Il paradigma Orientato agli Oggetti MODELLAZIONE DELLE RELAZIONI TRA CLASSI mediante UML (prima parte)

Programmazione II corso A
Corso di Laurea in ITPS
Università degli Studi di Bari Aldo Moro
a.a. 2022-2023
ultima modifica: 25 ottobre 2022

Testo di riferimento: «UML DISTILLED» di Martin Fowler, Pearson

Sommario

- Il linguaggio UML *(già affrontati)*
- Classi ed oggetti in UML *(già affrontati)*
- Relazioni tra classi
  - Associazione
  - Aggregazione
  - Classe associativa
  - Generalizzazione, «insieme generalizzazione» (generalizationset)
  - Dipendenza
  - Implementazione Interfaccia
- Aggregazione/Composizione Vs. ereditarietà
- Classi Astratte
- Interfacce
  - Ereditarietà multipla
- Classi nidificate

DIAGRAMMA DELLE CLASSI

Diagramma delle Classi

Grafico che fornisce una vista strutturale del sistema in termini di

- classi
- attributi
- operazioni
- relazioni tra classi

Ossservazione

- non fornisce alcuna informazione temporale

Relazioni tra Classi

- Le principali relazioni sono
  - Associazione
  - Aggregazione
  - Composizione
  - Classe Associativa
  - Generalizzazione
  - GeneralizationSet
  - Dipendenza
  - Implementazione Interfaccia

Relazione di associazione

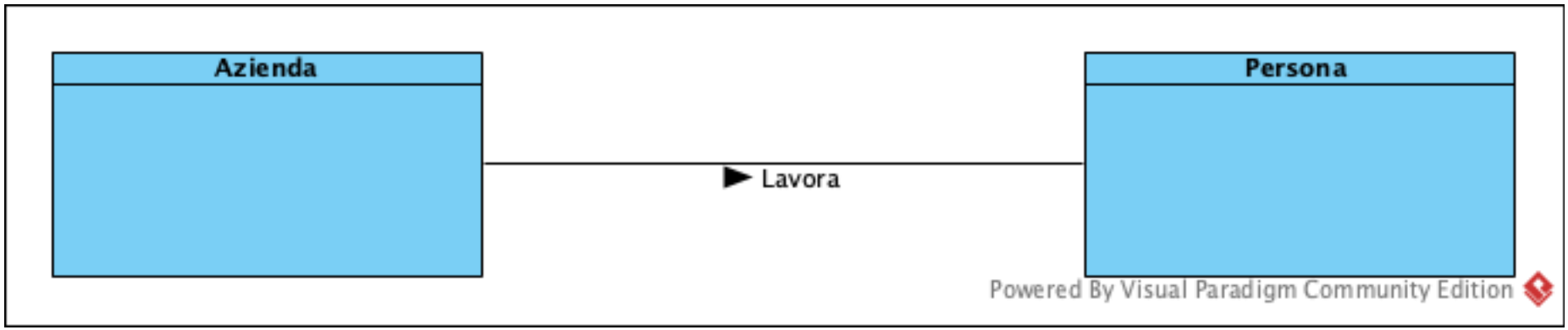
- Definizione: indica la presenza di un legame logico relazionale tra entità del dominio applicativo del problema (tra istanze di classi)
  - Esempi:
    - studente «iscrizione» Corso di Laurea;
    - sportivo «pratica» uno sport;
    - batteria «parte di» un cellulare;
    - Libro «composto da» pagine
- Proprietà
  - è sempre bi-direzionale
  - può essere binaria (coinvolge due entità/classi) o n-aria (coinvolge tre o più entità/classi)
- Osservazione
  - un'associazione binaria è un caso speciale di relazione n-aria con una propria notazione

Un esempio di associazione binaria: persona-azienda

A partire dai seguenti requisiti:
R1. Una persona lavora presso un'azienda;
R2. presso un'azienda lavorano delle persone.
Si potrebbe modellare la relazione di associazione binaria di sotto mostrata

Osservazione: Tra due entità ci potrebbe essere una associazione qualora nei requisiti ci fosse un verbo che crei un legame tra esse. Nell'esempio *«lavora»*

"Lavora" è il nome dell'associazione (opzionale);
è possibile porre accanto al nome un piccolo triangolo nero indicante la direzione in cui leggere la relazione (l'implementazione del triangolo differisce in base al tool utilizzato)



Associazioni: specifica della molteplicità

Nei requisiti dell'esempio precedente non è in alcun modo specificato:

- quale sia il numero minimo di persone che possono lavorare presso un'azienda;
- quale sia il numero massimo di persone che possono lavorare presso un'azienda;
- presso quante aziende una persona può lavorare.

- IMPORTANTE: In una relazione di associazione (binaria) è necessario specificare, sempre, quante istanze di una classe possono essere associate con una singola istanza dell'altra classe

- OSSERVAZIONE: Qualora la molteplicità non fosse indicata, NON SI OPERA ALCUNA ASSUNZIONE. Di fatto si rimanda la specifica della molteplicità alla fase di codifica della modellazione

Associazioni: notazione per la molteplicità

⇨ Notazione

- ❑ stringa di testo che comprende sequenze di intervalli di interi separate da una virgola
- ❑ un intervallo, potenzialmente illimitato, è nel formato
  - **limite inferiore..limite superiore**

