DIAGRAMMA DELLE CLASSI - CONCETTI FONDAMENTALI

LEZIONE 12
05/11/2025
VINCENZO RICCIO

RIFERIMENTI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Fowler - Capitolo 3, 5
- Sommerville - Capitolo 5.3

DEFINIZIONE

DIAGRAMMA UML DELLE CLASSI

- Diagramma **strutturale**
- Un Diagramma UML delle Classi mostra una vista **statica** del modello, ossia mostra l'organizzazione del progetto del sistema ma non mostra informazioni temporali
- I suoi elementi corrispondono a concetti fondamentali del paradigma object-oriented (classi, oggetti, ereditarietà)
- Diagramma UML più diffuso

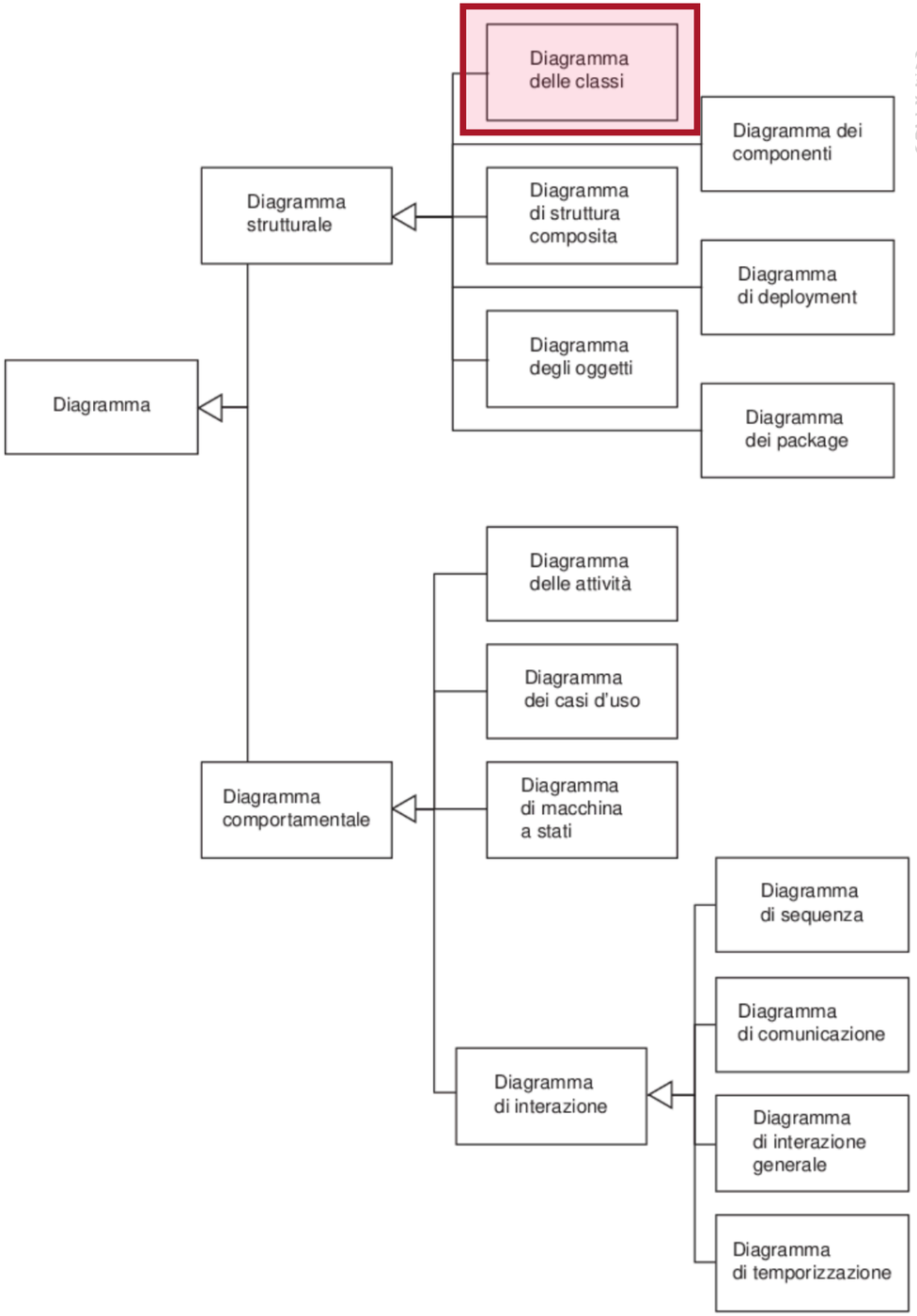


DIAGRAMMA UML DELLE CLASSI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Rappresenta:
  - I tipi di oggetti (le classi) del sistema che corrispondono alle entità che esistono nel sistema
  - Le relazioni <u>statiche</u> tra classi ed i vincoli che si applicano a tali relazioni
  - Le caratteristiche delle classi: proprietà e responsabilità
- Le proprietà corrispondono agli attributi e le responsabilità ai metodi

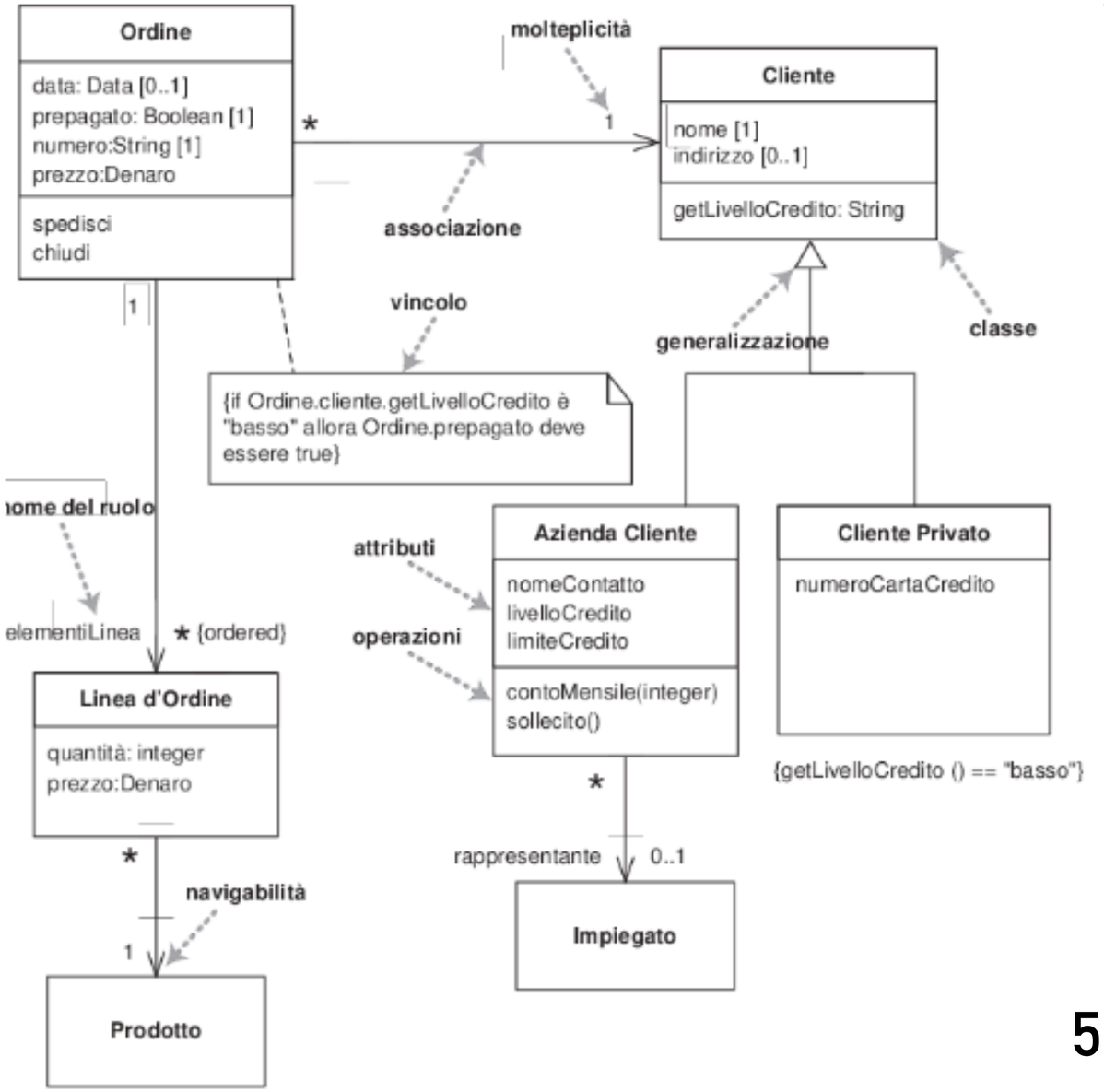


DIAGRAMMA DELLE CLASSI NEL PROCESSO SOFTWARE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Può essere utilizzato nella fase di definizione dei requisiti
- Consente una vista concettuale delle **entità** nel dominio del problema
- Cattura i concetti principali del dominio e fissa i confini del sistema (cosa fa parte e cosa non fa parte del sistema)
- Semplice modello architetturale rivolto alla comprensione di cosa fa il sistema, non come lo fa

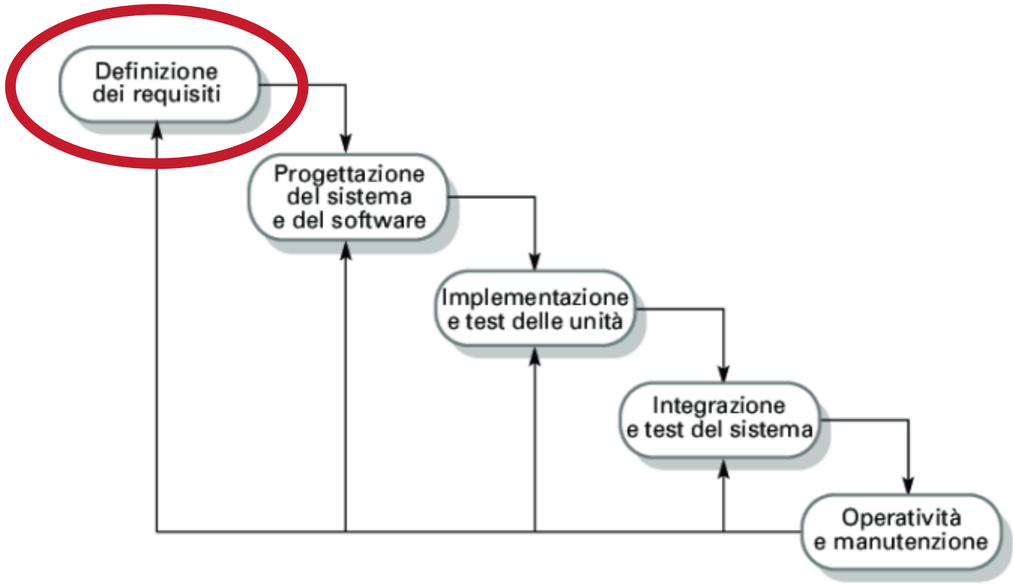


DIAGRAMMA DELLE CLASSI NEL PROCESSO SOFTWARE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Può essere utilizzato nella fase di progettazione
- Identifica le **entità** del sistema e le relazioni tra esse
- Anche in questo caso, prescinde dall'implementazione

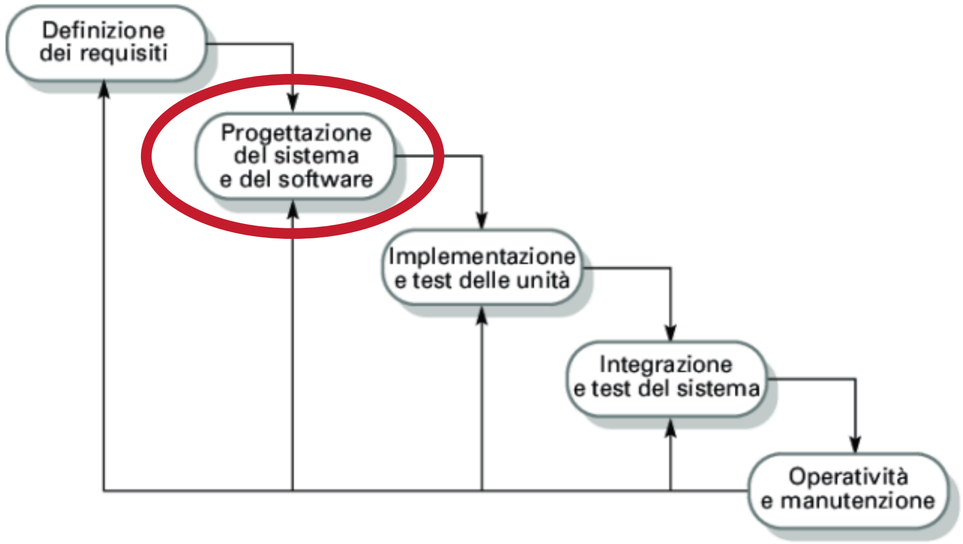
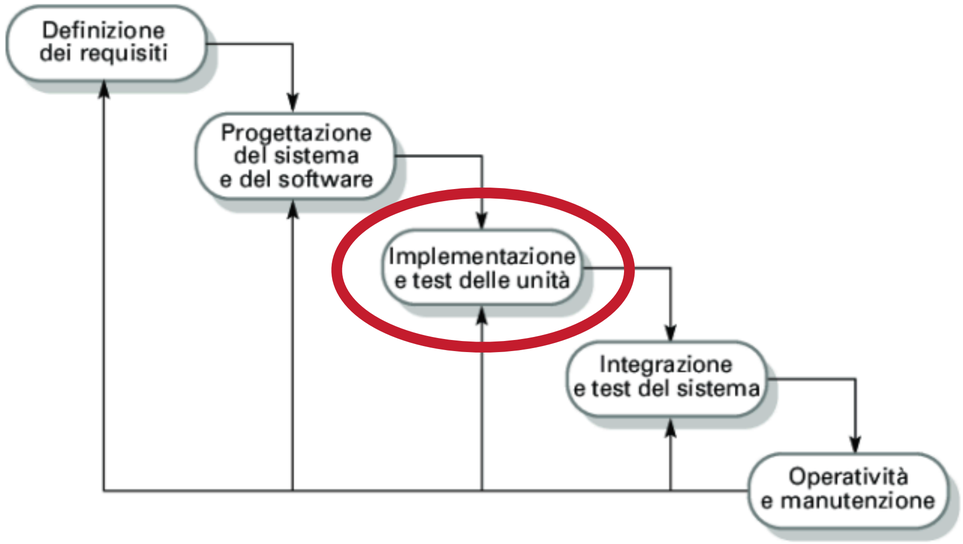


DIAGRAMMA DELLE CLASSI NEL PROCESSO SOFTWARE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Può essere utilizzato nella fase di implementazione
- Identifica le **classi** del software e le relazioni tra esse
- Corrisponde alla reale struttura del software poiché ne modella l'implementazione object-oriented



