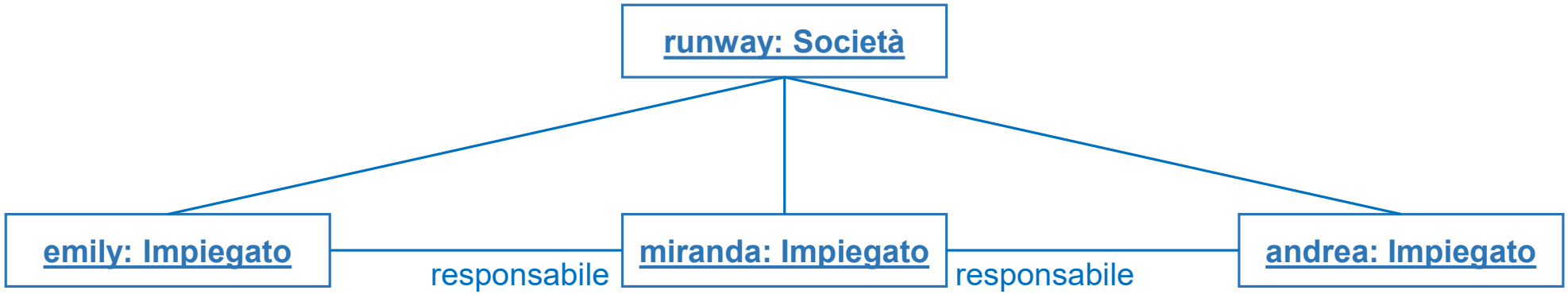
ALTRI ESEMPI

Le associazioni istanziate si possono **dedurre** dal tipo di oggetti istanziati



ALTRI ESEMPI (CONT.)

Quando l'associazione istanziata non è ovvia, meglio **specificare i ruoli**

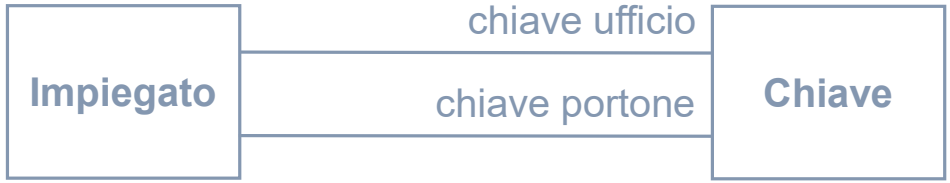


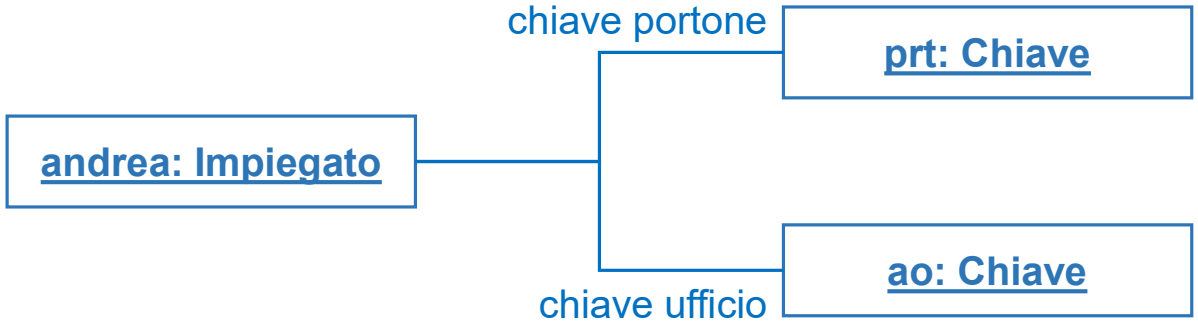
ALTRI ESEMPI (CONT.)

Quando l'associazione istanziata non è ovvia, meglio **specificare i ruoli**

ALTRI ESEMPI (CONT.)

Quando l'associazione istanziata non è ovvia, meglio **specificare i ruoli**





ALTRI ESEMPI (CONT.)

Quando l'associazione istanziata non è ovvia, meglio **specificare i ruoli**


Q
A

RIFERIMENTI

Contenuti

- **Capitolo 4** di "UML@Classroom" (M. Seidl et al., 2015)