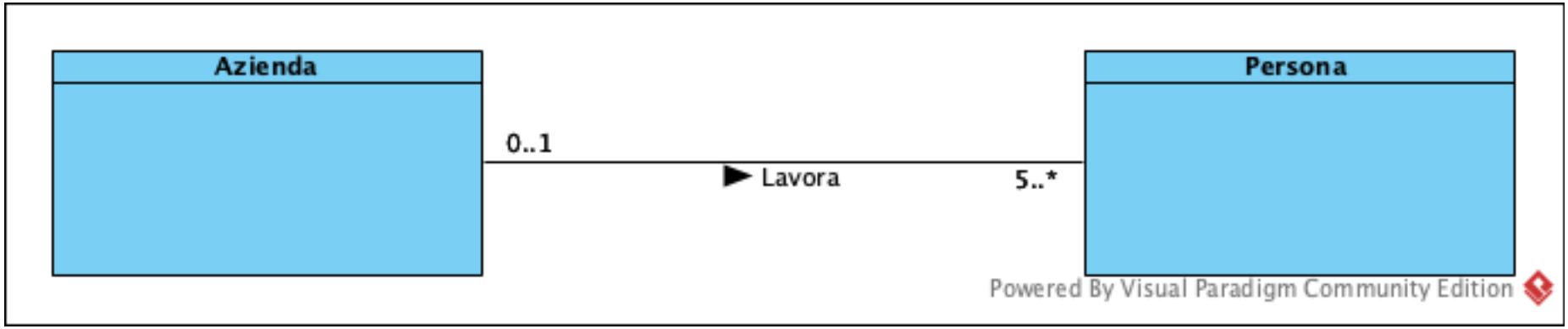
⇨ Esempi di molteplicità

- ❑ 0..1 (da zero ad una istanza)
- ❑ 0..* (zero o più istanze)
- ❑ 1..* (una o più istanze)
- ❑ 1..6 (da una a sei istanze)
- ❑ 1..3, 7..10, 15, 19..*

Specifica della molteplicità in «persona-azienda»

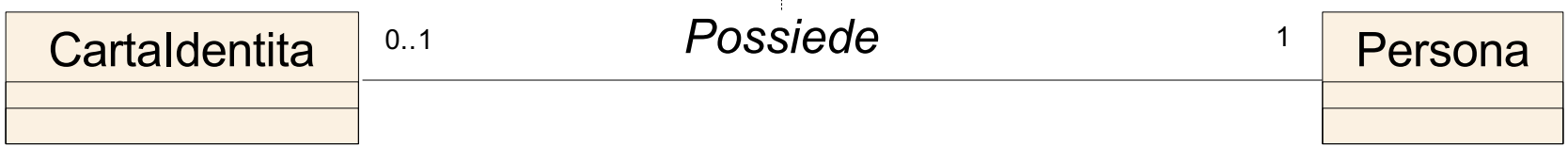
Se oltre ai requisiti *R1 ed R2 per l'esempio «persona-azienda» avessimo anche i seguenti:*
R3. Presso un'azienda devono lavorare almeno 5 persone;
R4. Una persona può lavorare al più presso un'azienda

La relazione, in precedenza modellata, sarebbe ulteriormente specificata come di sotto mostrato



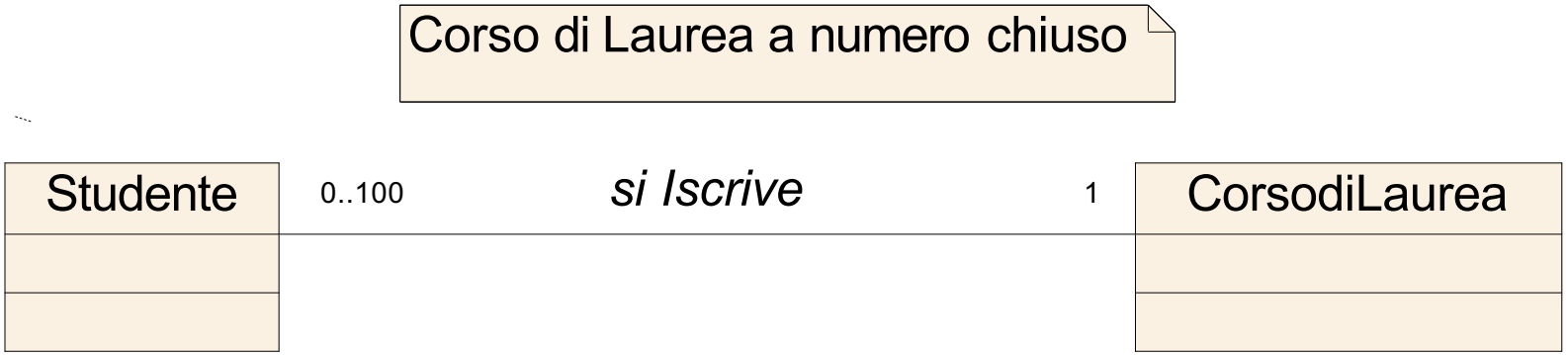
Molteplicità nelle associazioni: esempi …

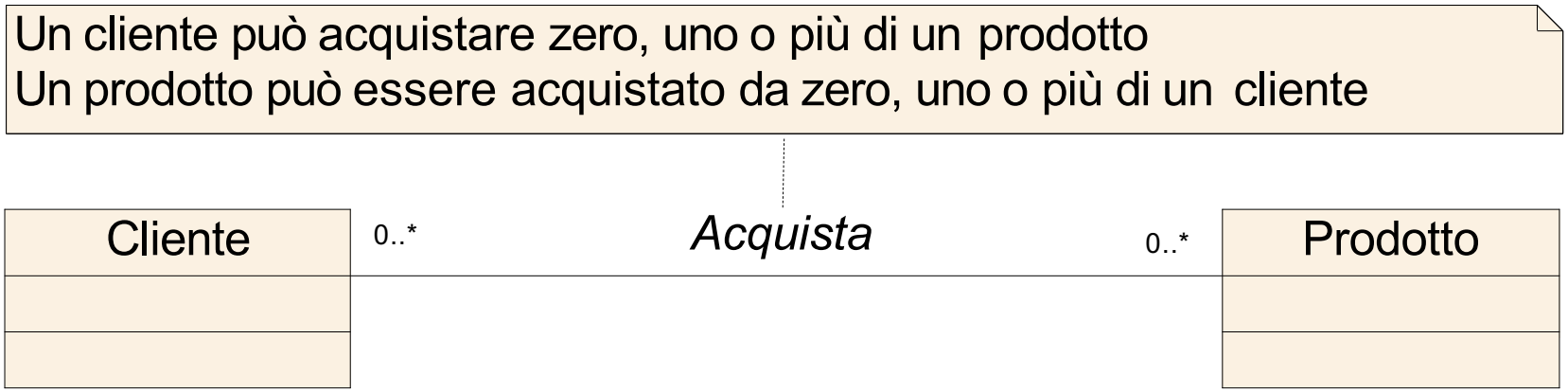
Una persona può possedere al più una carta d'identità
Una carta d'identità è posseduta da una ed una sola persona