DIAGRAMMI DELLE CLASSI NEL PROGETTO DI SOFTWARE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

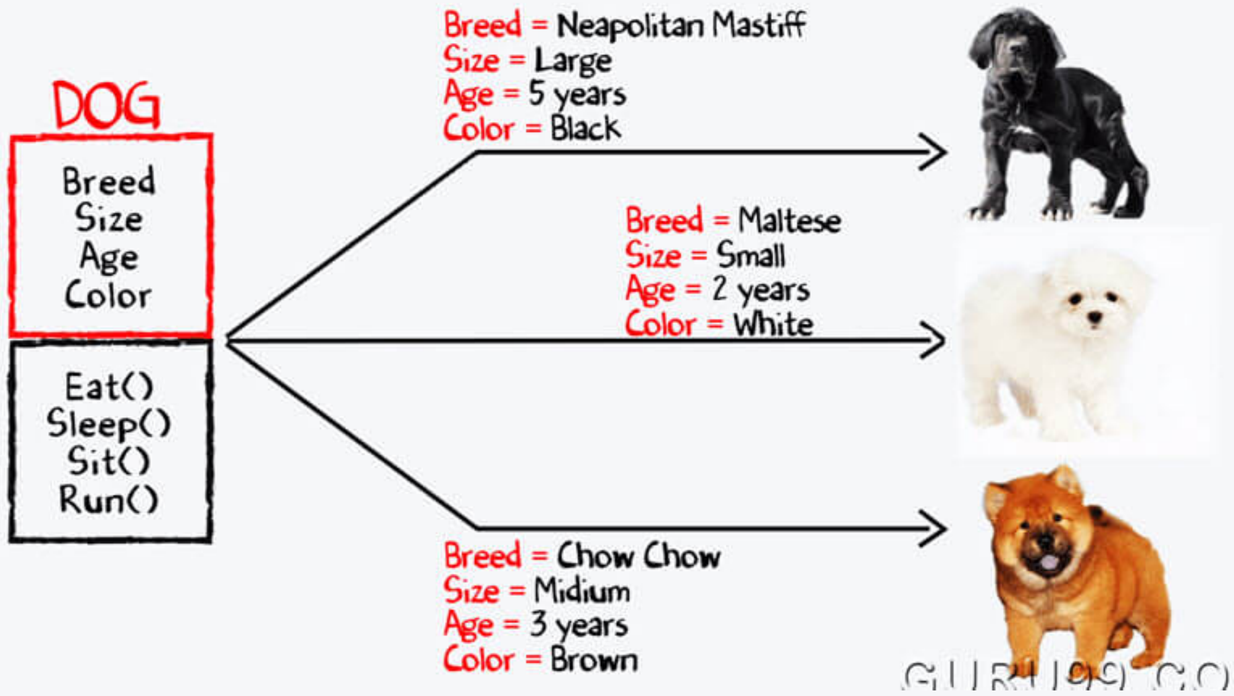
- I diagrammi delle classi si concentrano solo sulla struttura (classi, attributi, relazioni) e trascurano il comportamento del sistema
- Affianca i diagrammi delle classi con diagrammi comportamentali (es. diagrammi di sequenza, diagrammi di attività) per rappresentare come il sistema si comporta in scenari reali
- Un buon approccio di modellazione prevede il passaggio costante tra diagrammi di struttura e di comportamento, migliorando la comprensione globale del sistema

SINTASSI

CLASSE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Descrittore di un insieme di oggetti con le stesse proprietà (attributi), comportamento (operazioni) e relazioni con altri oggetti
- Stesso concetto della programmazione OO

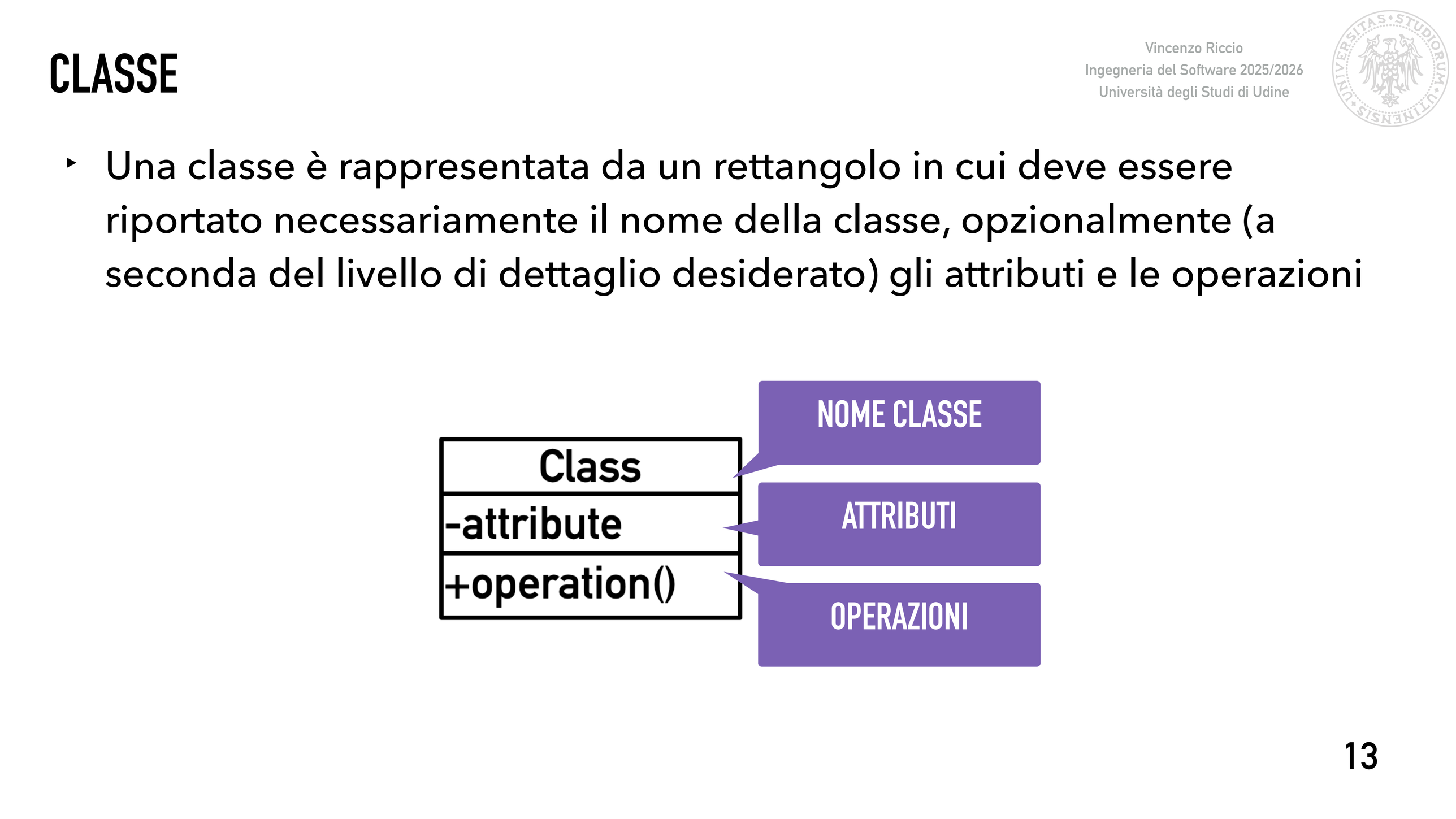


CLASSE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Ogni oggetto è un'istanza di una sola classe
- In un dato istante, un oggetto ha uno specifico stato (valore dei suoi attributi)
- In base al proprio stato, due oggetti possono rispondere diversamente alla stessa operazione
- Ad esempio, se si tenta di prelevare 100 Euro da un oggetto conto corrente, il risultato sarà diverso a seconda della disponibilità

CLASSE

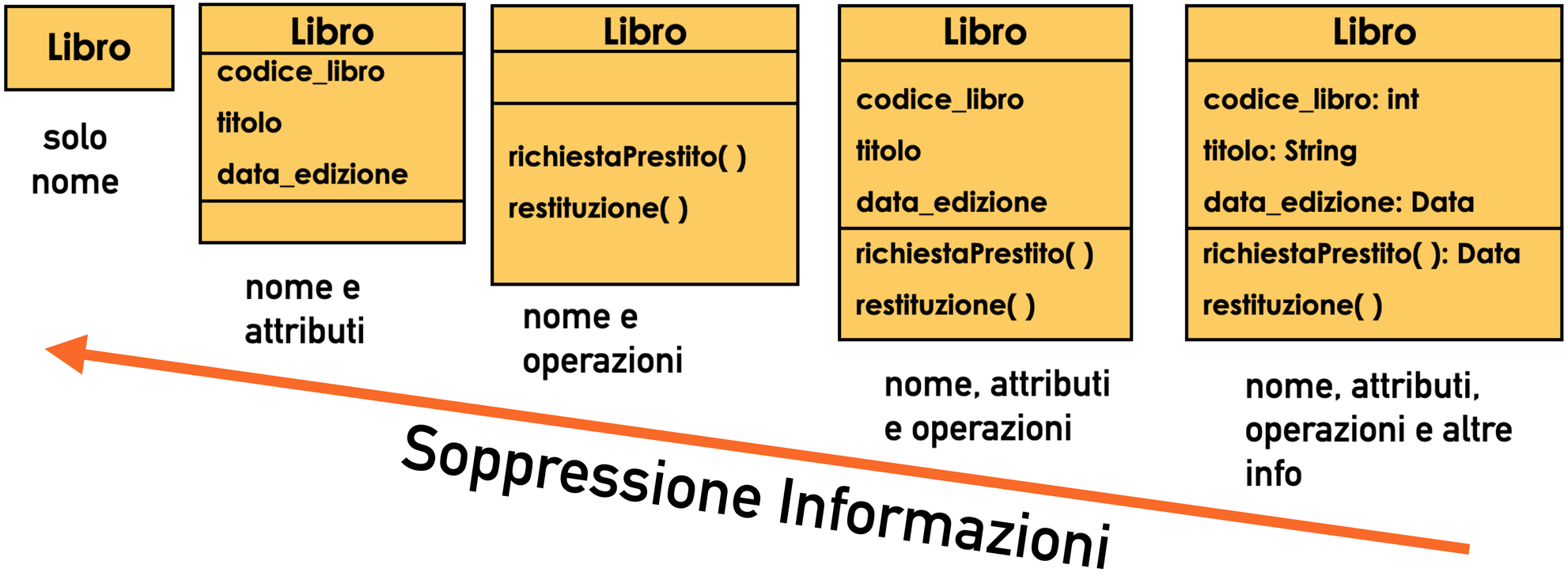


- Una classe è rappresentata da un rettangolo in cui deve essere riportato necessariamente il nome della classe, opzionalmente (a seconda del livello di dettaglio desiderato) gli attributi e le operazioni

CLASSE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Una classe è rappresentata da un rettangolo con tre diverse sottosezioni: nome, attributi, operazioni.
- Il nome è al **singolare** e solitamente scritto in **UpperCamelCase**

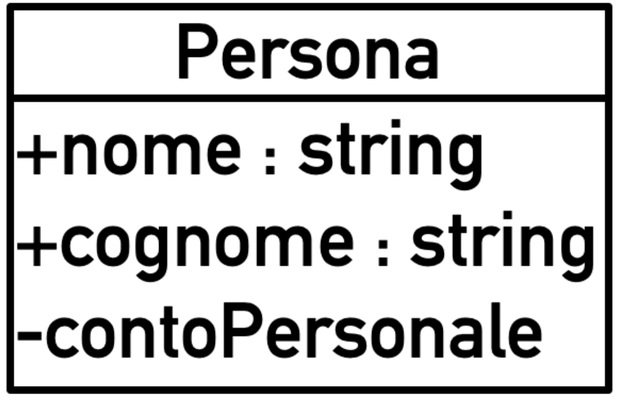


ATTRIBUTI: VISIBILITÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** molteplicità: tipo = valoreDefault {proprietà}

- Sono ammessi tre livelli di **visibilità** degli attributi:
  + pubblico: l'accesso è esteso a tutte le altre classi
  # protetto: l'accesso è consentito soltanto alle classi che derivano dalla classe originale
  - privato: soltanto la classe originale può accedere a tali attributi



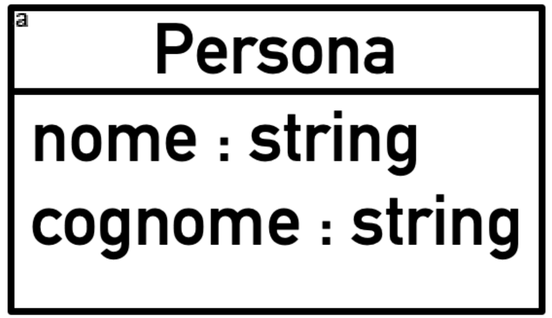
ATTRIBUTI: NOME E TIPO

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** molteplicità: tipo = valoreDefault {proprietà}

- Il **nome** dell'attributo è l'unico parametro obbligatorio
- Il **tipo** può essere un tipo un tipo primitivo (int, double, char, etc…), oppure il nome di una classe definita nello stesso diagramma (in tal caso l'attributo può anche essere indicato con un'associazione). Definisce un vincolo sugli oggetti che possono corrispondere all'attributo

nome: string

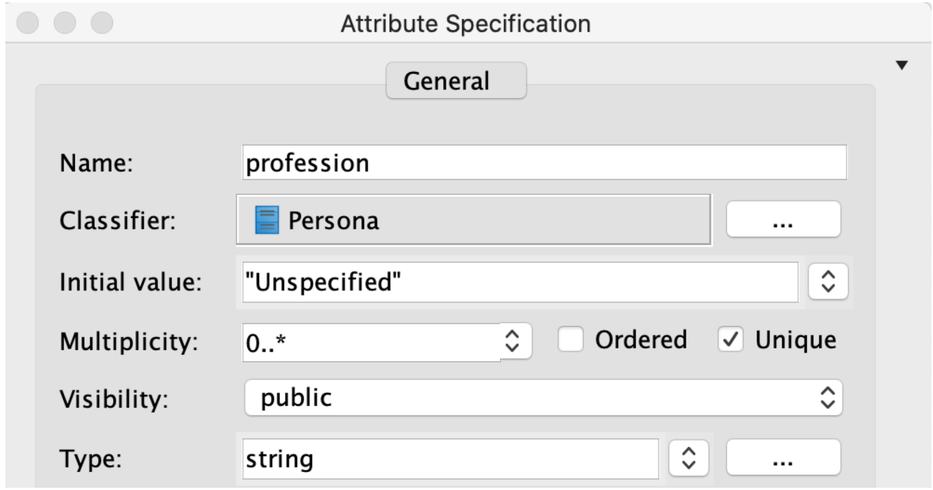


ATTRIBUTI: MOLTEPLICITÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** molteplicità: tipo = valoreDefault {proprietà}

- Indica il quantitativo degli attributi di un certo tipo. Permette di indicare attributi come array o matrici. Il valore di default è 1
- Il numero minimo e massimo possono essere racchiusi tra parentesi quadre quando il tipo è semplice: [ ]
  - **[1..1]** o **[1]** esattamente 1
  - **[0..1]** al più uno
  - **[0.. *]** o **[*]** numero imprecisato
  - **[1..*]** almeno 1