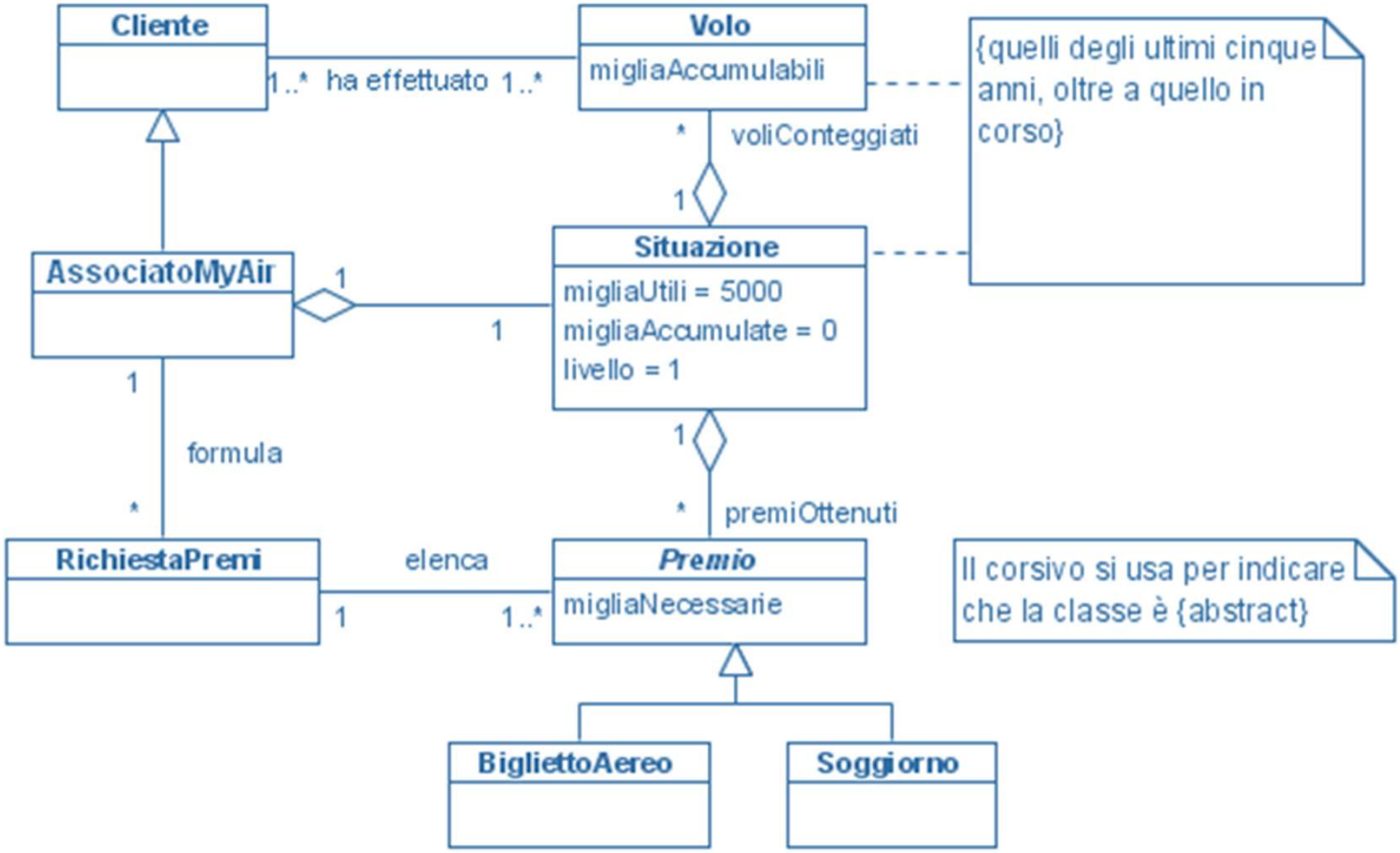
Approfondimenti

- **Capitolo 5** di "Software Engineering" (G. C. Kung, 2023)

HOMEWORK: 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.

HOMEWORK: MYAIR (SOLUZIONE)