Una persona lavora al più per una azienda (vincolo committenza)
Una azienda ha almeno una persona che vi lavora

… Molteplicità nelle associazioni: esempi





I ruoli nelle associazioni

Sempre in riferimento all'esempio «persona-azienda» i requisiti potrebbero anche specificare il ruolo che una persona riveste quando lavora presso un'azienda. Ad esempio si potrebbe specificare che

R5. una persona che lavora presso un'azienda è impiegata presso tale azienda;

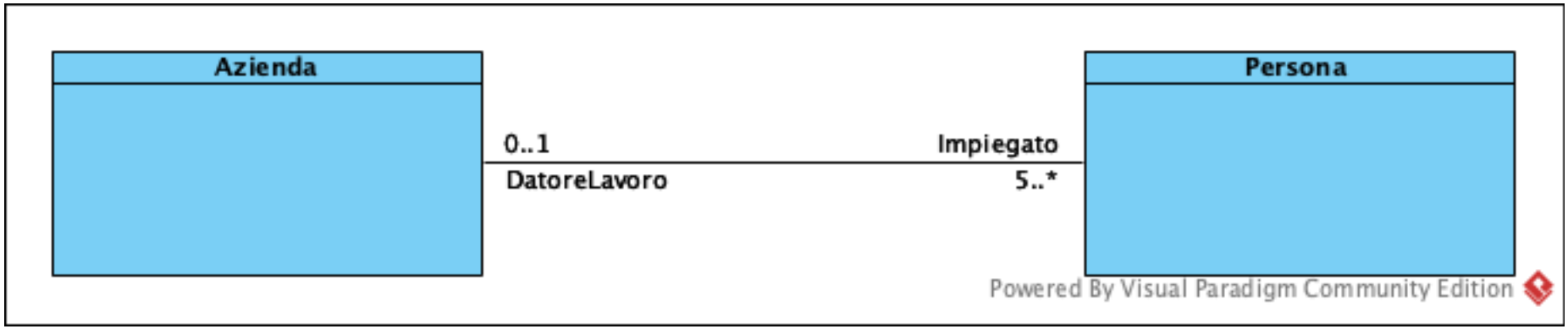
R6. un'azienda è il datore di lavoro dei propri impiegati.

I ruoli forniscono una modalità per attraversare le relazioni da una classe all'altra

⇨ Generalità sui ruoli

- ❑ Possono essere usati in alternativa ai nomi delle associazioni
- ❑ spesso usati per relazioni tra oggetti della stessa classe (relazione riflessiva)
- ❑ «Suggeriscono» al programmatore il nome della variabile che definisce l'associazione

Persona-azienda con la specifica dei ruoli



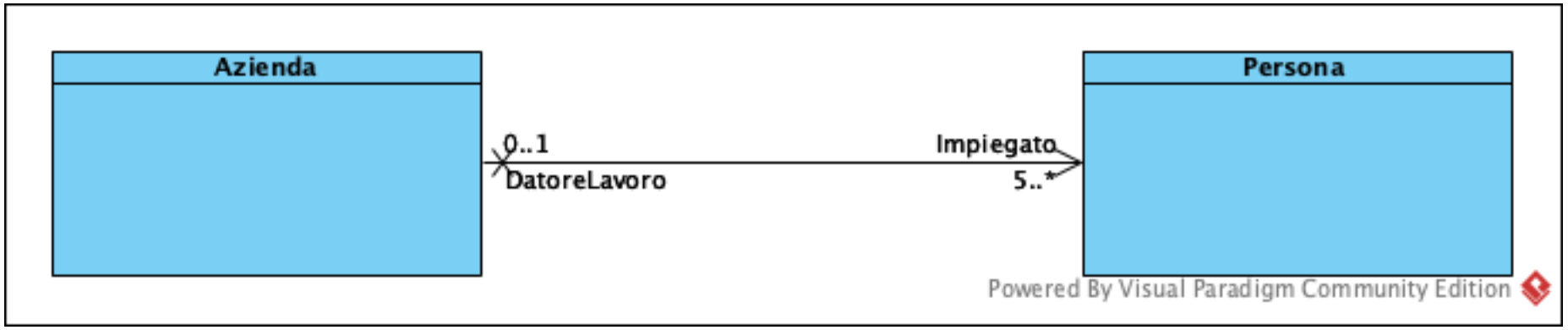
Notazione dei ruoli: ulteriori esempi

Il nome di un ruolo di una classe è espresso da un sostantivo ed è posto accanto ad essa alla fine del simbolo di associazione