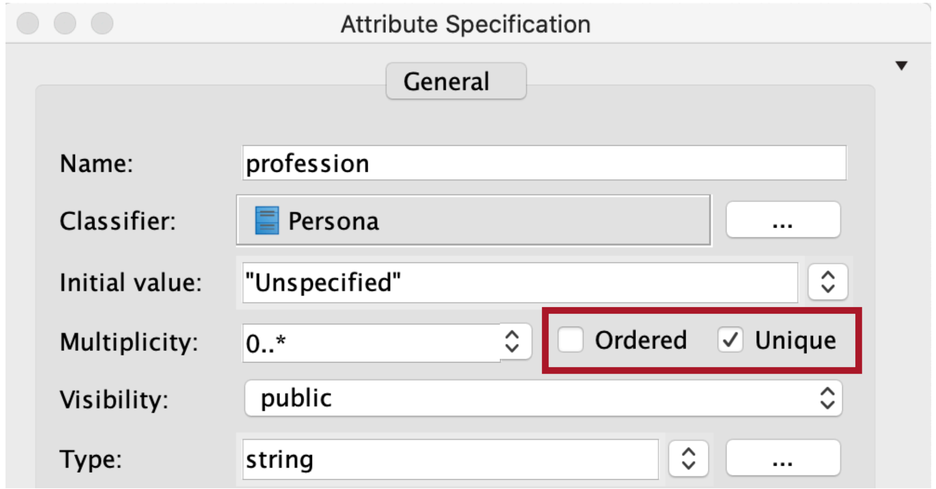


ATTRIBUTI: MOLTEPLICITÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** molteplicità: tipo = valoreDefault {proprietà}

- Gli elementi di una molteplicità a più valori sono considerati come un insieme (e quindi sono non ordinati e non vi sono elementi duplicati)
- Se sono dotati di ordine si usa la notazione **{ordered}**
- Se sono possibili valori duplicati si aggiunge la notazione **{nonunique}**



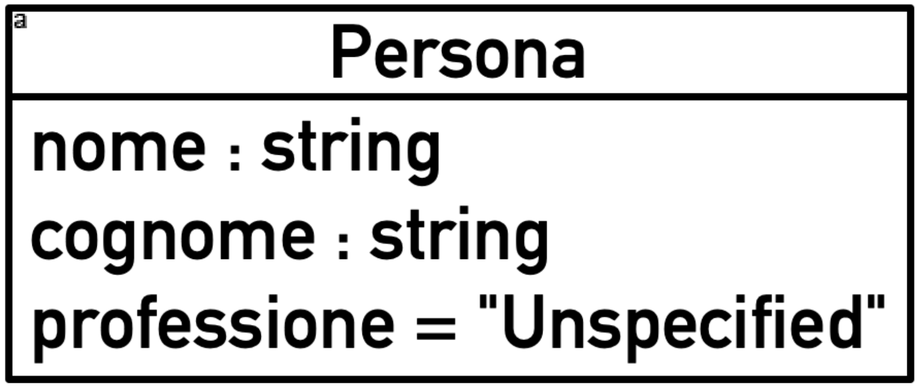
ATTRIBUTI: VALORE DI DEFAULT E PROPRIETÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** molteplicità: tipo = valoreDefault {proprietà}

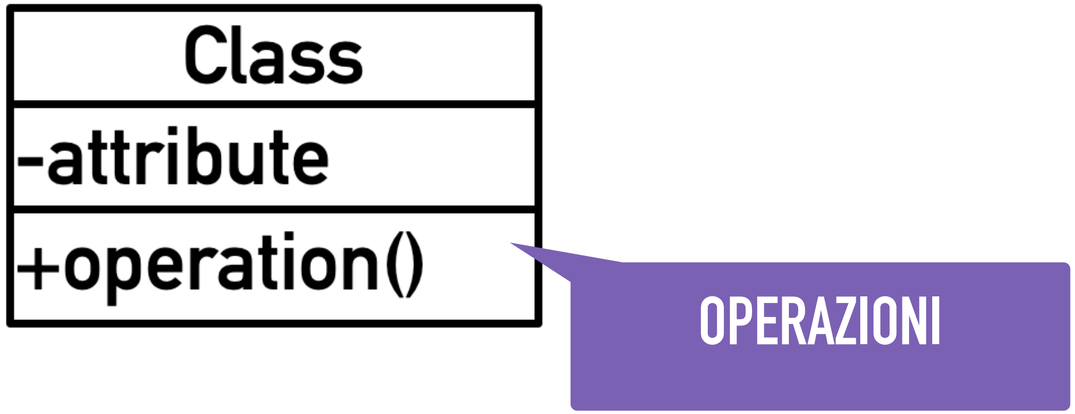
- **valoreDefault** è il valore assegnato all'attributo di default, se nessun valore viene specificato durante la creazione
- **{proprietà}** indica caratteristiche aggiuntive dell'attributo (ad esempio la sola lettura)

Esempio: name: String [1] ="Untitled" {readOnly}



OPERAZIONI

- Corrispondono ai metodi di una classe ossia ad operazioni invocabili sugli oggetti istanza della classe
- Sono le azioni che possono essere svolte da una classe di oggetti
- Solitamente, nei diagrammi non sono riportate le operazioni che si occupano soltanto di modificare gli attributi, poiché queste sono facilmente deducibili

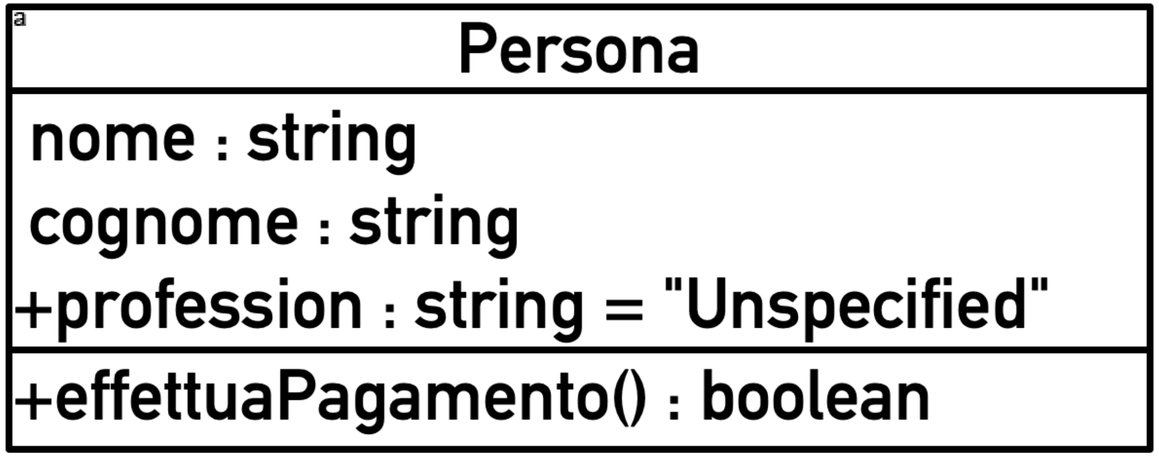


OPERAZIONI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

visibilità **nome** (listaParametri) : tipoRestituito {proprietà}

- Operazioni invocabili sugli oggetti istanza della classe
- **visibilità** e il **nome** analoghi agli attributi
- **tipoRestituito** é il tipo del valore di ritorno



OPERAZIONI

visibilità **nome** (listaParametri) : tipoRestituito {proprietà}

- **listaParametri** contiene nome e tipo dei parametri nella forma:

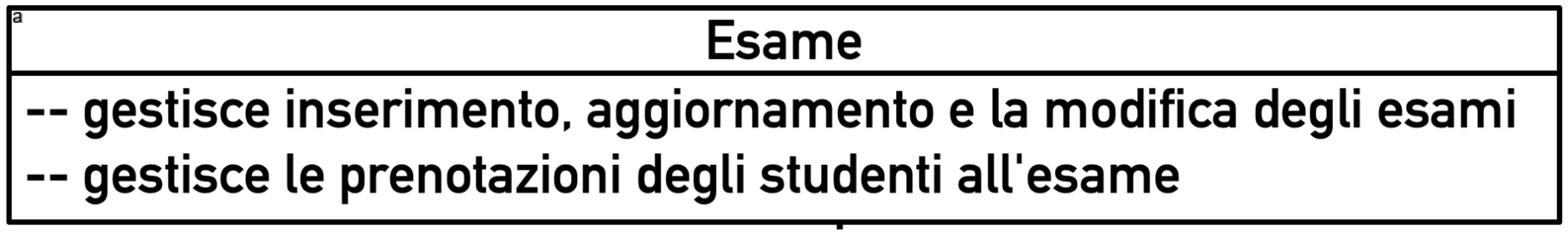
direzione nome parametro: tipo = valoreDefault

- **direzione:** input (in), output (out) o entrambi (inout). Il valore di default é in.
- **nome**, **tipo** e **valoreDefault** sono analoghi a quelli degli attributi
- Esempio: + saldoAllaData(in giorno: Date): float

RESPONSABILITÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

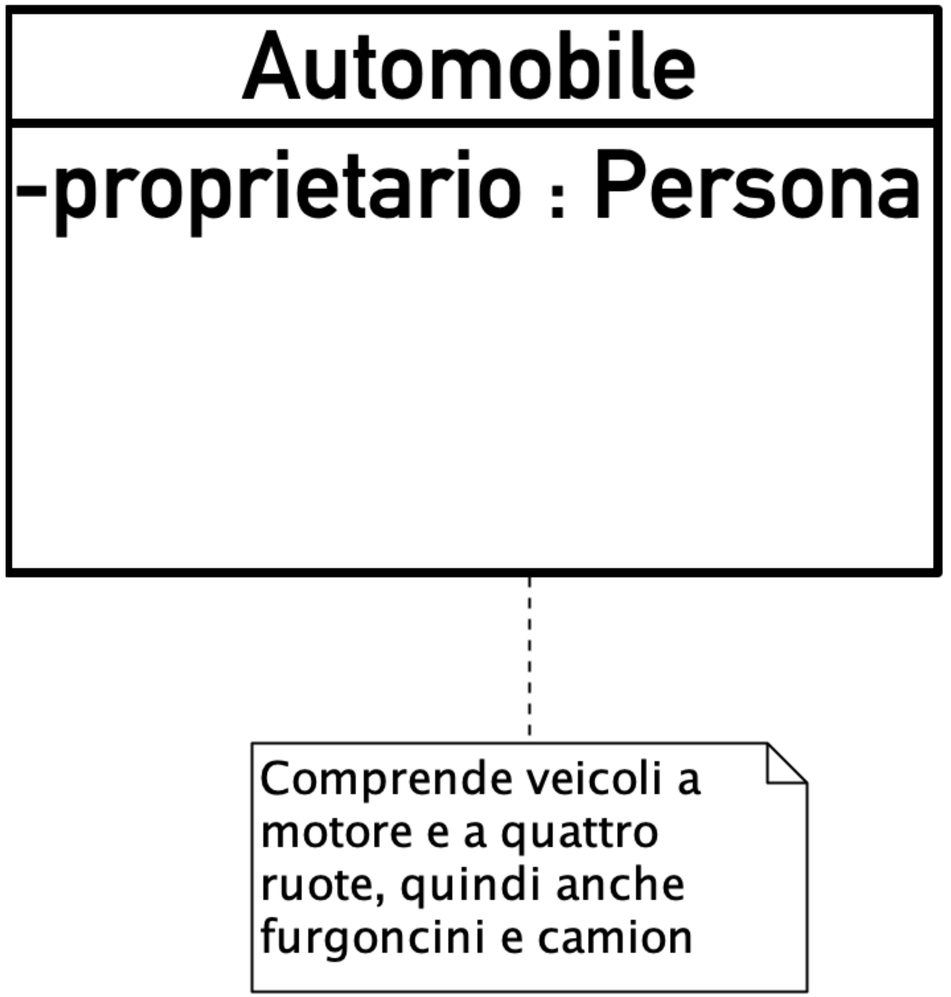
- Nei diagrammi concettuali che descrivono il dominio del sistema non si inseriscono di solito metodi perché essi rappresenterebbero un elemento del dominio della soluzione
- In alternativa si possono utilizzare le **responsabilità**: insieme di funzioni principali che la classe dovrebbe garantire, utili per controllare la completezza del modello di dominio
- Sono riportate come stringhe di commento (precedute da - -) nel riquadro delle operazioni



NOTE E TESTO DESCRITTIVO

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Commenti aggiuntivi in linguaggio naturale possono essere utilizzati in ogni tipo di diagramma UML
- Simile a un commento in un linguaggio di programmazione
- Possono essere collegati all'elemento a cui fanno riferimento tramite una linea tratteggiata oppure fluttuare senza collegamenti
- Ad esempio: Utilizzato all'esame per spiegare scelte progettuali



RELAZIONI TRA CLASSI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

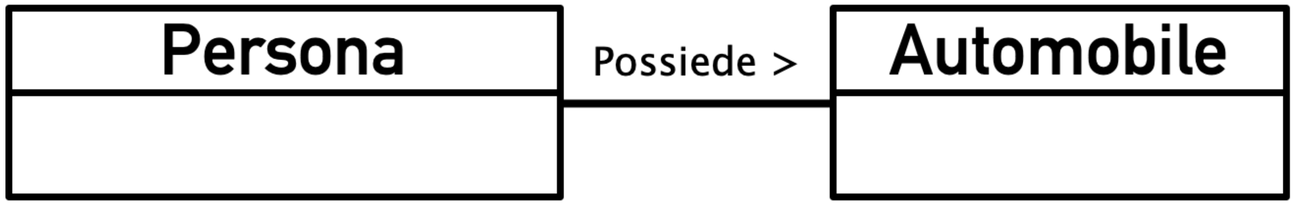
- Le classi in un diagramma delle classi non sono in generale isolate, ma in relazione con altre entità con legami di varia natura, detti “relazioni”
- Le principali relazioni tra le classi sono:
  - La relazione di **associazione**
  - La relazione di **generalizzazione-specializzazione**
  - La relazione di **contenimento**

ASSOCIAZIONI

ASSOCIAZIONI

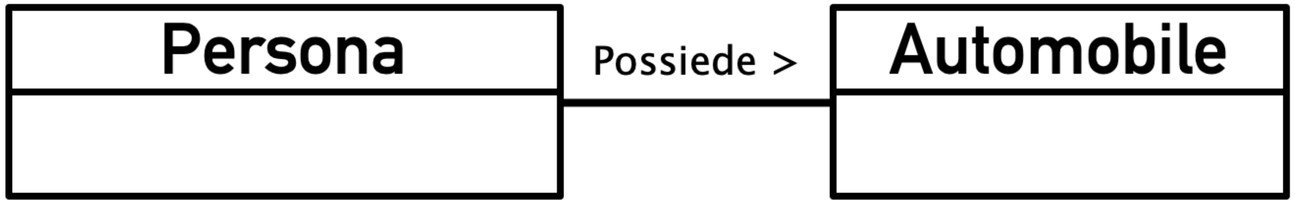
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- L'associazione tra due classi esprime una relazione (fisica o concettuale) tra le istanze (oggetti) delle classi
- Un'associazione è caratterizzata da:
  - un **nome**: esprime il legame semantico tra le classi associate
  - un eventuale **ruolo** giocato da ciascuna delle parti associate
  - la **molteplicità** dell'associazione
  - Il **verso di navigazione** dell'associazione
- Esempio: Ogni Automobile ha per proprietario una Persona



