

# **Unified Modeling Language Class Diagram**

Sandro Morasca

Università degli Studi dell'Insubria

Dipartimento di Scienze Teoriche e Applicate

Via Ottorino Rossi 9 – Padiglione Rossi

21100 Varese, Italy

[sandro.morasca@uninsubria.it](mailto:sandro.morasca@uninsubria.it)



- Introduzione
- Classi
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

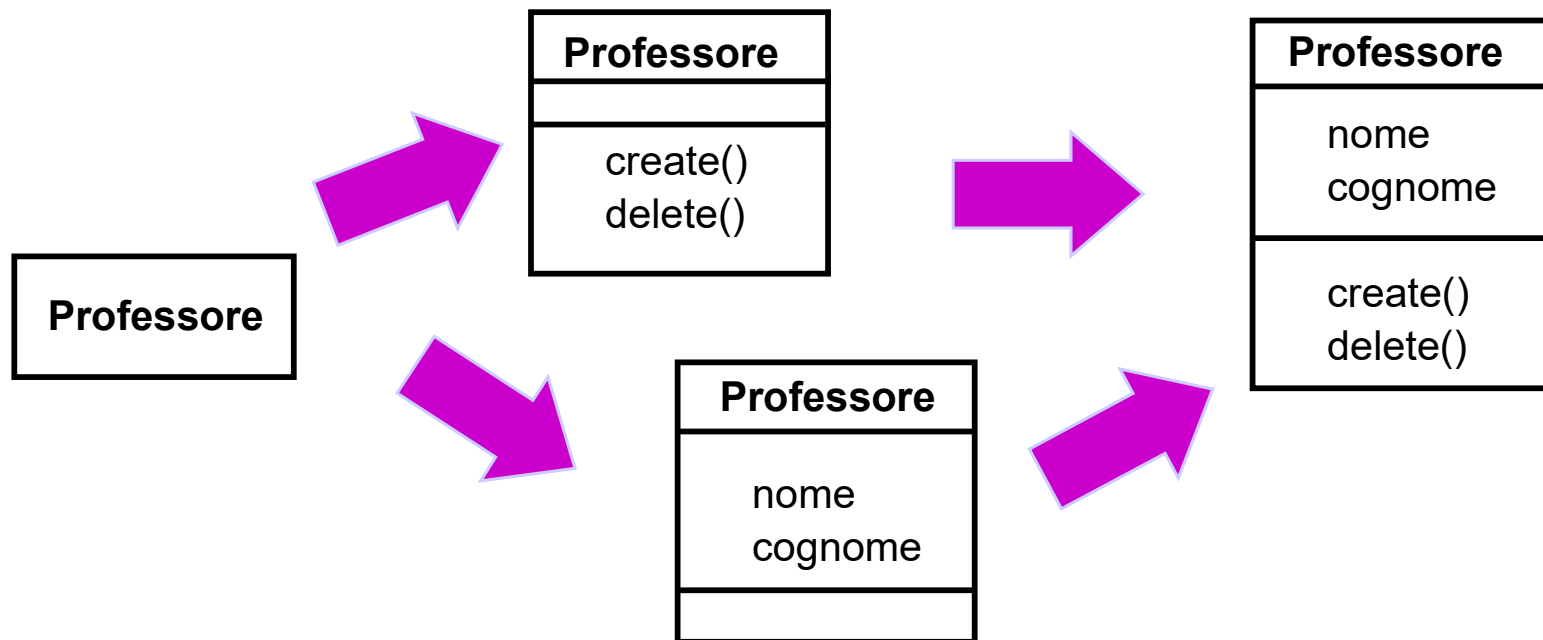
- Definiscono la visione statica del sistema
  - classi
  - relazioni tra classi
    - associazione (uso)
    - aggregazione (contenimento)
    - generalizzazione (ereditarietà)
- È forse il modello più importante perché definisce gli elementi base del sistema



- Introduzione
- **Classi**
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