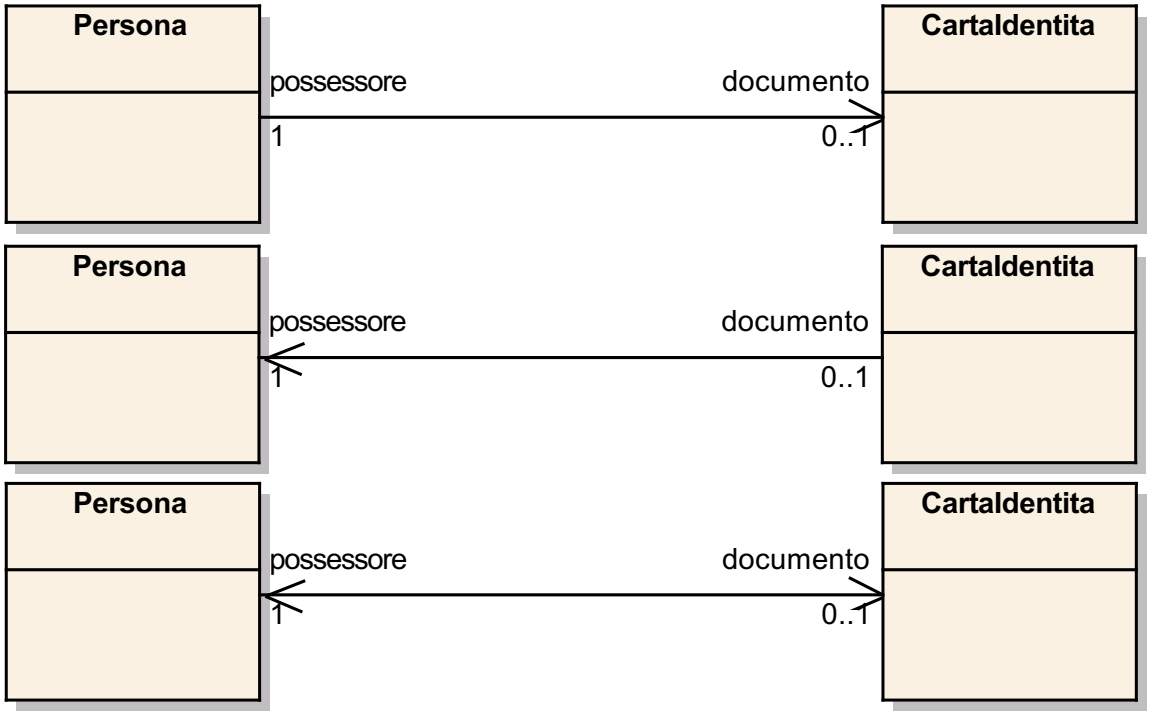
Persona-azienda con la specifica della navigabilità

Per quanto, per definizione, una relazione sia sempre bidirezionale è sempre necessario volerla navigare in ambedue le direzioni?

I requisiti di «azienda-persona» potrebbero specificare che:
R7. Il sistema realizzerà un'interfaccia grafica mediante la quale si potranno selezionare le aziende, una per volta; una volta selezionata un'azienda sarà possibile accedere all'elenco di tutti i suoi impiegati



Navigabilità delle associazioni: ulteriori esempi



possibili navigabilità per un'associazione tra due classi

Implementazione dell'associazione…

Generalità

- non esiste un *mapping* diretto su un costrutto di un qualunque linguaggio di programmazione
- può essere implementata introducendo degli attributi *ad hoc* che contengono dei riferimenti alle istanze delle classi associate
- il riferimento a più istanze di una classe può essere modellato con delle strutture dati quali, ad esempio, liste, array, …
- in fase di implementazione si può decidere di codificarla solo in una delle due direzioni

… Implementazione dell'associazione

- In ogni classe partecipante all'associazione si definisce un'operazione *associa* la cui semantica è "crea un riferimento all'altra classe"
- L'associazione, di fatto, si realizza a run-time all'interno di un'altra classe che passa, tramite l'operazione *associa*, ad ogni oggetto, partecipante all'associazione, i riferimenti degli altri oggetti

Implementazione associazione 1 A 1 (*static-time*)


```
// navigabilità da A verso B
class A {
   private B associazionex; //nome dell'associazione  o del ruolo
   public boolean setB(B partner);
}
```

Associazione «1 a molti»

⇨ Un'associazione uno a molti tra una classe A ed una classe B descrive una connessione semantica per cui a ciascun oggetto di A possono corrispondere zero, uno o più oggetti di B mentre ad un oggetto di B corrisponde al più un oggetto di A

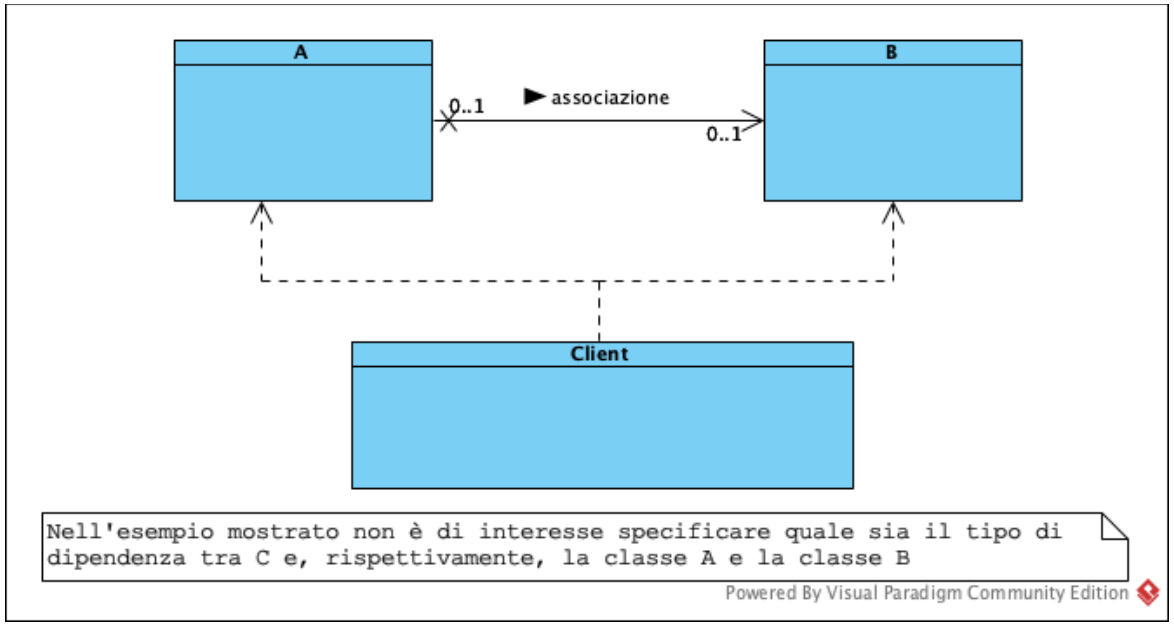
- la classe che partecipa con molteplicità 1 nell'associazione rappresenta la relazione attraverso una variabile di tipo contenitore di riferimenti alla classe associata
- se è noto il numero massimo di oggetti associati si usa un contenitore statico, in caso contrario un contrario dinamico