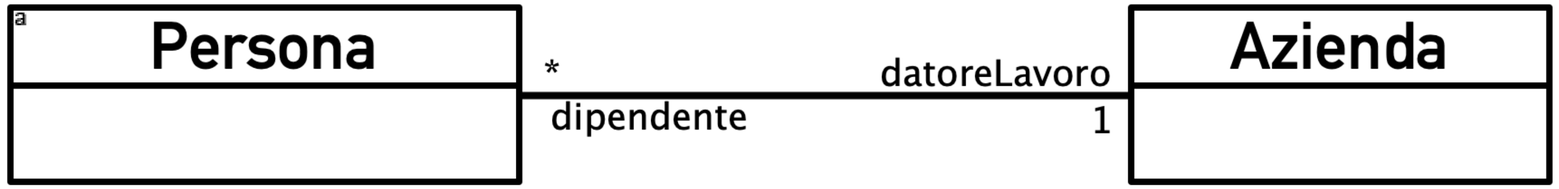
ASSOCIAZIONI: NOME

- Il **nome** è un'etichetta dell'associazione, solitamente è un verbo
- Il nome dell'associazione permette di formare frasi di senso compiuto (*"Una persona possiede un'automobile"*)
- Per evitare ambiguità è possibile specificare la direzione dell'associazione (ad esempio con il simboli < o >)



ASSOCIAZIONI: RUOLO

- È possibile specificare il **ruolo** che gli oggetti di una classe rivestono quando sono collegati da istanze dell'associazione
- Il ruolo è specificato sull'estremità dell'associazione prossima alla classe
- Usato spesso in alternativa al nome, usare entrambi appesantirebbe la notazione



- Nell'esempio:
  - Persona è dipendente dell'Azienda
  - L'Azienda è datore di lavoro della Persona

ASSOCIAZIONI: RUOLI E NOMI

- Non tutte le associazioni necessitano di un nome: assegnalo solo se migliora la chiarezza e la comprensione del modello
- Evita nomi generici come "has" o "is related to", che non aggiungono informazioni utili
- Quando due classi sono coinvolte in associazioni diverse tra le stesse classi, per distinguerle è opportuno riportare un nome dell'associazione oppure il ruolo delle classi nelle diverse associazioni

ASSOCIAZIONI: MOLTEPLICITÀ

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

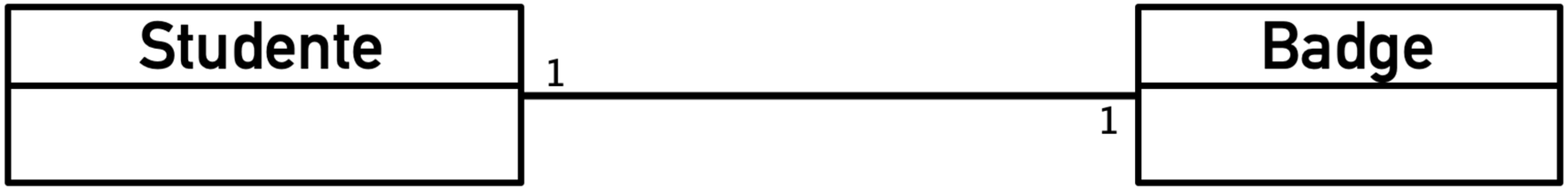
- La molteplicità vincola il numero di oggetti di una classe che possono partecipare ad una associazione in ogni istante
- Sono riportate sull'estremità dell'associazione prossima alla classe nella forma: **minimo..massimo**
  - **0..1** Zero (partecipazione opzionale) o uno
  - **1** Esattamente 1
  - **0..*** oppure soltanto ***** Zero o più (non c'è limite superiore)
  - **1..*** Uno (partecipazione obbligatoria) o più
  - **1..6** Da 1 a 6

ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Ciascun badge è utilizzato per identificare uno e un solo studente

- Sappiamo per certo che in quest'associazione uno studente ha molteplicità 1 nell'associazione
- Non è specificato ciascuno studente quanti badge può possedere. Se ipotizziamo 1, abbiamo un'associazione 1 a 1
- **Uno ad uno:** uno studente può possedere un solo badge in un dato istante. Un badge è posseduto da un solo studente.

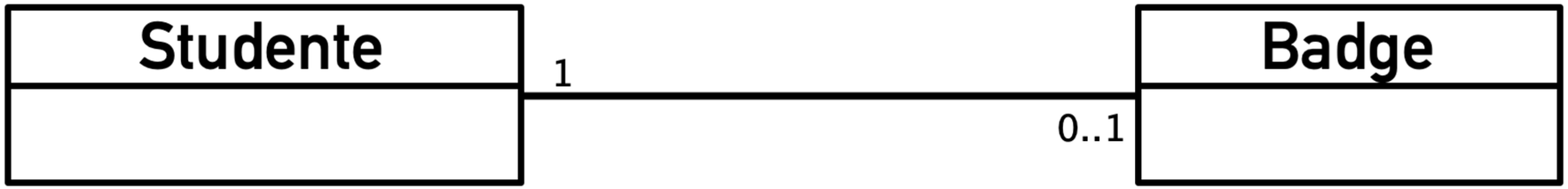


ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Ciascun badge è utilizzato per identificare uno e un solo studente

- Se ad esempio vogliamo modellare il caso di smarrimento o di attesa di rilascio dopo la prima iscrizione
- **Uno ad al più uno (0..1):** uno studente può possedere nessuno o al massimo un solo badge in un dato istante. Un badge è posseduto da un solo studente.

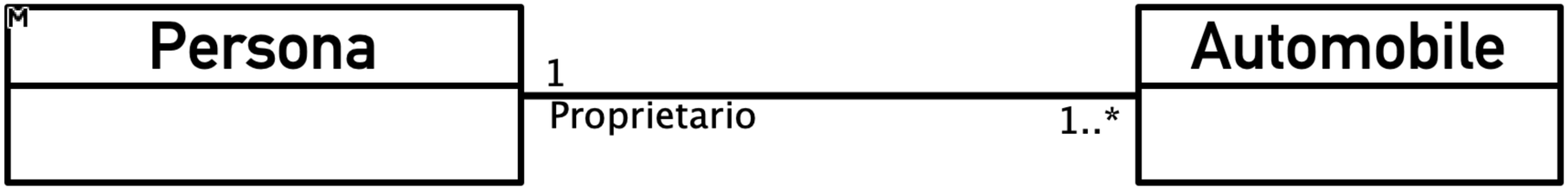


ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Una Persona possiede almeno un'Automobile. Un'Automobile può essere posseduta da una e una sola Persona

- La modellazione del problema non rappresenta tutti i casi presenti nel mondo reale:
  - Le persone che non possiedono Automobile non fanno parte del problema
  - Non è nel problema in oggetto il mantenimento di informazioni riguardo i proprietari di automobili di seconda, terza mano, etc.
- **Associazione uno a molti:** una persona, una o più automobili

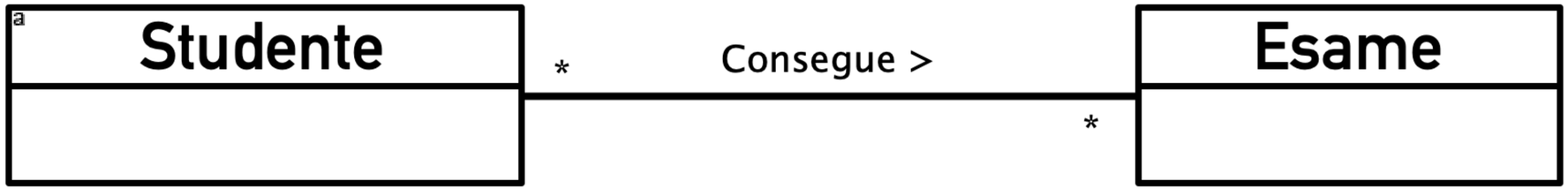


ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Uno Studente può conseguire più Esami. Ciascun Esame può essere conseguito da più Studenti.

- Uno studente può conseguire potenzialmente illimitati esami. Un esame può essere conseguito da un numero non limitato di studenti
- Possono esserci studenti che non hanno conseguito esami. Possono esserci esami non conseguiti (ancora, in un dato istante) da alcuno studente
- **Associazione molti a molti:** molti studenti, molti esami

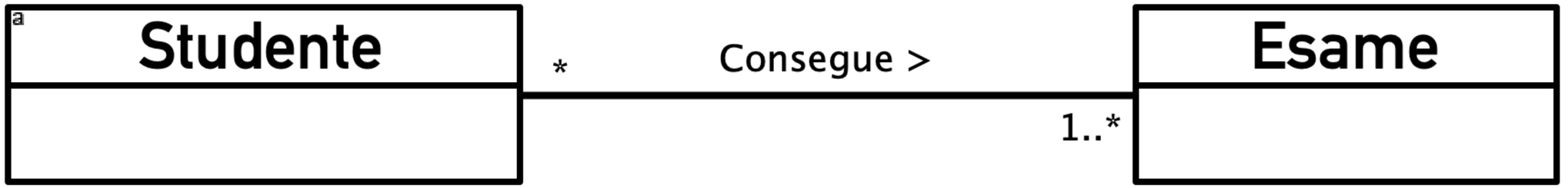


ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Uno Studente può conseguire più Esami. Ciascun Esame può essere conseguito da più Studenti.

- Se in alternativa, avessimo voluto considerare il caso in cui uno studente era considerato dal sistema soltanto dal momento del conseguimento del primo esame, avremmo avuto che la molteplicità di Esame 1..* poiché lo studente deve aver conseguito almeno un esame:

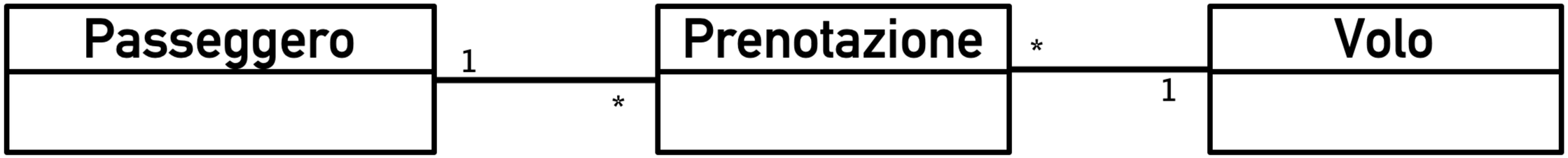


ASSOCIAZIONI: MOLTEPLICITÀ – ESEMPI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

Una prenotazione si riferisce sempre ad uno e un solo passeggero. Un Passeggero può avere più Prenotazioni. Una Prenotazione si riferisce ad un Volo. Un Volo può avere più Passeggeri prenotati.

- Prima di creare una prenotazione devono esistere passeggero e volo
- Il passeggero può essere memorizzato nel sistema anche prima di effettuare una prenotazione: ci possono essere passeggeri senza prenotazione



ASSOCIAZIONI: MOLTEPLICITÀ – DETTAGLI E CONSIGLI

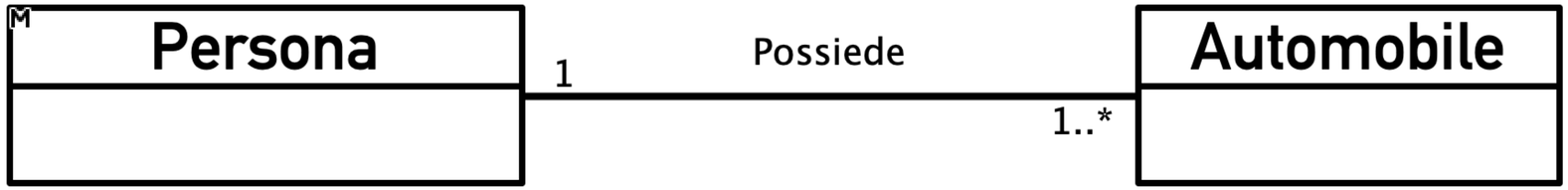
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- **Molteplicità discontinue:** UML 1 consentiva molteplicità discontinue, come 2, 4 (es. due o quattro persone per auto). UML 2 ha eliminato le molteplicità discontinue perché non erano molto comuni e appesantiscono la notazione
- **Molteplicità predefinita:** La molteplicità predefinita di un attributo è [1] nel meta-modello. In un diagramma UML, un attributo senza molteplicità non indica automaticamente [1], poiché la molteplicità potrebbe essere nascosta (soppressione delle informazioni). Se la molteplicità [1] è importante, indicala esplicitamente per evitare ambiguità.

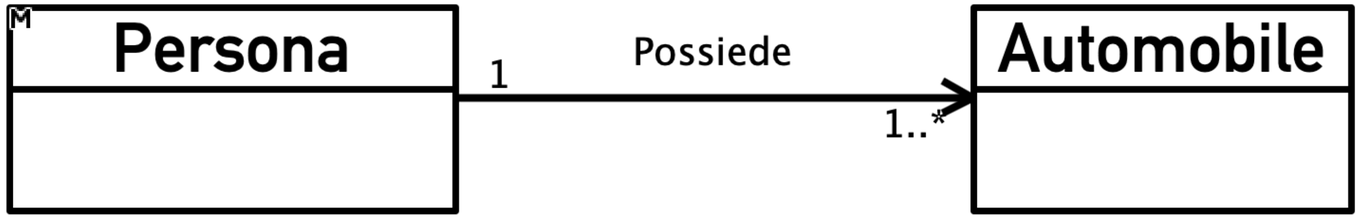
ASSOCIAZIONI: VERSO DI NAVIGAZIONE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Da un punto di vista concettuale, una associazione non ha un verso di percorrenza (una direzionalità). Se A è legato a B, B è legato ad A



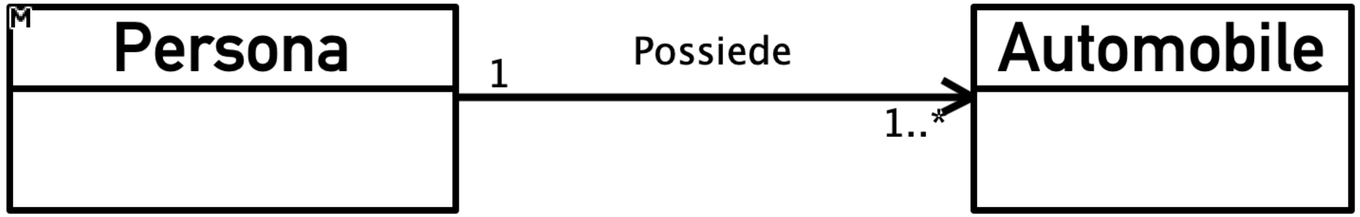
- Se specificata con una **freccia**, la direzionalità attribuisce alla classe origine del verso di percorrenza la responsabilità di tenere traccia dell'associazione (entrambe le classi in caso di freccia bidirezionale)



ASSOCIAZIONI: VERSO DI NAVIGAZIONE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Indica in quale direzione è possibile reperire le informazioni

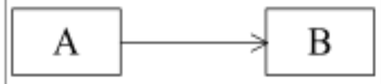


- In questo diagramma, da una persona è possibile sapere quali sono le automobili che possiede
- Considerata un'istanza di automobile non è possibile conoscere il possessore
- **Nota Importante:** Tale informazione è utile soprattutto nel **progetto di dettaglio**, rispecchiando una scelta di progetto. Non è presente (di solito) nei **progetti concettuali**

ASSOCIAZIONI: VERSO DI NAVIGAZIONE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