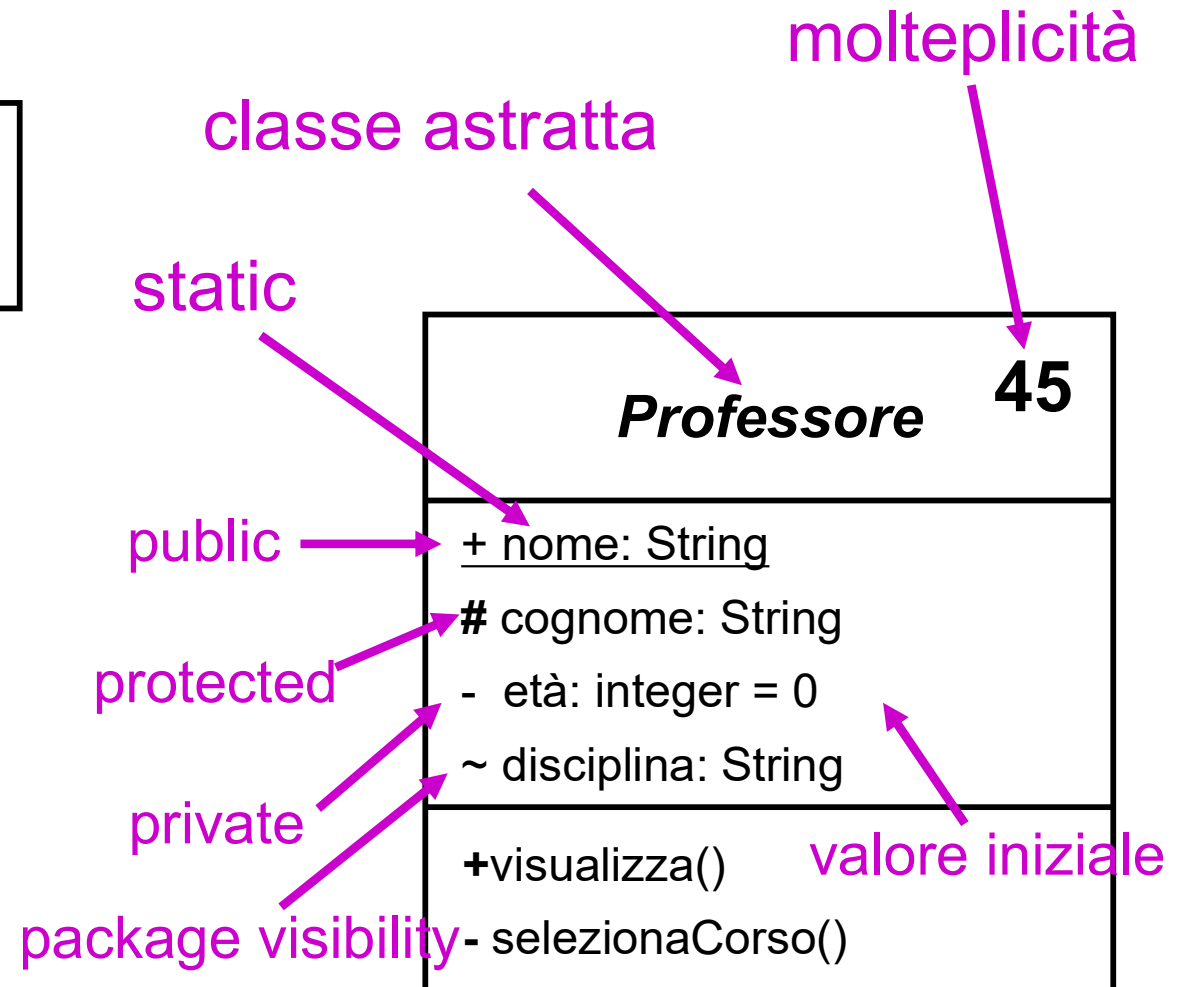
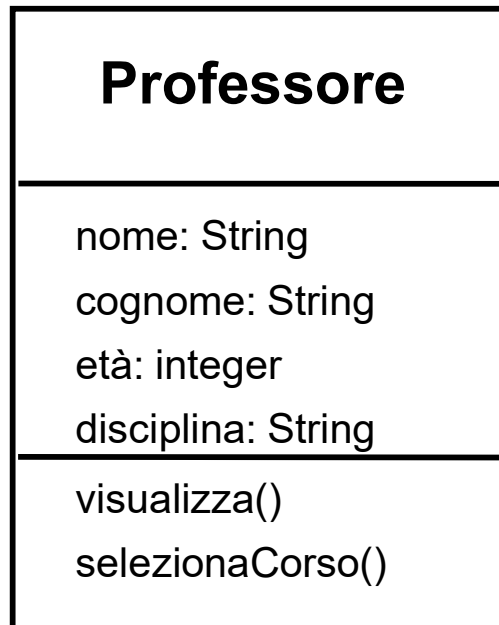
● In UML una classe è composta di tre parti