Schema implementazione 1 a molti (*static-time*)


```
// navigavibilità da A verso B
class A{
   private ContenitoreB[0..*] b; // contenitore di oggetti di tipo B
   // addB: accoda l'elemento newB a quelli già presenti
   public boolean    addB (B newB);
   // addAllB: accoda gli elementi di bElements a quelli già presenti
   public boolean addAllB(Contenitoreb[0..*] bElements);
}
```

Schema implementazione 1 a molti (*run-time*)

```
class Client{
   // omissis attributi ed operazioni
   public void something () {
            // omissis istruzioni
            a instantiation of A;
            // omissis istruzioni
            b instantiation of B;
   // istruzione che, a run-time,
   //associa ad un oggetto a un oggetto b
            a.add(b);
   }
}
```



Problema (da svolgere) sulla relazione di associazione

- Problema: si fornisca una modellazione OO dei requisiti di seguito riportati:

  R1. una persona possiede al più una carta di identità;

  R2. una carta di identità è posseduta da una ed una sola persona;

  R3. una persona è caratterizzata dai seguenti attributi:
    - Nome, cognome, luogo e data di nascita;

  R4. una carta identità è caratterizzata da
    - un numero identificativo; un ente di rilascio; una data di rilascio; nome e cognome del detentore della carta; luogo e data di nascita del detentore della carta;

Classe associativa

- Talvolta le proprietà di un'associazione sono proprie della relazione e non delle classi coinvolte, pertanto può essere opportuno definire una classe associativa

  Esempio: studente si iscrive ad un corso di laurea; l'iscrizione è caratterizzata da una data di iscrizione, un'a.a., un costo, …

  - utilizzata, in genere, nelle associazioni "uno a molti" e "molti a molti"

- Osservazioni

  - la classe associativa potrebbe contenere un riferimento ad ognuno degli oggetti in relazione (*doppia navigabilità*)
  - ogni oggetto in relazione potrebbe contenere un riferimento ad un oggetto della classe associativa (*doppia navigabilità*)

Esempio di classe associativa: studente-insegnamento

Problema:

R1. Uno studente può frequentare degli insegnamenti
R2. Un insegnamento può essere frequentato da studenti
R3. La frequenza di un insegnamento prevede:
- una data di inizio, una data di fine, un anno accademico di frequenza ed numero di presenze da parte dello studente

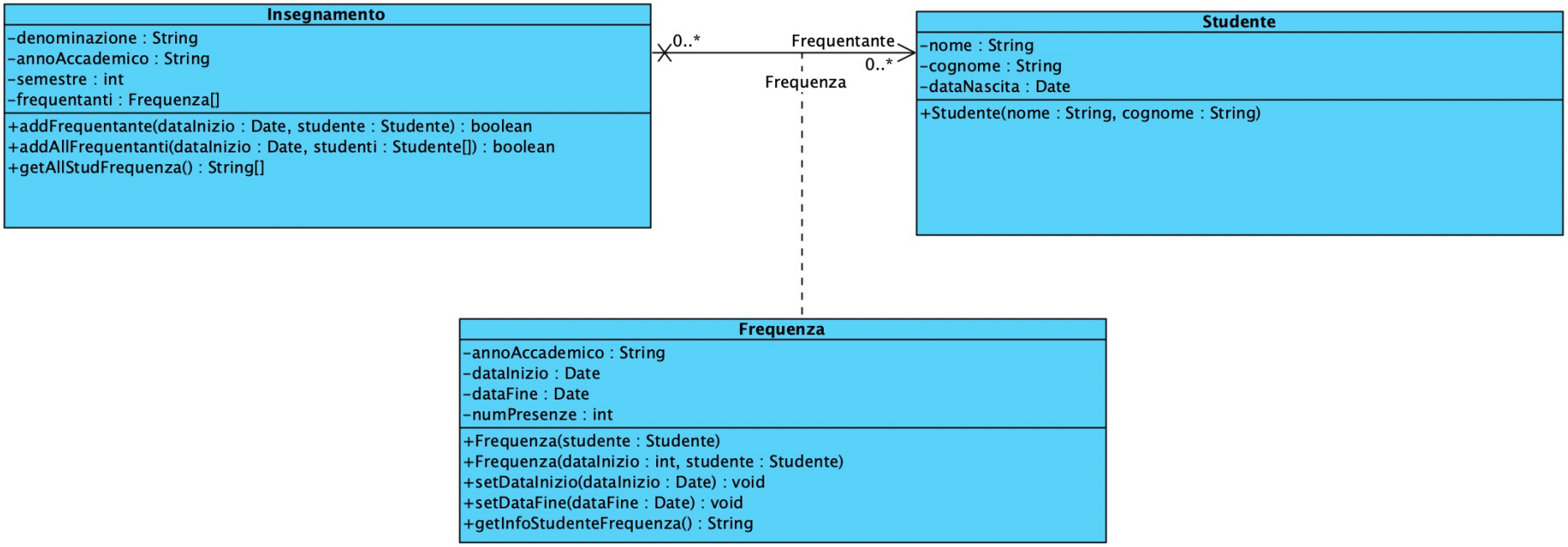
R4. Uno studente è caratterizzato da:
- un nome, un cognome e da una data di nascita

R5. Un insegnamento è caratterizzato da:
- una denominazione, da un anno accademico e da un semestre

R6. Uno studente che frequenta un insegnamento si dice frequentante
R7. Per ogni insegnamento presente il sistema permetterà di visualizzare l'elenco degli studenti frequentanti con le relative informazioni sulla frequenza

Come si modella?