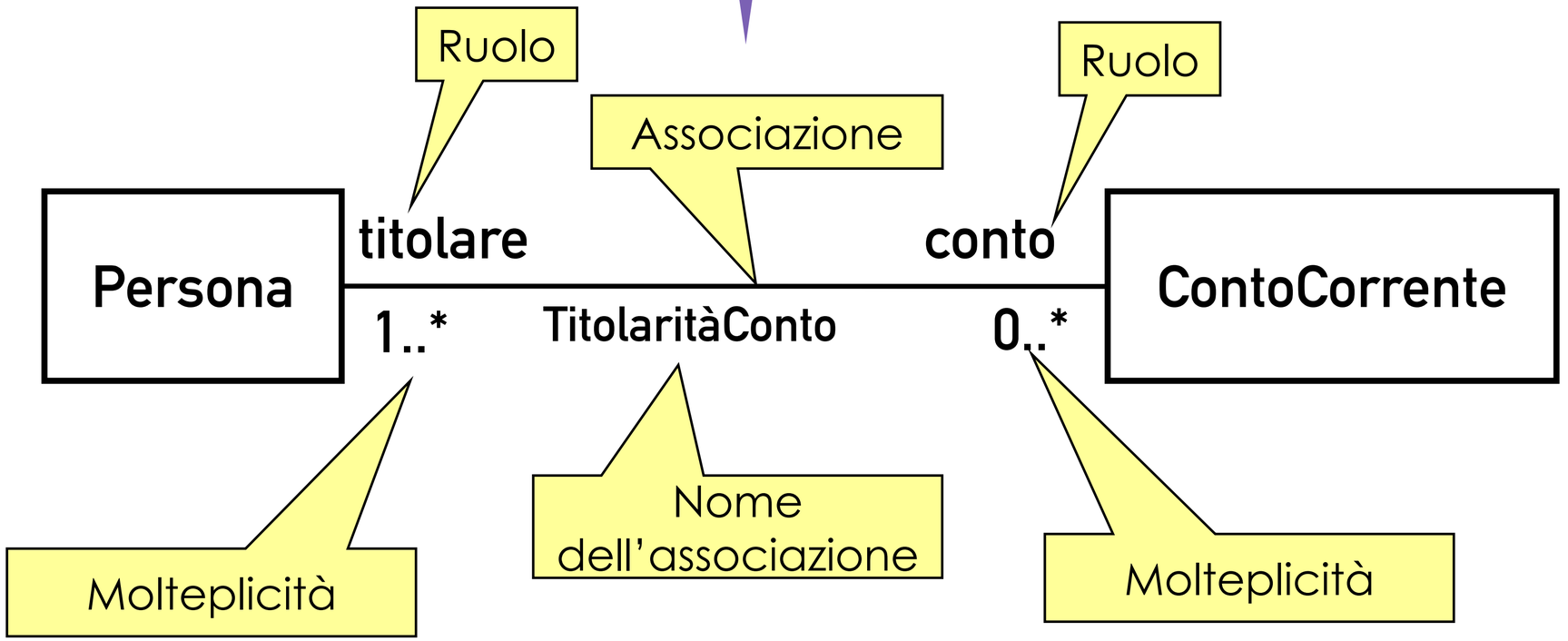
- UML 2 supporta diversi idiomi per esprimere la navigabilità tra associazioni

Sintassi UML 2	Idioma 1: Navigabilità UML 2.0 stretta	Idioma 2: Nessuna navigabilità	Idioma 3: standard nella pratica
	Da A a B è navigabile Da B a A è navigabile		
	Da A a B è navigabile Da B a A non è navigabile		
	Da A a B è navigabile Da B a A è indefinito		Da A a B è navigabile Da B a A non è navigabile
	Da A a B è indefinito Da B a A è indefinito	Da A a B è indefinito Da B a A è indefinito	Da A a B è navigabile Da B a A è navigabile
	Da A a B non è navigabile Da B a A non è navigabile		

ASSOCIAZIONI: RIEPILOGO

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

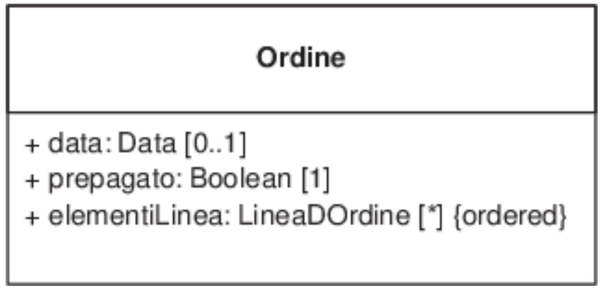
CONSIGLIO DI USARE NOMI DELL'ASSOCIAZIONE OPPURE RUOLI. USARLI INSIEME SAREBBE RIDONDANTE E APPESANTIREBBE IL DIAGRAMMA

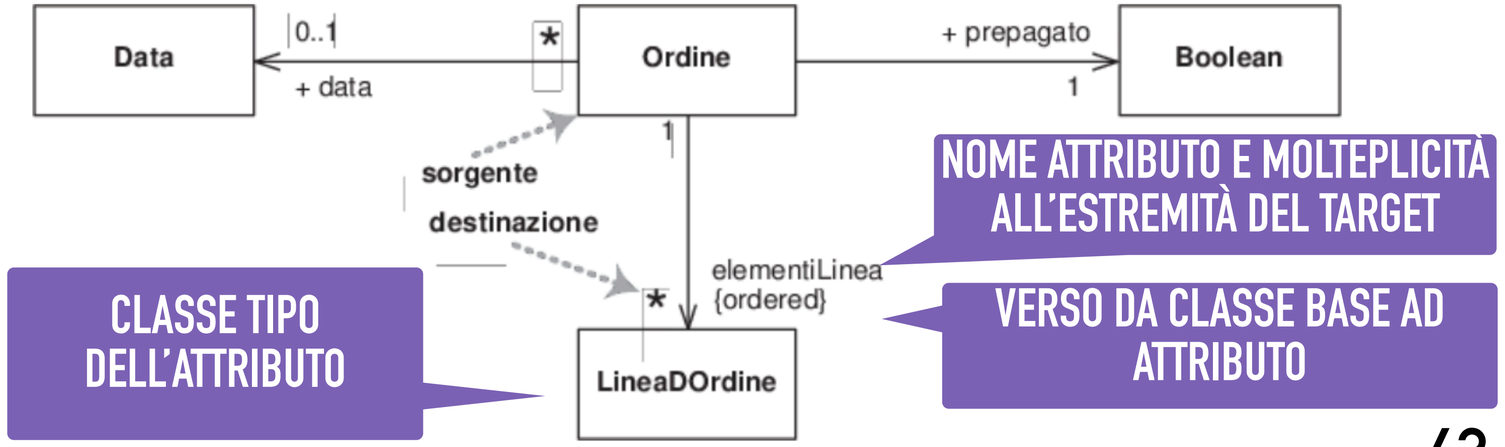


ASSOCIAZIONI VS ATTRIBUTI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

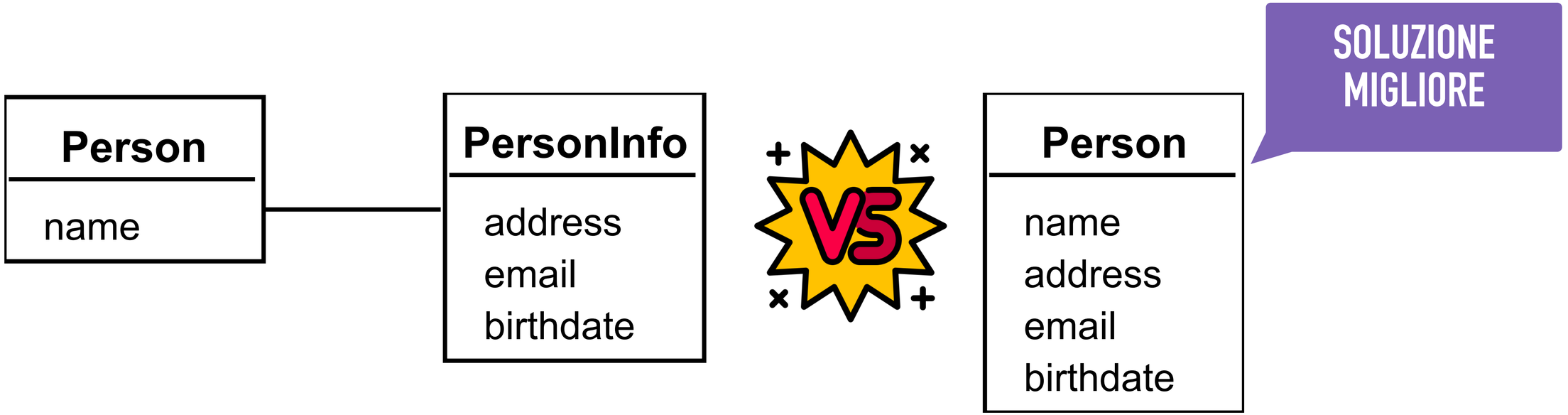
- Le caratteristiche strutturali di una classe possono essere rappresentate sia come associazioni che come attributi di una classe
- A differenza degli attributi, le associazioni possono riportare anche le molteplicità di entrambe le classi ma sono meno compatte
- Il progettista deve scegliere le entità che hanno maggiore importanza e rappresentarle come classi per dar loro più enfasi





ASSOCIAZIONI VS ATTRIBUTI

- Meglio non appesantire il diagramma con troppe associazioni 1 a 1
- Si possono usare **attributi** per concetti secondari (come tipi semplici) e **associazioni** per classi più significative
- La scelta dipende da cosa si vuole enfatizzare nel diagramma

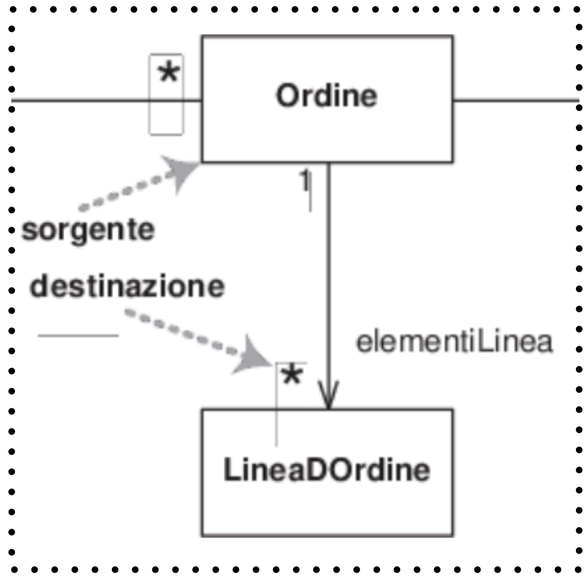


IMPLEMENTAZIONE DELLE ASSOCIAZIONI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- La rappresentazione più comune di un'associazione o un'attributo di un entità in linguaggi di programmazione OO è tramite un attributo di una classe (in alternativa si possono usare metodi getter e setter)
- Se definito un verso di navigazione, la classe origine ha un attributo con:
  - Nome: il ruolo della destinazione
  - Tipo: la classe destinazione
  - Molteplicità: la molteplicità della destinazione

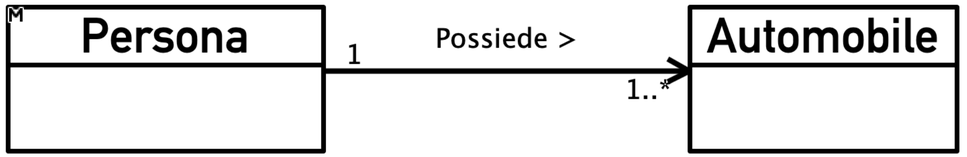
```
public Class Ordine{
  private LineaDOrdine[] elementiLinea;
}
```



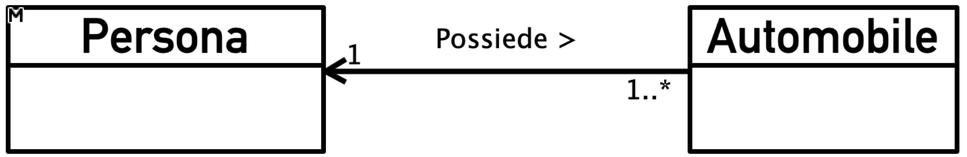
IMPLEMENTAZIONE DELLE ASSOCIAZIONI: UNO A MOLTI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

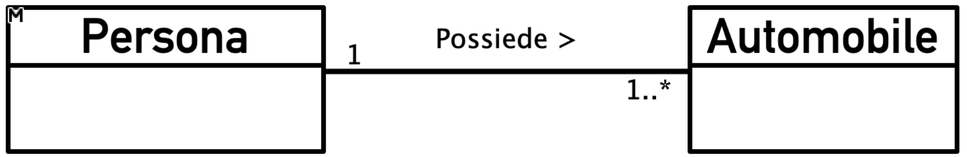
- La classe Persona ha un attributo vettore di Automobili



- La classe Automobile ha un attributo Persona



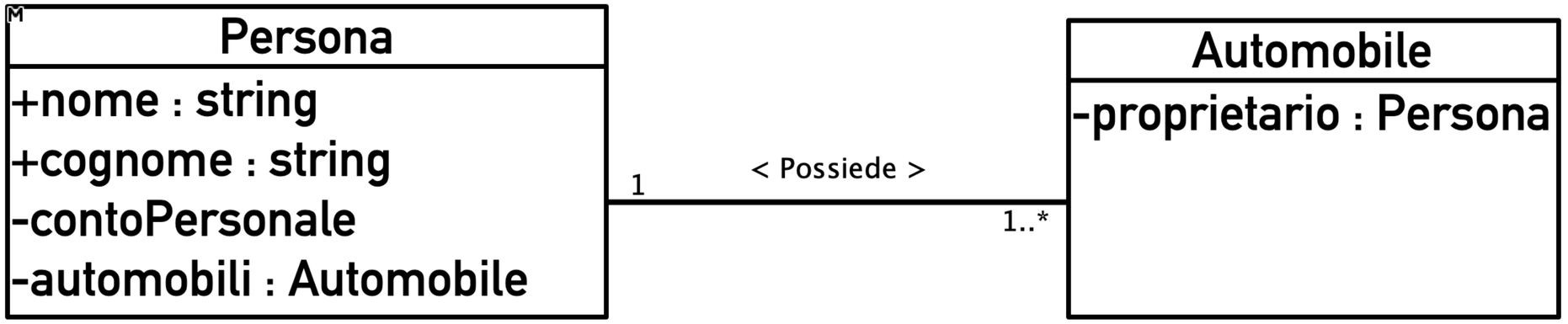
- Entrambe: tutte e due le classi permettono di navigare l'altra classe dell'associazione (nessuna freccia o doppia freccia)



IMPLEMENTAZIONE DELLE ASSOCIAZIONI: UNO A MOLTI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- Entrambe le classi permettono di navigare l'altra classe dell'associazione
- Persona ha un attributo automobili di tipo Automobile con molteplicità [1..*]
- Automobile ha un attributo proprietario di tipo Persona con molteplicità [1]