- nome
- attributi (lo stato)
- metodi (il comportamento)





- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram



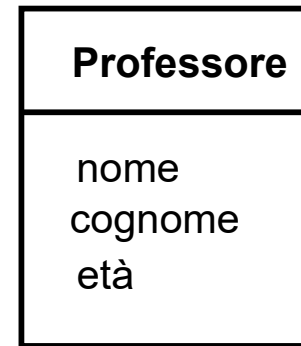
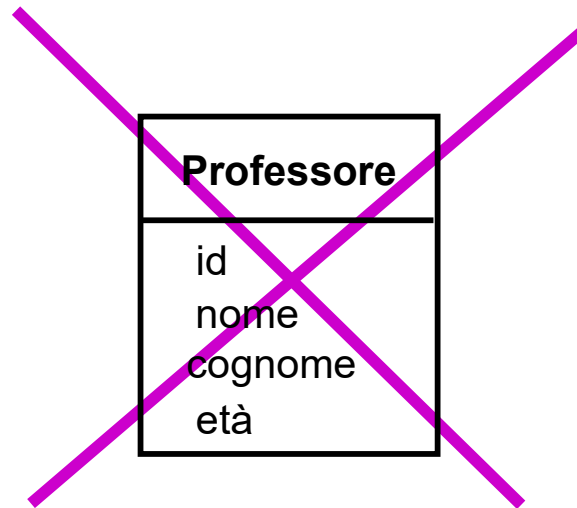


- Introduzione
- **Classi**
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- Un attributo è una caratteristica della classe
- Gli attributi non hanno identità
- Ogni attributo deve essere definito in modo preciso
- Attributi buoni per Studente
  - nome, cognome, ...
- Attributi cattivi
  - corsiScelti



- Gli oggetti avranno una loro identità: non bisogna aggiungerla, come fanno ad esempio i progettisti di database relazionali



Codice fiscale?

- Attenzione: Il codice fiscale per una persona, la targa di un'automobile, il numero di matricola di uno studente, ecc... esistono nel mondo reale e quindi devono esistere anche come attributi (se hanno senso per l'applicazione)



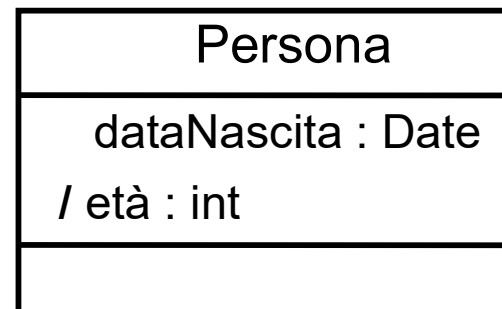
- Introduzione
- **Classi**
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- Nomi che non sono diventati classi
- Durante la definizione delle classi stesse
- Conoscenza del dominio applicativo
  - Persona (ambito bancario)
    - nome, cognome, codiceFiscale, numeroConto
  - Persona (ambito medico)
    - nome, cognome, allergie, peso, altezza



- Introduzione
- Classi
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- Calcolati, non memorizzati
- Si usano quando i loro valori variano frequentemente e la correttezza (precisione) del valore è importante
- Il valore viene calcolato in base ai valori di altri attributi
  - età =  $f(\text{dataDiNascita}, \text{oggi})$
  - area, perimetro =  $f(\text{vertici})$