Modellazione studente-insegnamento



Come si implementa?

Implementazione (corretta) della classe Insegnamento

```
class Insegnamento {
   private contenitoreFrequenza frequentanti;
   public boolean addFrequentante(data Inizio, Studente frequentante) {
     boolean risultato = false;
        if (NOT frequentanti.contiene(frequentante)) {
                Frequenza frequentanteStud = new Frequenza(datalnizio,Studente);
                frequentanti.add(frequentanteStud);
                risultato= true;
        }
     return risultato;
   }
   // realizzazione analoga a quella di addFrequentante
   public boolean addAllFrequentanti(datalnizio, ContainerStudente frequentanti);
}
```

Implementazione classe Frequenza

```
class Frequenza {
   private Studente studente;
   //istanzia un oggetto Studente
   Frequenza(Studente studenteF)
   Frequenza(Date dataInizio, Studente studenteF)
}
class Studente { // dichiarazione degli attributi nome, cognome, dataNascita}
```

DOMANDE:

- Per Frequenza si potrebbe definire un costruttore privo di un riferimento ad un oggetto Studente?
- Per Frequenza si potrebbe definire un'operazione setStudente(Studente studente)?

Una implementazione (errata) della classe Insegnamento

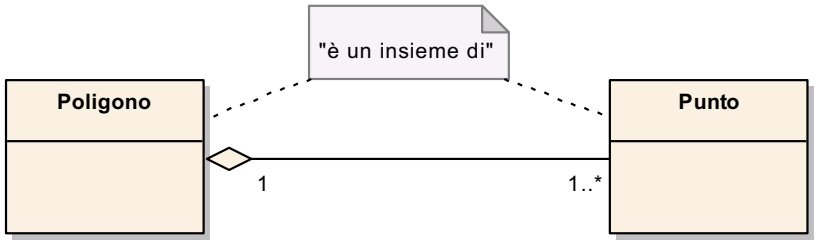
```
// Ipotesi: la modellazione prevede di fornire in input ad Insegnamento istanze di Frequenza
class Insegnamento {
   private contenitoreFrequenza frequentanti;
   public boolean addFrequenza (Frequenza frequentante);
   public boolean addAllFrequenze(ContainerFrequenza
   frequentanti);
}
```


DOMANDA:

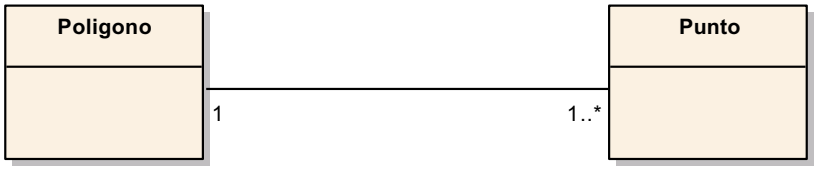
- Perché i metodi *addFrequenza()* ed *addAllFrequenze ()* *NON* sono corretti rispetto ai requisiti da soddisfare?

Relazione di aggregazione …

⇨relazione di associazione che definisce una relazione tra un "tutto" ed un insieme di parti di cui il "tutto" è costituito (*whole-part relationship*)



Tale rappresentazione è equivalente a



… Relazione di aggregazione

⇨ Proprietà

- ❑ gode delle proprietà transitiva ed antisimmetrica
- ❑ a *run-time* un oggetto contenuto sopravvive all'oggetto contenitore

⇨ Esempi

- ❑ autoveicolo composta da carrozzeria, pneumatico, …
- ❑ impresa composta da aree, unità, …
- ❑ pc composto da mouse, tastiera, monitor, piastra madre
- ❑ piastra madre composta da cpu, hard disk, floppy disk, …

Aggregazione: quando usarla?