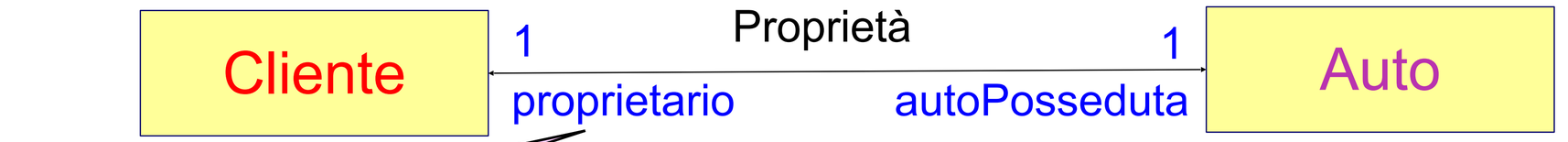


IMPLEMENTAZIONE DELLE ASSOCIAZIONI

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

- **Uno a Uno (es: uno Studente, un Badge)**
  - La classe Badge è implementata avendo un attributo Studente
  - Oppure la classe Studente è implementata avendo un attributo Badge
  - O entrambe se bidirezionale
- **Molti a molti (es: molti Studente, molti Esame)**
  - La classe Studente è implementata avendo anche un attributo array di Esame,
  - Oppure la classe Esame è implementata avendo anche un attributo array di Studente
  - O entrambe se bidirezionale

ESEMPIO CODIFICA IN JAVA: ASSOCIAZIONE BIDIREZ. UNO-A-UNO



ruolo svolto nell'associazione

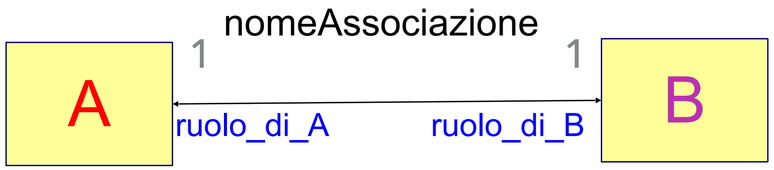
metodo per associare un'auto a un cliente

```
class Cliente {
  public associaAuto(Auto x) {this.autoPosseduta = x;}

  private Auto autoPosseduta;
};

class Auto {
  public associaProprietario(Cliente p)
     {this.proprietario=p;}

  private Cliente proprietario;
};
```

```
public static void main() {
  Cliente c;
  Auto a;

  c.associaAuto(a);
  a.associaProprietario(c);
};
```

il ruolo diventa il nome della variabile membro

CODIFICA JAVA ASSOCIAZIONE BIDIREZ. UNO-A-UNO (CASO GENERICO)

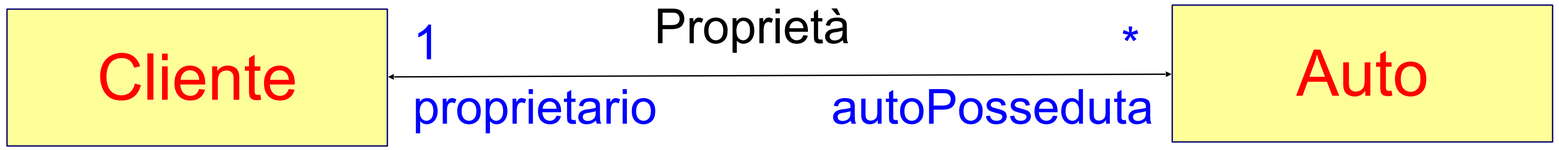



```
public static void main() {
 A a;
 B b;
 a.associa(b); //collega b ad a
 b.associa(a); //collega a a b

 //scollega b da a:
 a.rimuoviRuolo_di_B();
 //senza dimenticare di
 //scollegare a da b!
  b.rimuoviRuolo_di_A();
};
```

```
class A {
   public associa(B b) {this.ruolo_di_B = b;}
   public rimuoviRuolo_di_B()
                        {this.ruolo_di_B=NULL;}
   private B ruolo_di_B;
};

class B {
  public associa(A a) {this.ruolo_di_A = a;}
  public rimuoviRuolo_di_A
                        {this.ruolo_di_A = NULL;}
  private A ruolo_di_A;
};
```

IN ASSOCIAZIONI BIDIREZIONALI BISOGNA MANTENERE ENTRAMBE LE PROPRIETÀ SINCRONIZZATE

ESEMPIO CODIFICA IN JAVA ASSOCIAZIONE BIDIREZ. UNO-A-MOLTI

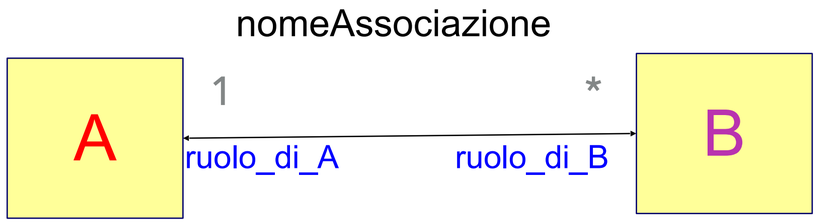


La classe Cliente ha come variabile membro un ArrayList di Auto.

```
class Cliente {
  public associaAuto(Auto x) {inserisci x in ArrayList}
  public rimuoviAuto(Auto x) {rimuovi x da ArrayList}
  private ArrayList<Auto> autoPosseduta;
};

class Auto {
  public associaProprietario(Cliente p)
     {this.proprietario=p;}
  public rimuoviProprietario() {this.proprietario=NULL;}

  private Cliente proprietario;
};
```

```
public static void main() {
  Cliente c;
  Auto a1, a2;

  c.associaAuto(a1);
  a1.associaProprietario(c);
  c.associaAuto(a2);
  a2.associaProprietario(c);
  …
  c.rimuoviAuto(a1);
  a1.rimuoviProprietario();
};
```

CODIFICA JAVA ASSOCIAZIONE BIDIREZ. UNO-A-MOLTI (CASO GENERICO)




```
public static void main() {
  A a;
  B b1, b2;
  a.aggiungi(b1); //collega b1 ad a
  b1.associa(a);  //ed a a b1
  a.aggiungi(b2); //collega b2 ad a
  b2.associa(a);  //ed a a b2
  …
  a.rimuovi(b1); //scollega b1 da a
  //non dimenticare di
  //scollegare a da b1:
  b1.rimuoviRuolo_di_A();
};
```

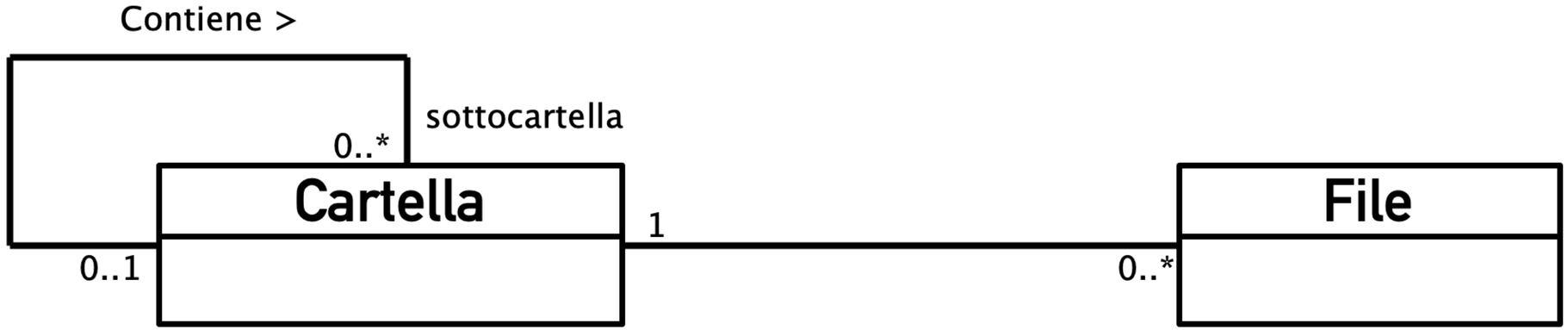
```
class A {
   public aggiungi(B b) {inserisci b in ArrayList}
   public rimuovi(B b) {rimuovi b da ArrayList}
   private ArrayList<B> ruolo_di_B;
};
class B {
  public associa(A a) {this.ruolo_di_A = a;}
  public rimuoviRuolo_di_A()
          {this.ruolo_di_A = NULL;}
  private A ruolo_di_A;
};
```

ASSOCIAZIONI RIFLESSIVE E CLASSI ASSOCIATIVE

ASSOCIAZIONI RIFLESSIVE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

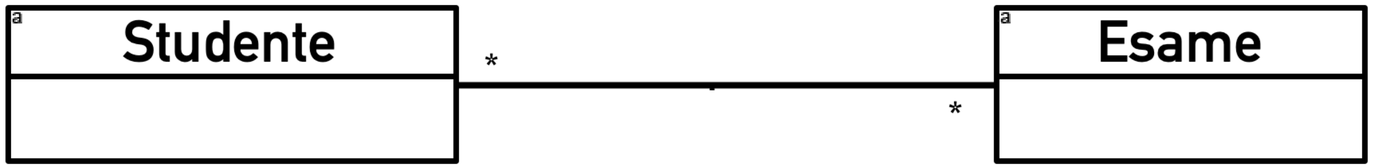
- Associazione che collega una classe con sé stessa
- Quindi oggetti di una classe possono essere associati ad oggetti appartenenti alla classe stessa



- Una cartella può contenere 0 o più sottocartelle (nessun valore massimo, ipoteticamente infinite)
- Una cartella può essere contenuta al massimo in una sola altra cartella

CLASSI ASSOCIATIVE

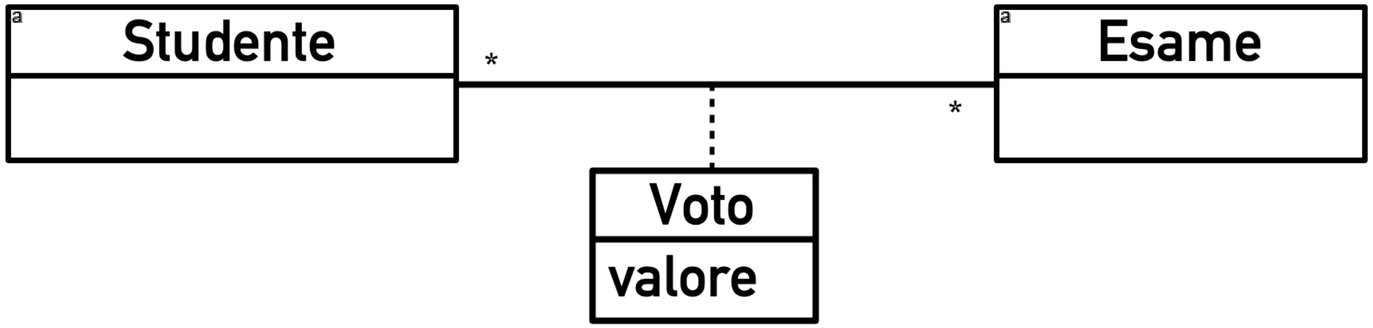
Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine



- Si vuole rappresentare che ad ogni studente viene assegnato un voto per ciascun esame
- A quale classe assegniamo il valore del voto come attributo?
- Il voto è una caratteristica dell'associazione stessa perché caratterizza l'associazione tra un'istanza di studente ed un'istanza di esame: uno studente può avere un solo voto finale per un dato esame

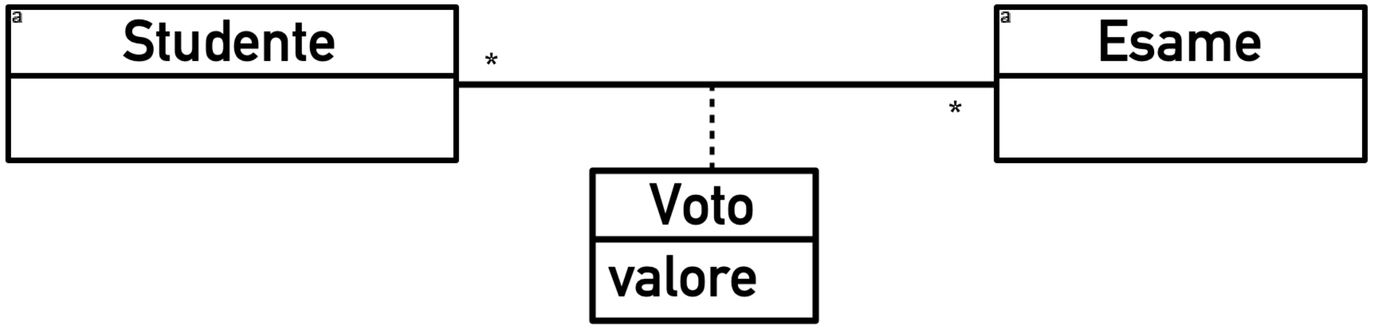
CLASSI ASSOCIATIVE

- In alcuni casi, un attributo caratterizza un'associazione ed è difficile stabilire a quale delle classi coinvolte appartiene
- In questi casi, può essere opportuno definire una classe associativa
- Una classe associativa può avere attributi, metodi, altre associazioni
- Sono utilizzate in genere per associazioni molti a molti



CLASSI ASSOCIATIVE

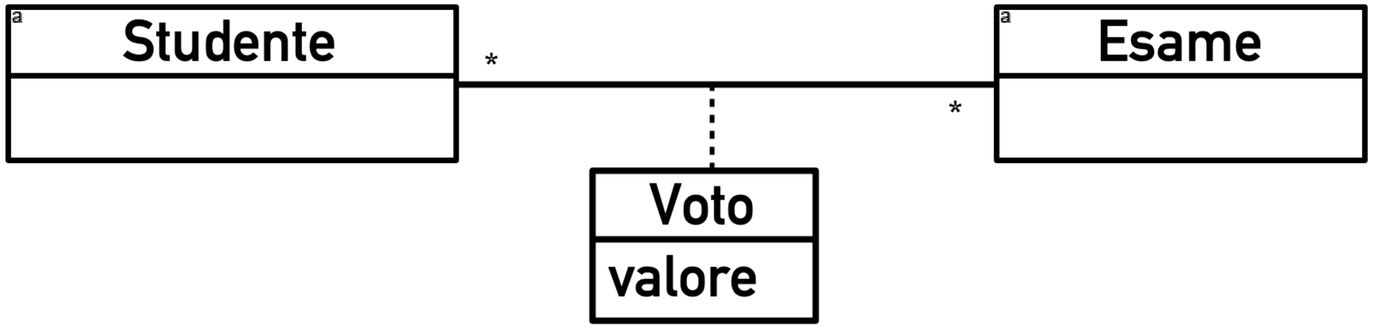
- La classe associativa connette due classi e definisce un insieme di caratteristiche proprie della associazione stessa
- Ciascuna istanza di classe associativa, rappresenta un'associazione tra due istanze delle classi e contiene specifici valori degli attributi
- Poiché caratterizza un'associazione, prima di creare un'istanza della classe associativa, devono esistere le istanze delle classi collegate



CLASSI ASSOCIATIVE

Vincenzo Riccio
Ingegneria del Software 2025/2026
Università degli Studi di Udine