{età = oggi - dataNascita}





- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- Un'operazione è una funzione o una trasformazione applicabile ad un'istanza di una classe (ogni operazione ha come argomento implicito un oggetto obiettivo)
  - la stessa operazione può essere definita in classi diverse
  - il comportamento dipende dalla classe a cui appartiene l'oggetto obiettivo

Persona
nome: string età: int
cambiaLavoro cambiaIndirizzo

File
nome: string dimensione: int creazione: data
stampa

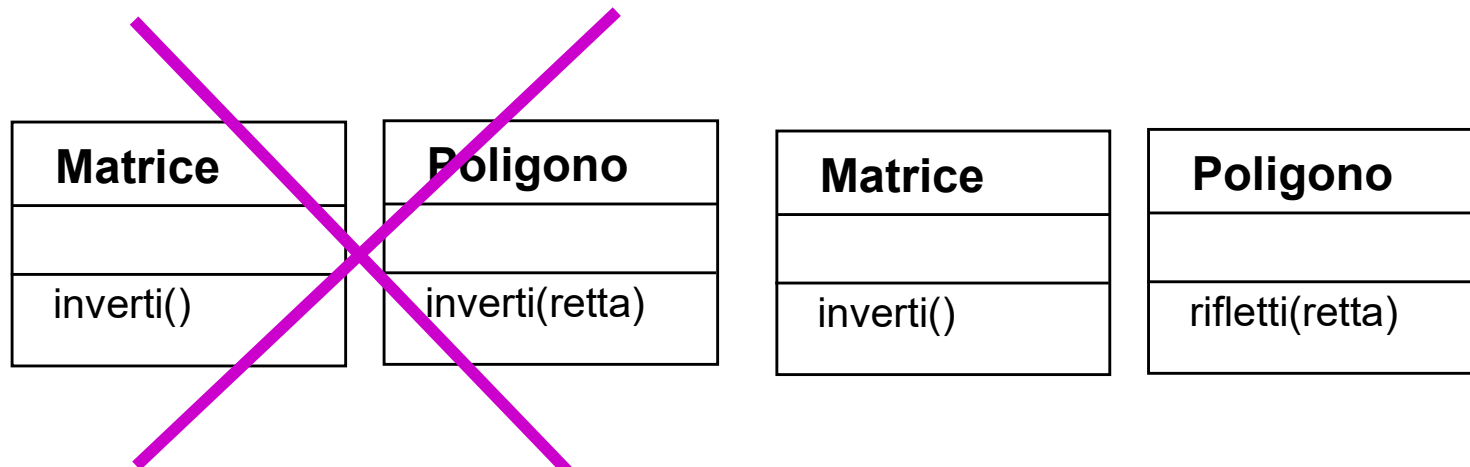
Disegno
titolo: string altezza: int larghezza: int
stampa ruota(gradi: int)

*Classi con  
attributi e  
operazioni*



- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

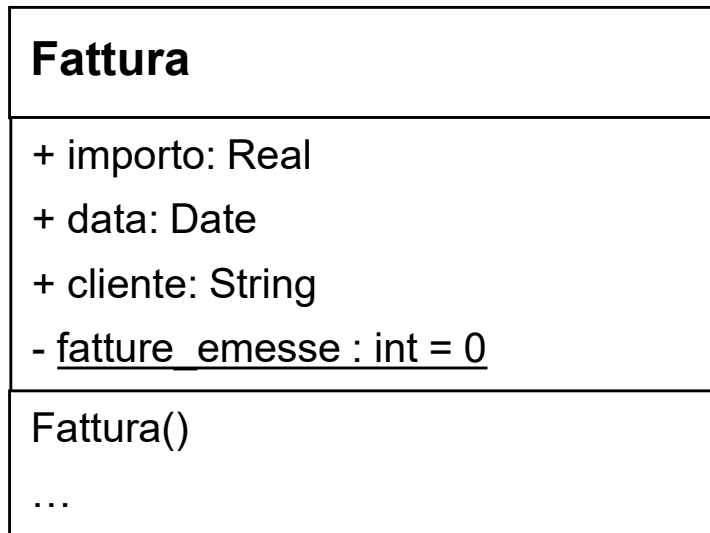
- In classi diverse è possibile riutilizzare più volte lo stesso nome di operazione, ma...
  - non riutilizzare lo stesso nome per operazioni di natura concettualmente diversa
  - stessa operazione stessi parametri
- Se l'operazione è (logicamente) la stessa, allora i parametri (tranne quello implicito, ossia l'oggetto stesso dell'operazione) devono essere identici