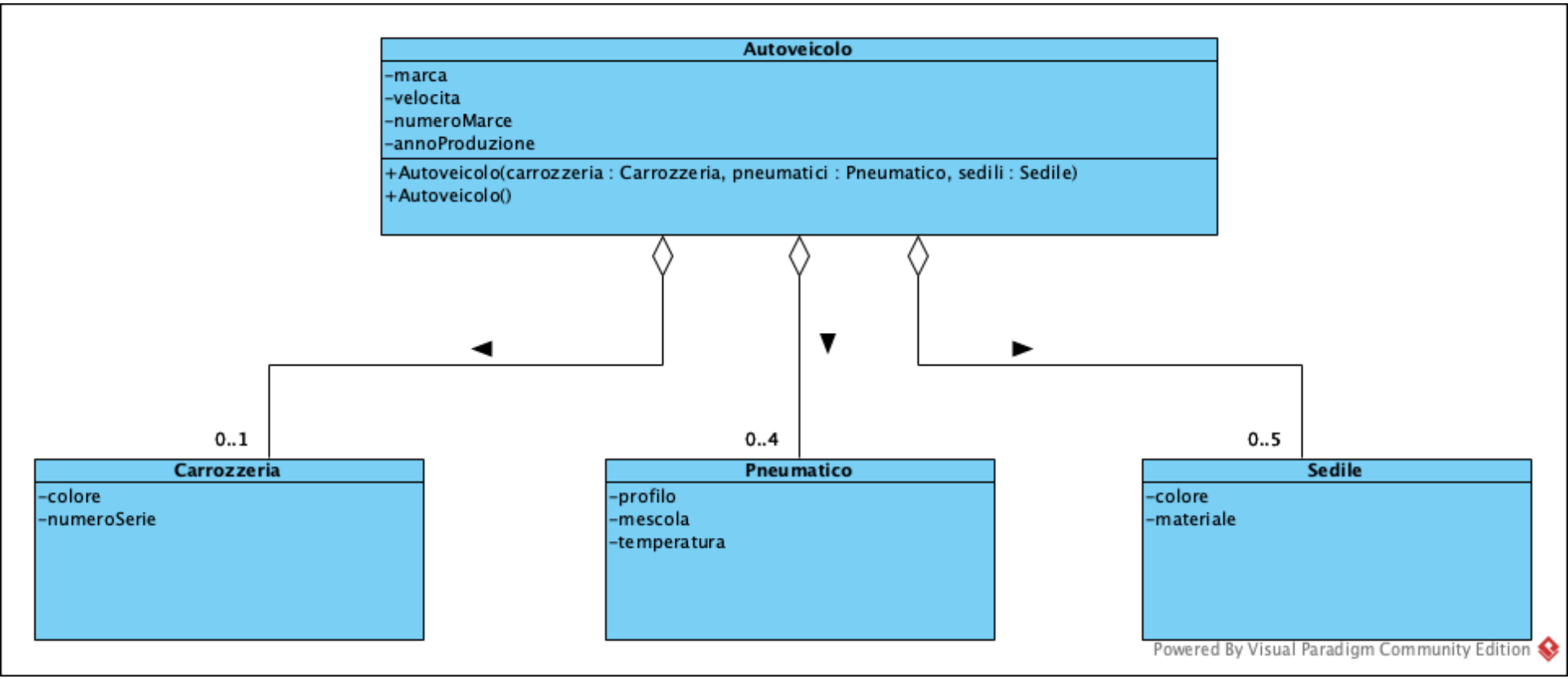
Grady Booch suggerisce l'uso dell'aggregazione nelle seguenti situazioni:

1. Contenimento fisico: la pagina di un libro.
2. Appartenenza (*membership*): il giocatore di una squadra di calcio
3. Composizione funzionale: le ruote di un'automobile.

Osservazione: l'aggregazione non implica una dipendenza esistenziale:

- un'automobile può essere distrutta (rottamata) ma alcune sue parti possono essere riutilizzate
- una squadra di calcio può fallire, ma i suoi giocatori non vengono "soppressi"

Esempio di aggregazione



Aggregazione: implementazione

Per realizzare l'aggregazione è necessario soddisfare **tutte** le seguenti regole

- Definire degli attributi (privati) del tipo delle classi aggregate o, in alternativa, definire un'opportuna struttura di dati, privata, attraverso la quale mantenere le istanze aggregate
  - La dimensione di ogni attributo dipende dalla molteplicità
- Ottenere i riferimenti alle istanze aggregate e mapparle con i corrispondenti attributi privati mediante i *costruttori* e/o metodi *add(), addAll() o set()*
- Definire, se previsto dalle specifiche, delle operazioni per esportare i riferimenti delle istanze aggregate, metodi *get()* e *getAll()*

Aggregazione: implementazione autoveicolo

```
class Autoveicolo {
   // omissis attributi

   public Autoveicolo();

   public Autoveicolo(Carrozzeria carrozzeria, ContainerSedile[0..5]

   sedili, ContainerPneumatico[0..4] pneumatici)

   public boolean addSedile(Sedile sedile);

   public boolean addAllSedili(ContainerSedile[0..5] sedili) ;

   // omissis altre operazioni

}
```

Relazione di composizione

- Relazione di aggregazione in cui gli oggetti contenuti (componenti) non hanno vita propria ma esistono in quanto parte della classe contenente (composta)
- Proprietà
  - la classe composta è responsabile della creazione e della distruzione degli oggetti contenuti
  - la classe composta è l'unica a poter usare le classi componenti
  - la molteplicità dal lato della classe composta deve essere al più uguale ad uno, mentre può essere qualsiasi per le classi componenti

Composizione: Regola di non condivisione

Regola di "non condivisione": benché una classe possa essere componente di molte altre classi, ogni sua istanza può essere componente di un solo oggetto

Un'istanza di *Punto* può essere parte di un poligono oppure il centro di un cerchio, ma non entrambe le cose.

- Un diagramma delle classi può mostrare più classi di potenziali possessori di oggetti componenti, ma ogni istanza di componente deve appartenere a un solo oggetto possessore.
- Questa regola è caratterizzante della composizione.

Implementazione Composizione: regole di base

Per tradurre tale relazione si devono rispettare **tutte** le seguenti:

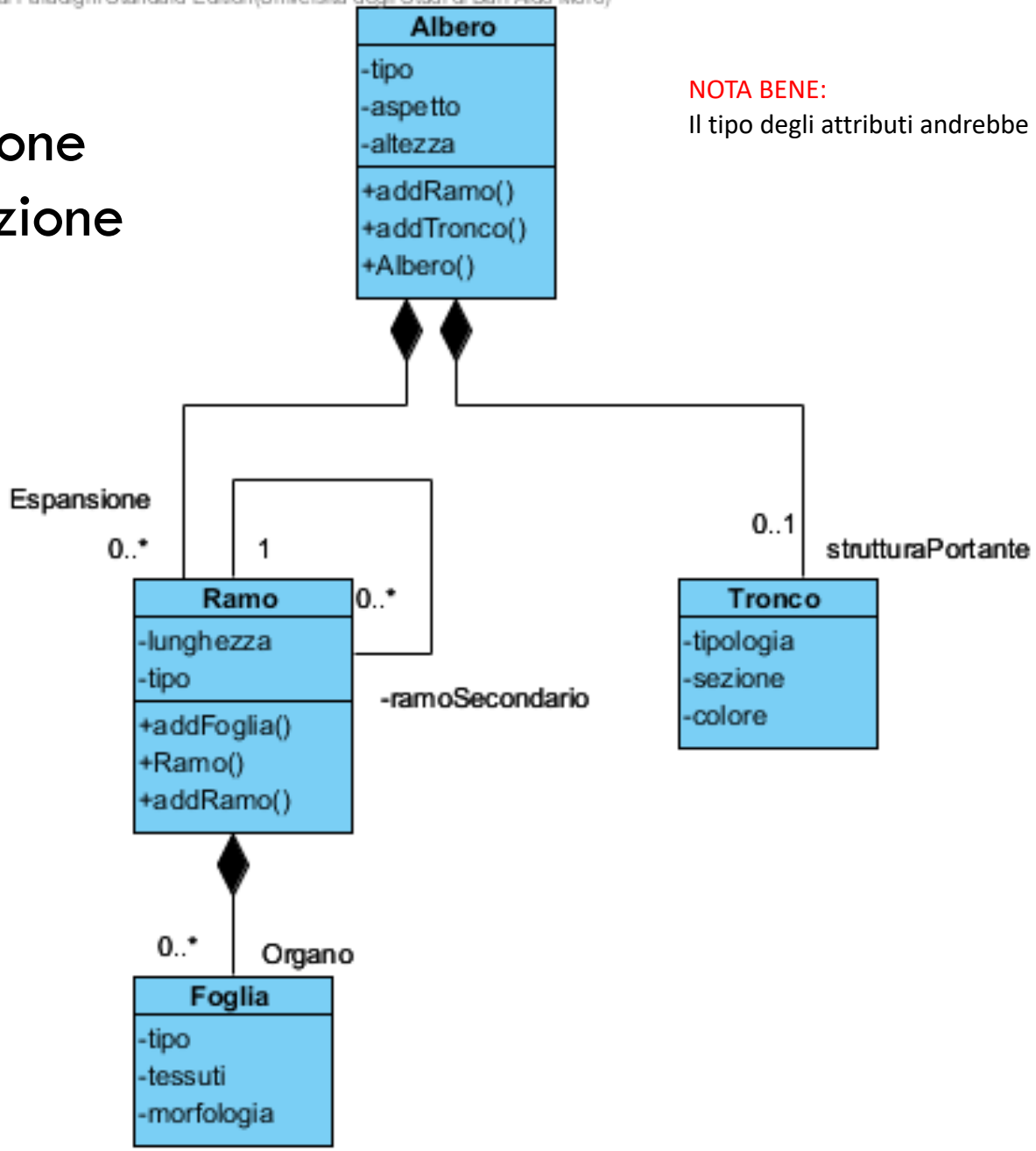
- definire degli attributi (privati) del tipo delle classi componenti o, in alternativa, definire un'opportuna struttura di dati, sempre privata, attraverso la quale mantenere i riferimenti alle istanze delle classi componenti
  - La dimensione di ogni attributo dipenderà dalla molteplicità
- istanziare le classi componenti all'interno della classe composta mediante i *costruttori*
- Gli oggetti componenti non devono essere mai condivisi con oggetti di altri classi
  - componenti privati e classe composta priva di alcun metodo che restituisca alcun riferimenti delle classi componenti
- deallocare all'interno della classe composta mediante distruttore, o un suo sostitutivo, le classi componente

Implementazione Composizione: UML

⇨ Osservazioni

- UML **non** definisce né l'ordine né il modo in cui le istanze componenti di una composizione siano create
- la navigabilità dipende dalle specifiche
- Precisazione: in alcuni linguaggi OO il distruttore non esiste (e.g. in Java)

Notazione composizione



NOTA BENE:
Il tipo degli attributi andrebbe sempre specificato

Composizione: implementazione *Albero* …

```
class Albero {
    //in Java sarebbe necessario specificare il tipo di ogni variabile
    private tipo, aspetto;
    private altezza;
    private Tronco strutturaPortante; // composizione
    private ContenitoreRamo[0..*] espansione; // composizione

    // costruttore di Albero
    public Albero();
    // crea un Ramo e lo aggiunge nel contenitore espansione
    // non deve mai ricevere in ingresso un oggetto di tipo Ramo
    public boolean addRamo();
    // crea un Tronco e lo associa all'albero corrente;
    // non deve mai ricevere in ingresso un oggetto di tipo Tronco
    public boolean addTronco();
}
```

Alcuni quesiti sulla classe Albero

- Perchè la seguente operazione è concettualmente errata?

  ```
  public Tronco getTronco();
  ```

- Quali operazioni potrebbe eseguire il costruttore Albero()?
- L'operatore addRamo() su quale variabile agisce?
- Sarebbe possible aggiungere un operatore *addRamo*, oltre a quello esistente, che abbia un parametro formale per specificare la lunghezza del ramo?
- L'assenza del costruttore nel diagramma quali informazioni veicola all'implementatore della classe?

… Composizione: implementazione *Albero*

```
class Ramo {
    //in Java sarebbe necessario specificare il tipo di ogni variabile
    private lunghezza, tipo;
    // ramiSecondari esempio di relazione riflessiva
    private ContenitoreRamo[0…*] ramiSecondari;
    private ContenitoreFoglia[0…*] organo;
    //omissis altri attributi
    public boolean addFoglia();
    public Ramo();
    public boolean addRamoSecondario(int lunghezza);
// altre operazioni tra le quali ci saranno, sicuramente:
// setLunghezza(lunghezza) e setTipo(tipo)
}
```