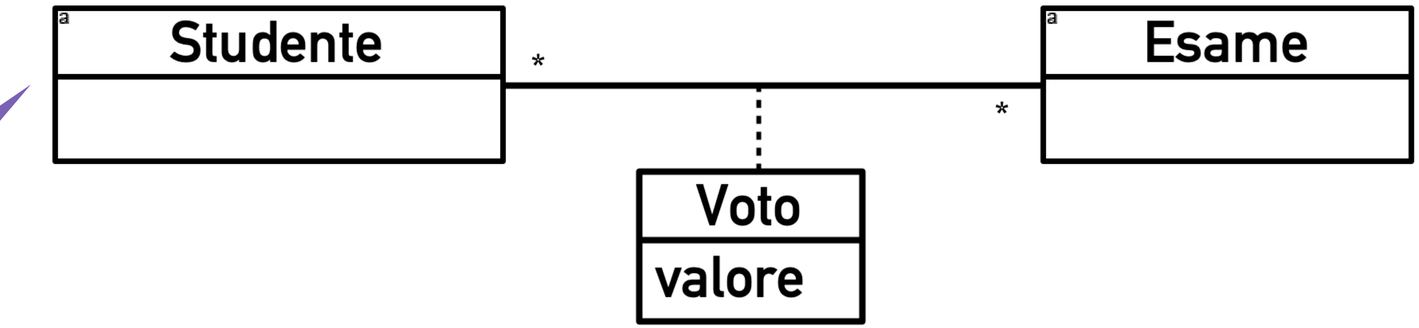
- Ci può essere solo un'istanza di una specifica classe associativa tra ogni coppia di oggetti associati (istanze delle classi)
- Uno studente avrà un voto specifico per ogni esame
- Uno studente può avere un solo voto in un esame



CLASSI ASSOCIATIVE

- Rappresentano collegamenti concettuali
- Per *reificare* (rendere reale) una classe associativa sarà opportuno (tipicamente in fase di progettazione) trasformarla in una classe

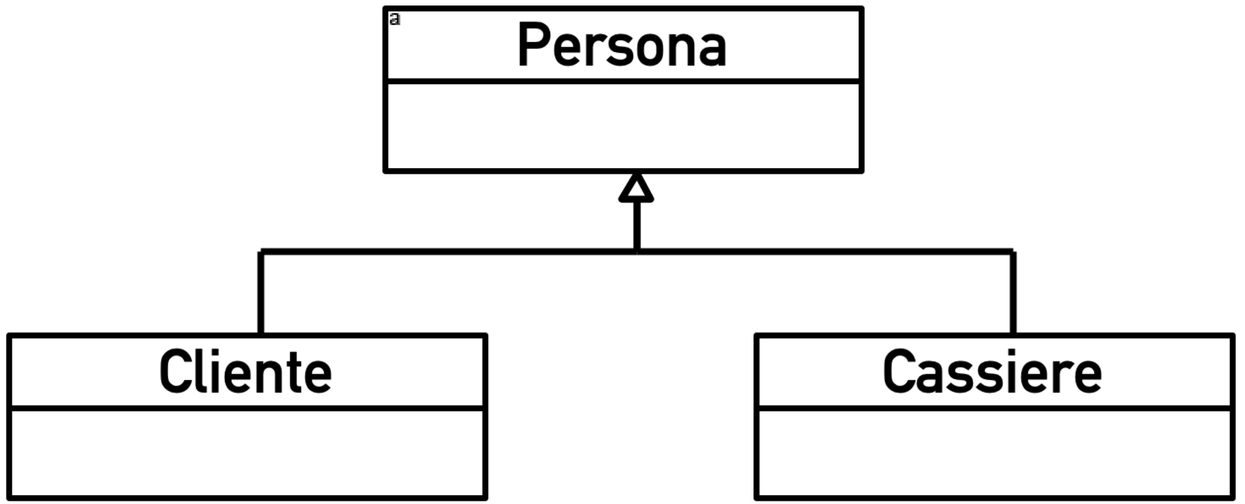
SOLUZIONI EQUIVALENTI MA LA CLASSE ASSOCIATIVA È PIÙ INTUITIVA NEL DIAGRAMMA CONCETTUALE



GENERALIZZAZIONE

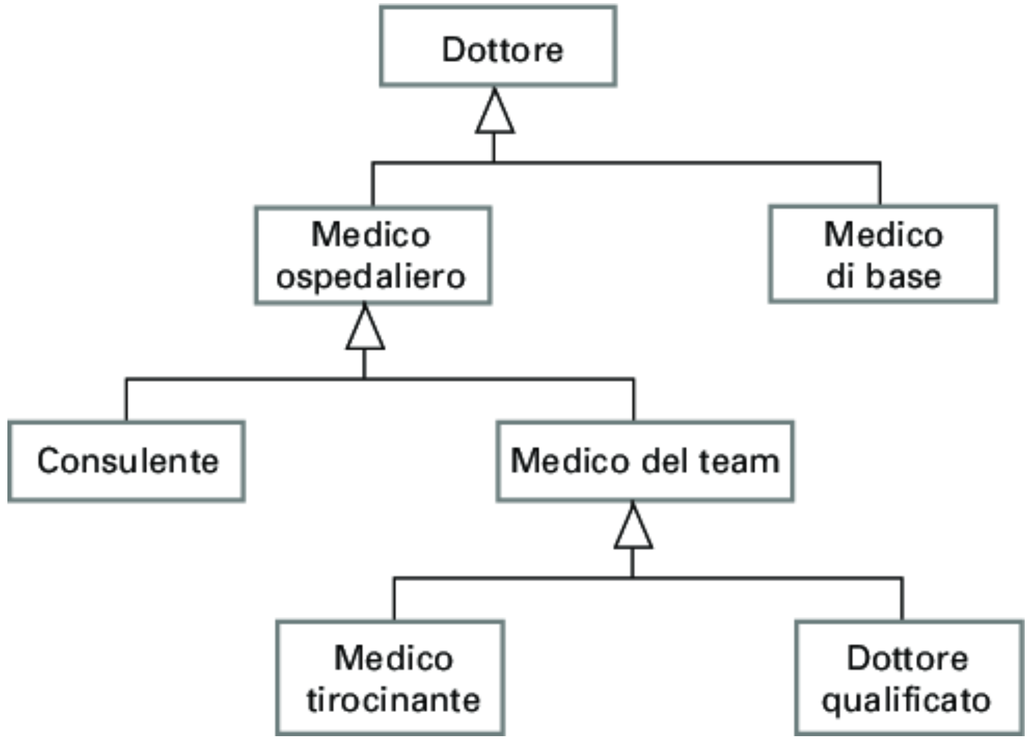
GENERALIZZAZIONE-SPECIALIZZAZIONE

- I concetti di generalizzazione e specializzazione in UML sono analoghi a quelli object oriented
- Indica legami del tipo **is-a** (“è un”) tra le sottoclassi e la superclasse
  - Es.: Il cliente è una persona
- La superclasse è un concetto generale, le sottoclassi sono sue specializzazioni



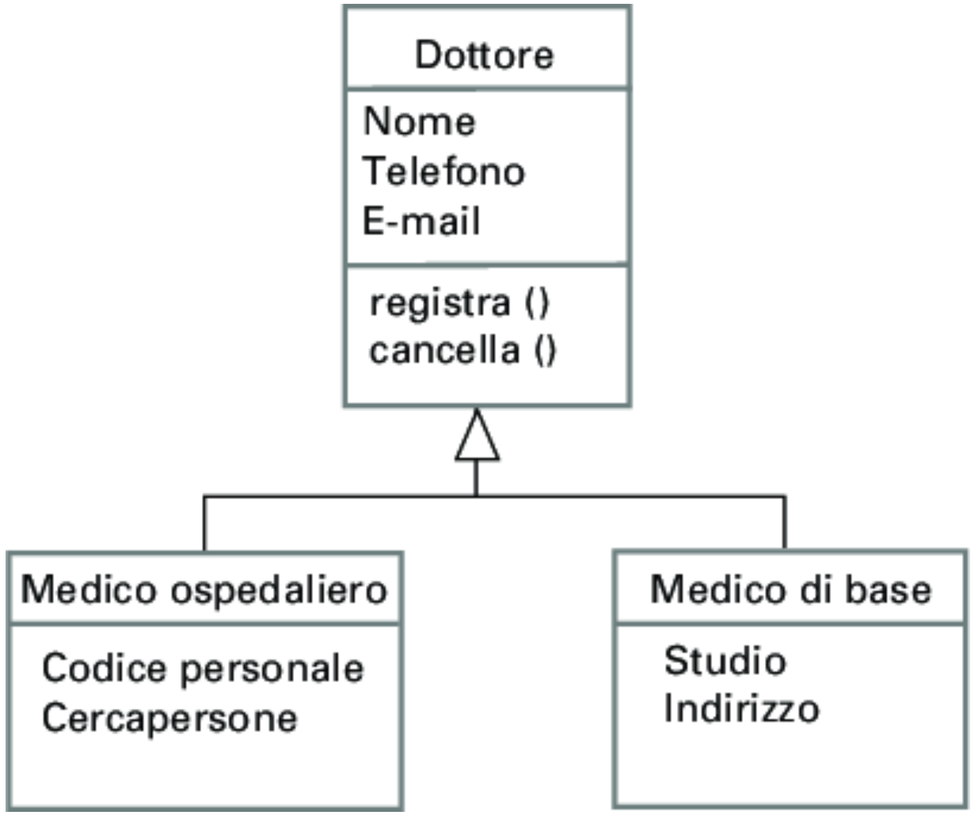
GENERALIZZAZIONE-SPECIALIZZAZIONE

- Tipo di associazione rappresentato da una freccia che va dalla classe specializzata alla classe generale
- Buona pratica di progettazione: informazioni comuni sono mantenute in un solo posto (la classe più generale), facilitando le modifiche del progetto



GENERALIZZAZIONE-SPECIALIZZAZIONE

- Le classi di livello più basso (sottoclassi) hanno tutti gli attributi e operazioni delle classi generali ed aggiungono operazioni e attributi più specifici
- Tutti i dottori hanno E-mail ma solo i medici ospedalieri hanno un cercapersone



GENERALIZZAZIONE-SPECIALIZZAZIONE

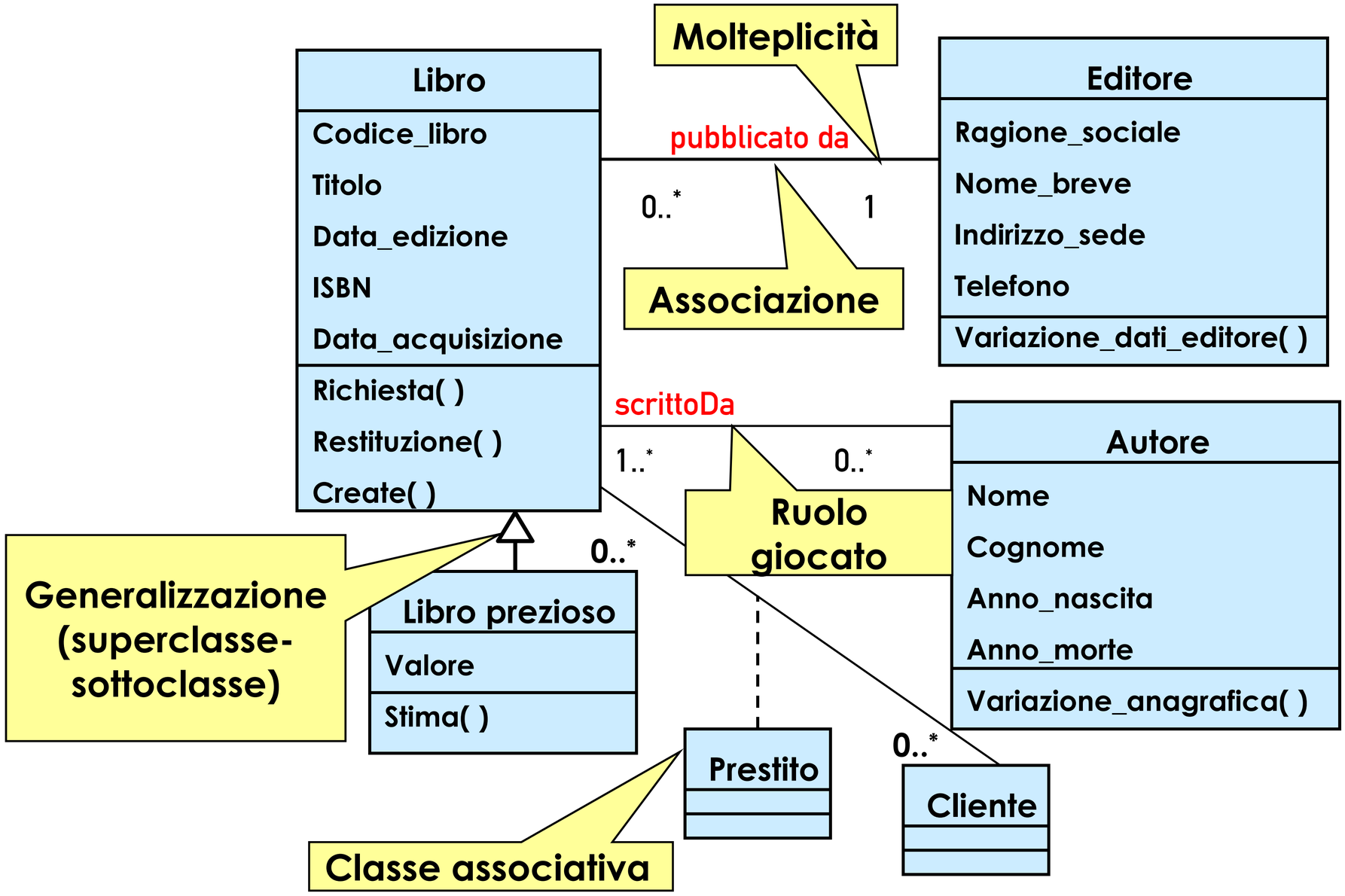
- A livello **concettuale** (*analisi dei requisiti*), gen-spec esprime una relazione is-a tra un concetto generale e le sue specializzazioni
- A livello di **dettaglio** (*progetto* e *implementazione*), gen-spec può essere interpretato:
  - Come una relazione di **ereditarietà** tra due classi concrete. Le classi derivate ereditano attribuiti e metodi public e protected della classe base
  - Come una relazione di **realizzazione** tra una classe astratta e una classe concreta. La classe base ha soltanto prototipi di metodi, la classe specializzata implementa i metodi sfruttando l'overriding

GENERALIZZAZIONE-SPECIALIZZAZIONE

Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

- Quando Gen-Spec descrive una gerarchia di **ereditarietà:**
  - La superclasse raccoglie caratteristiche e comportamenti comuni agli elementi delle sottoclassi
  - La classe derivata (sottoclasse) eredita metodi e attributi della classe base (superclasse) e può:
    1. Aggiungere altri attributi e operazioni
    2. Ridefinire i metodi della classe base (*overriding*)
- Vedremo invece la realizzazione delle classi astratte nel prossimo blocco di slides