Corretto



- Esiste una corrispondenza tra la rappresentazione UML di una classe e l'implementazione con un linguaggio object-oriented (es. Java)



```
public class Fattura {  
    public double importo;  
    public Date data = new Date();  
    public String cliente;  
    static private int fatture_emesse = 0;  
    public Fattura() {  
        . . .  
    }  
    // Altri metodi  
    ...  
}
```



- Introduzione
- Classi
- **Associazioni**
  - Aggregazione/Composizione
  - Ereditarietà
  - Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- In un Class Diagram vengono rappresentate relazioni oltre alle classi
  - associazioni (semplici, aggregazioni, composizioni)
  - relazioni di generalizzazione
  - relazioni di dipendenza
  - relazioni di raffinamento



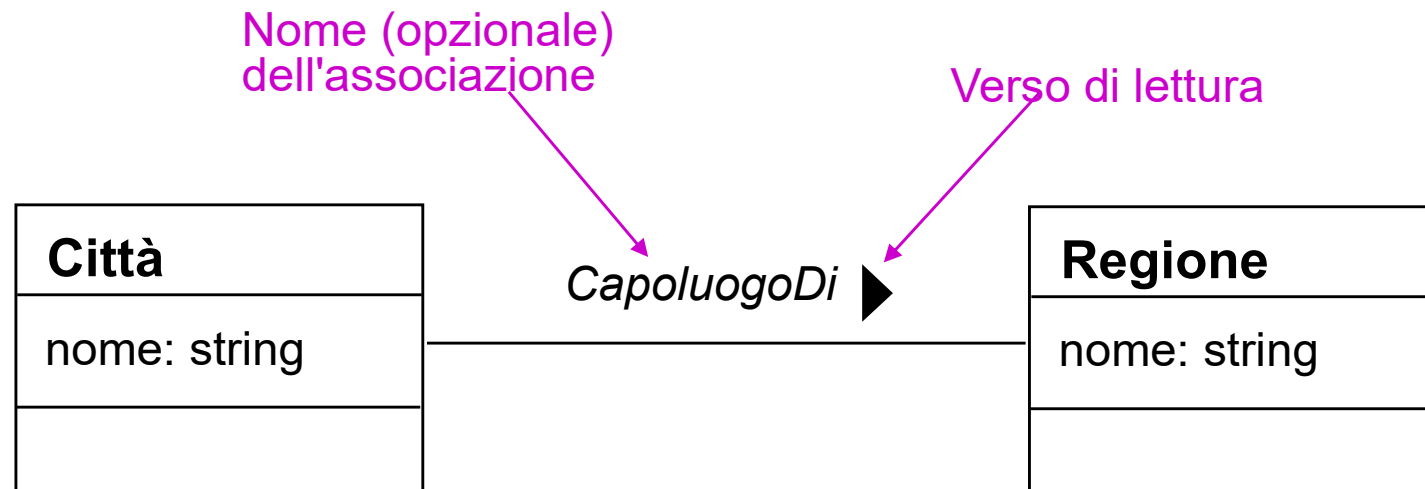
- Introduzione
- Classi
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- Un'associazione definisce un canale di comunicazione bidirezionale fra le due classi
- La molteplicità definisce il numero di istanze che prendono parte alla relazione
- I link sono istanze delle associazioni
  - un link connette due oggetti
  - un'associazione connette due classi