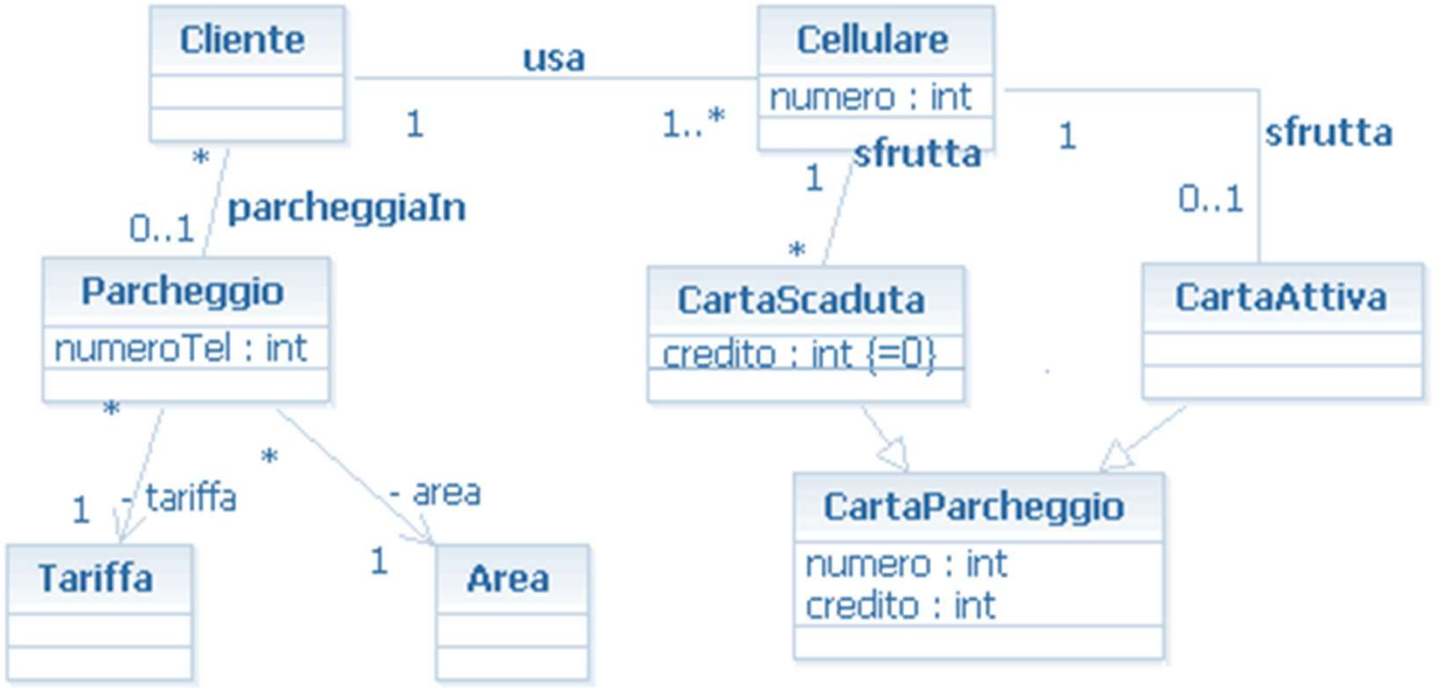


HOMEWORK: EASYPARK

Soluzione per il pagamento del parcheggio via telefono cellulare

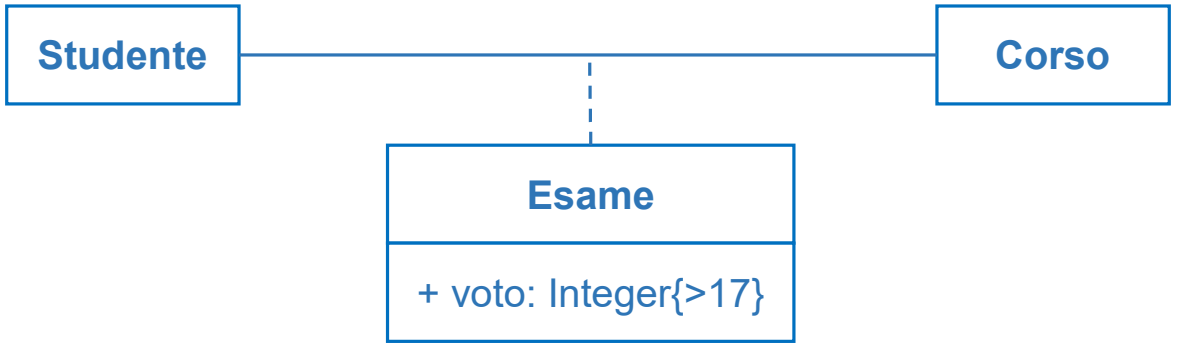
1. Il cliente acquista una Carta Parcheggio prepagata e l'attiva indicando il proprio numero di cellulare. Durante l'attivazione, il sistema trasferisce sulla nuova carta l'eventuale credito residuo su una carta già associata al numero di telefono indicato.
2. Il cliente parcheggia ed espone sul cruscotto 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.

HOMEWORK: EASYPARK (SOLUZIONE)



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

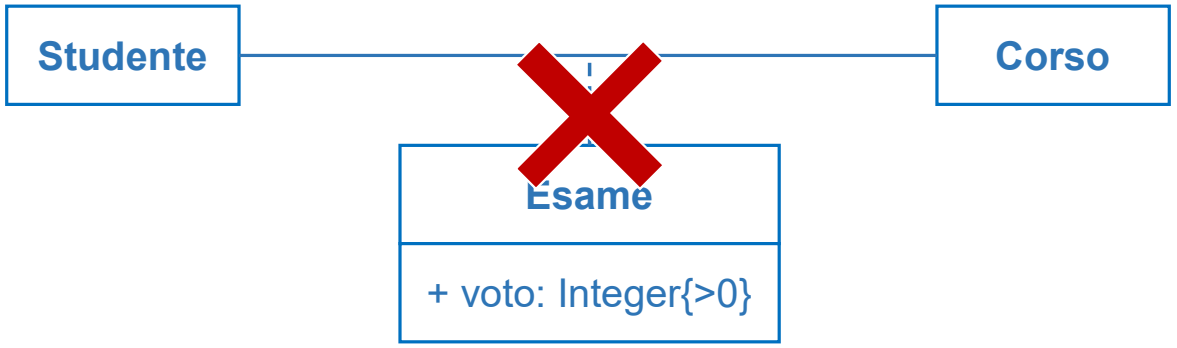
APPENDICE: CLASSI ASSOCIAZIONE



- Un’associazione può avere attributi propri, rappresentati con una **classe associazione**
- Le istanze sono collegamenti con attributi propri
  (nell’esempio, voto non è attributo né di Corso né di Studente)

APPENDICE: CLASSI ASSOCIAZIONE (CONT.)

Per ogni coppia di oggetti collegati, può esistere un **unico oggetto** della classe associazione



Se vogliamo tenere traccia dei voti negativi, non possiamo usare le classi associazione