ESEMPIO

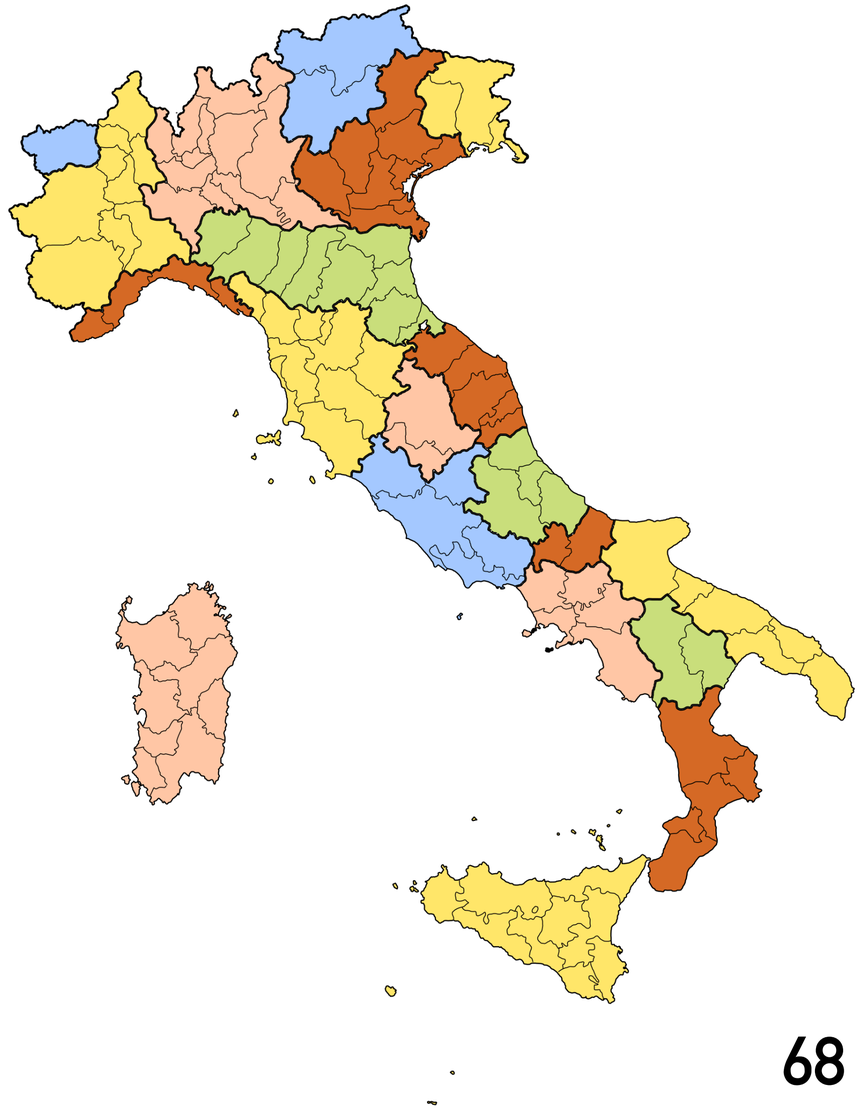

Molteplicità
Libro
Codice_libro
Titolo
Data_edizione
ISBN
Data_acquisizione
Richiesta( )
Restituzione( )
Create( )
pubblicato da
0..*
1
Associazione
Editore
Ragione_sociale
Nome_breve
Indirizzo_sede
Telefono
Variazione_dati_editore( )
scrittoDa
1..*
0..*
Ruolo giocato
Autore
Nome
Cognome
Anno_nascita
Anno_morte
Variazione_anagrafica( )
Generalizzazione (superclasse-sottoclasse)
0..*
Libro prezioso
Valore
Stima( )
Prestito
0..*
Cliente
Classe associativa

CONTENIMENTO

RELAZIONE DI CONTENIMENTO

Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

- Un tipo speciale di associazione tra classi è la relazione di contenimento, che rappresenta il legame tra un insieme (il “tutto”) e le sue parti (legame tutto-parti), ovvero tra un contenitore e il contenuto (legame contenitore-contenuto)
- Ad esempio: relazione tra stato, regioni e province

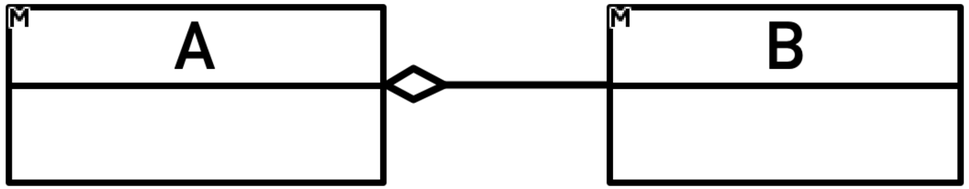


CONTENIMENTO: AGGREGAZIONE E COMPOSIZIONE

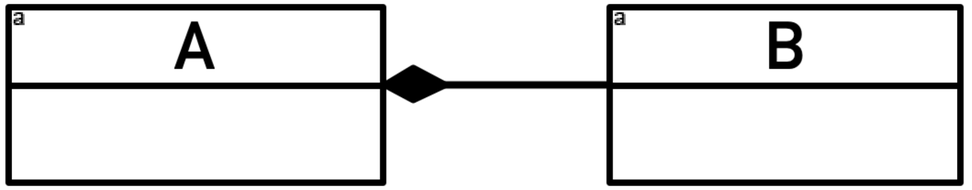
Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

- Il contenimento può essere:
  - debole (o lasco), detto anche **aggregazione**: la parte mantiene la sua identità quando entra a far parte del tutto (Es.: auto – motore)
  - stretto, detto anche **composizione**: la parte perde la sua identità quando entra a far parte del tutto (Es.: pane – farina)

- L'aggregazione è modellata da un rombo vuoto



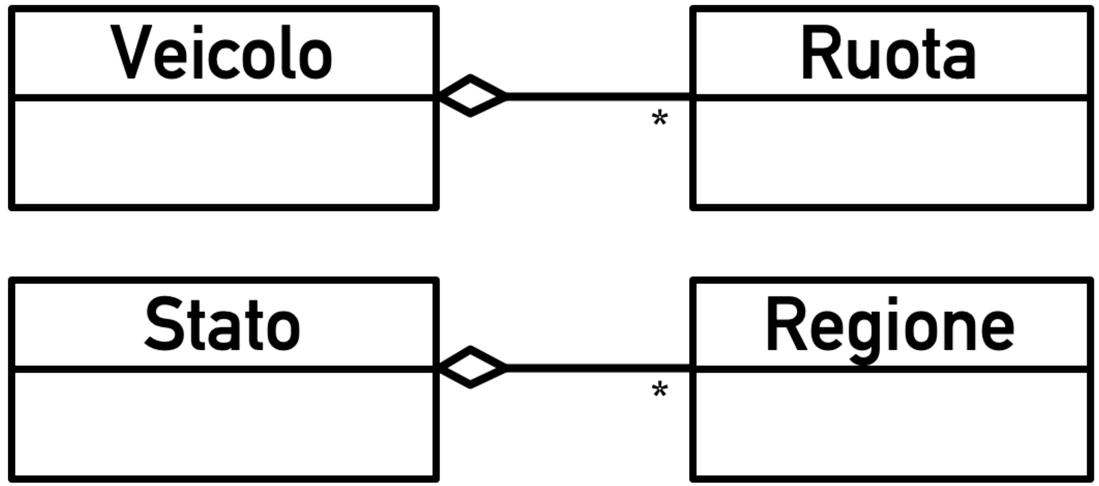
- La composizione è modellata da un rombo pieno



AGGREGAZIONE

Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

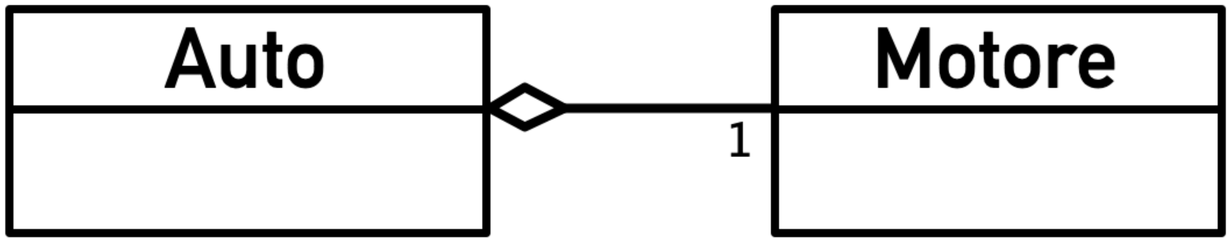
- Il lato del “tutto” (quella con il rombo) è spesso chiamato **aggregato**
- Le parti possono esistere indipendentemente dall'aggregato
- È possibile che più aggregati condividano una stessa parte
- La molteplicità dal lato dell’aggregato, quando è sottintesa, vale 0..1



AGGREGAZIONE: CICLO DI VITA

Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

- Il ciclo di vita dell'oggetto contenuto è indipendente da quello dell'oggetto contenitore. L'oggetto contenuto può esistere anche indipendentemente dal contenitore
- L'oggetto contenitore non è necessariamente responsabile della costruzione e distruzione dell'oggetto contenuto
- Es.: Un’auto ha un motore. Il motore di un’auto può vivere anche indipendentemente dall’auto (per es., si può rottamare l’auto ma prima smontarne il motore e montarlo su un’altra auto)



TRADUZIONE DELL'AGGREGAZIONE IN UN LINGUAGGIO AD OGGETTI

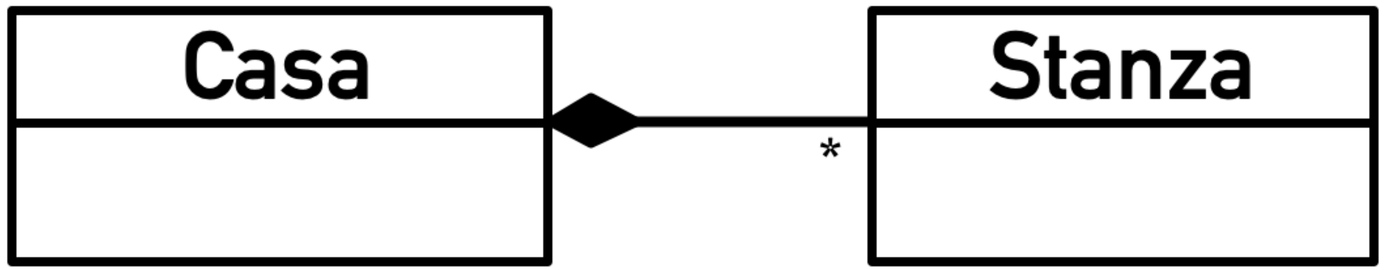
- L'aggregazione si traduce in un linguaggio ad oggetti:
  - Inserendo nella classe contenitore una variabile membro di tipo puntatore o un riferimento ad un oggetto della classe contenuto
  - Implementando un costruttore della classe contenitore che riceva in ingresso un puntatore o un riferimento ad un oggetto della classe contenuto

TRADUZIONE DELL'AGGREGAZIONE IN JAVA

- In particolare, l'aggregazione si può realizzare in Java attraverso un contenitore che:
  - Ha una variabile membro privata di tipo riferimento ad un oggetto del tipo contenuto X
  - Un costruttore che riceve in input un riferimento ad un oggetto di tipo X

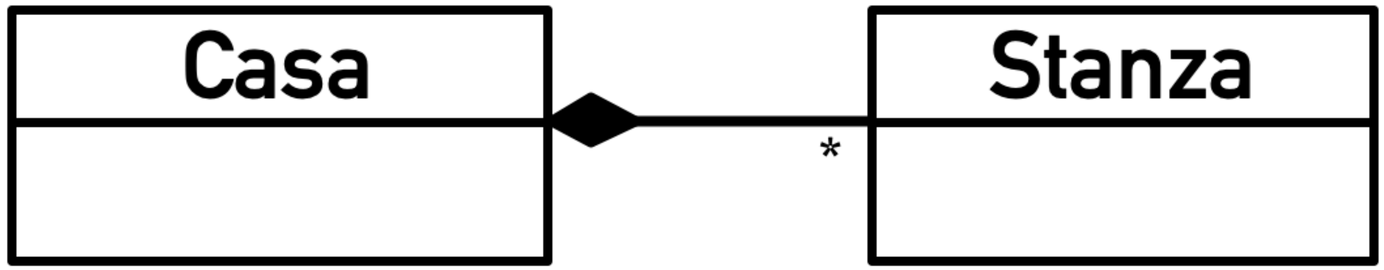
COMPOSIZIONE

- Forma forte di contenimento
- In un problema, potrebbe non aver senso parlare di stanze se non sono legate alla casa in cui si trovano



COMPOSIZIONE: CICLO DI VITA

- Le parti non esistono senza il tutto. Se il composito viene distrutto anche le sue parti saranno distrutte (o la responsabilità è ceduta a un altro oggetto)
- Il composito è responsabile della costruzione e distruzione degli oggetti contenuti
- Ogni parte può appartenere ad un solo composito per volta



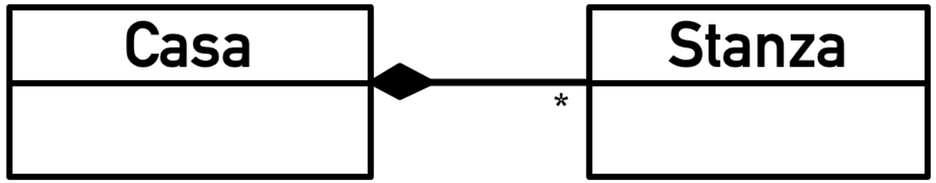
COMPOSIZIONE: PRINCIPIO DI NON CONDIVISIONE

Vincenzo Riccio
Ingegneria del Software 2023/2024
Università degli Studi di Udine

- Un'istanza di punto può essere parte di un poligono oppure il centro del cerchio ma non entrambi
- Una classe può essere componente di più classi
- Un'istanza di classe componente può essere componente di un solo oggetto composito, anche se la classe componente può essere componente di molteplici classi composito diverse

COMPOSIZIONE

- La molteplicità dal lato del “tutto” (il composito), quando è sottintesa, vale 1 nel caso in cui la classe componente ha solo un’altra classe composito



- L’altro caso è quando la classe componente può essere contenuta in due o più classi composito diverse. In questo caso la molteplicità é 0..1

COMPOSIZIONE IN UN LINGUAGGIO AD OGGETTI

- La composizione si traduce in un linguaggio ad oggetti:
  - Aggiungendo alla classe contenitore una variabile membro del tipo della classe contenuto
  - Implementando un costruttore del contenitore che richiami il costruttore del contenuto
  - Il metodo distruttore della classe contenitore deve gestire la distruzione dell'oggetto contenuto o il passaggio della sua responsabilità a un'altra classe
- L'utente della classe contenitore ha la responsabilità di instanziarne un oggetto, richiamando il suo costruttore con tutti i valori necessari all'inizializzazione sia del contenitore sia del suo contenuto

CODIFICA DELLA COMPOSIZIONE IN JAVA

- In particolare, la composizione si può realizzare in Java attraverso un contenitore C avente:
  - una variabile membro privata di tipo riferimento ad un oggetto del tipo contenuto X
  - un costruttore che costruisce un oggetto c di tipo C con dentro anche il suo contenuto x di tipo X

ESEMPIO DI COMPOSIZIONE (UNO-A-UNO) IN JAVA

```
class Contenitore {
  public Contenitore(String s1, String s2) {
     nomeContenitore = s1;
     c = new Contenuto(s2);
  }

  private String nomeContenitore;
  private Contenuto c;        // composizione: variabile membro di tipo Contenuto
};
```

```
class Contenuto {
   public Contenuto(String s) {nomeContenuto=s};

   private String nomeContenuto;
};
```