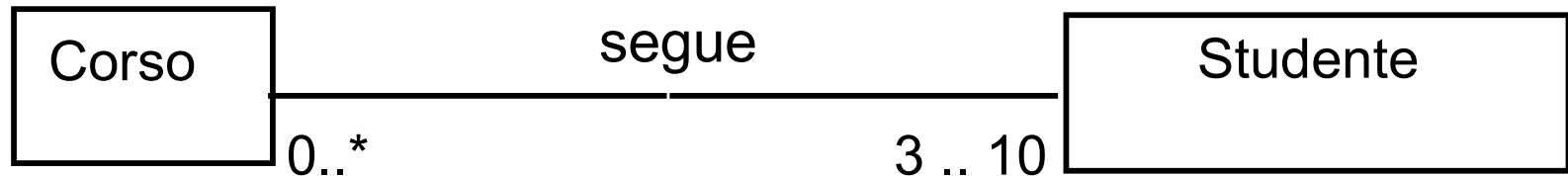
### ● Un'associazione

- individua una “connessione” logica tra classi
- si traduce in una connessione fra oggetti istanze delle classi coinvolte nell'associazione
- è bidirezionale (navigabile in ambo le direzioni)
- deve avere un nome, solitamente un verbo (un'etichetta al centro della linea che definisce l'associazione)



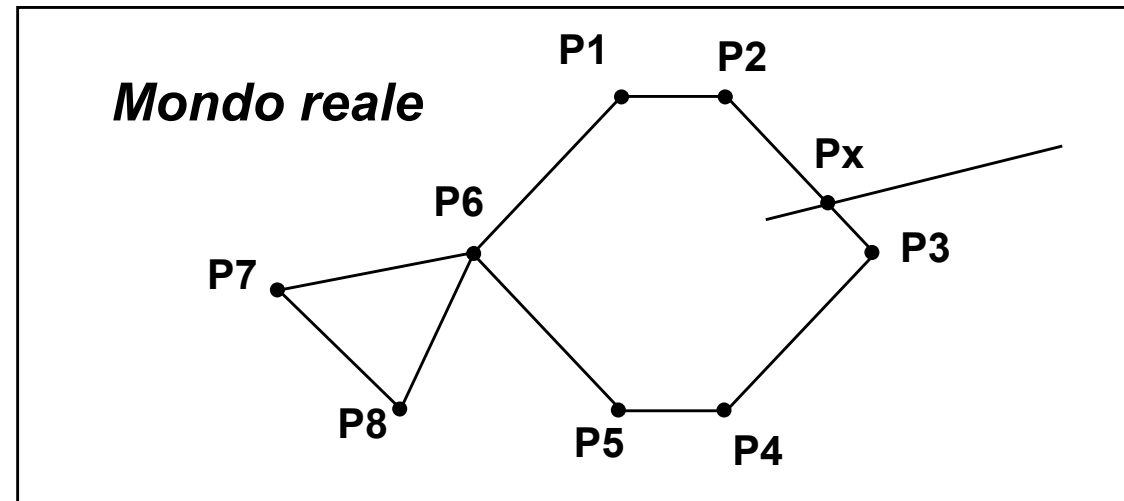


- La molteplicità dice
  - se l'associazione è obbligatoria oppure no
  - il numero minimo e massimo di oggetti che possono essere relazionati ad un altro oggetto
- Può non essere specificata ad uno degli estremi (“association-end”) o a entrambi
  - in questo caso non significa cardinalità pari a 1

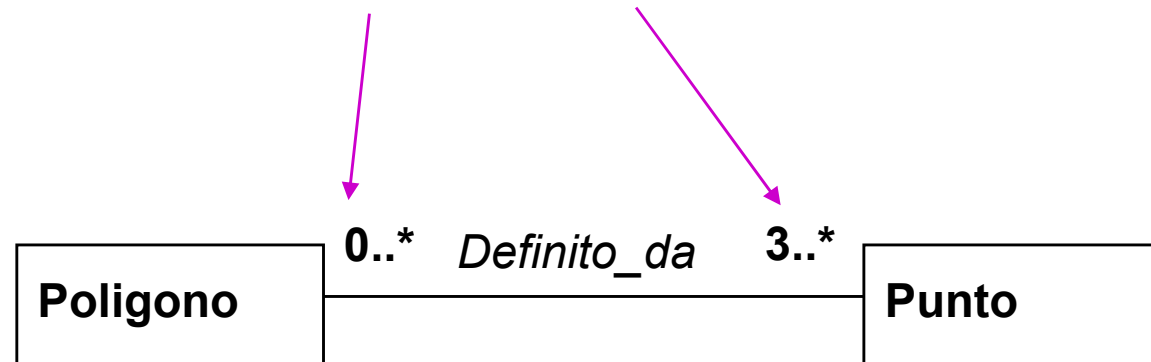




- Introduzione
- Classi
- **Associazioni**
  - Aggregazione/
  - Composizione
  - Ereditarietà
  - Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram



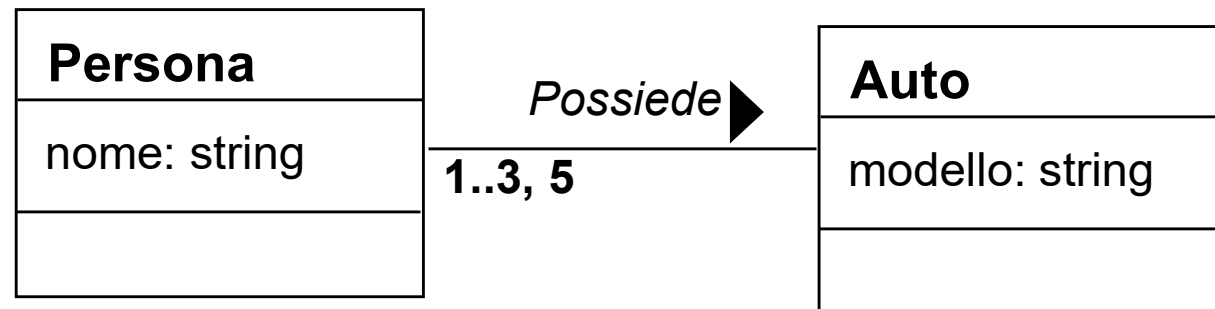
### Indicazione (opzionale) di molteplicità





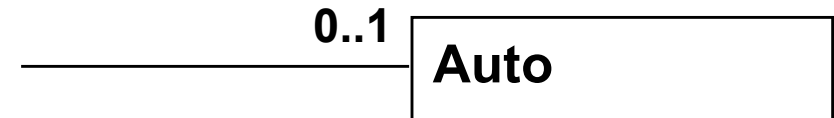


- Ogni persona può possedere zero o al più un'auto (magari in comproprietà)
- Ogni auto ha 1, 2, 3 o 5 (ma non 4) comproprietari





● Zero o un'auto



● Esattamente un'auto



● Zero o più auto

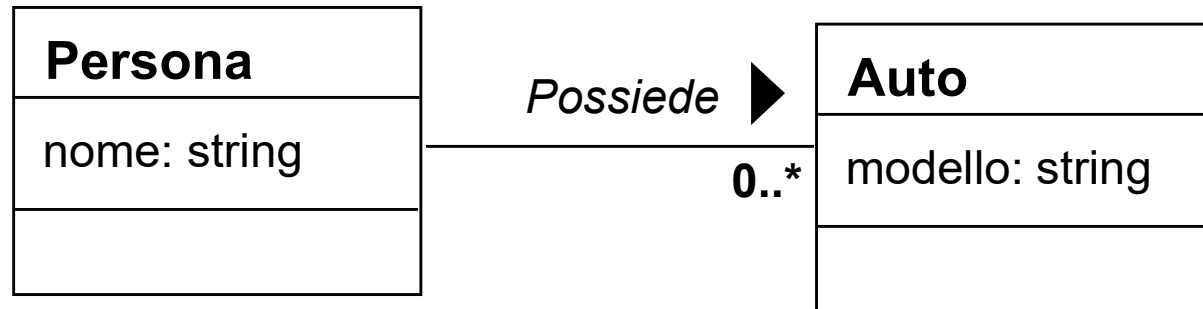


● Zero, da 3 a 5 oppure 7 auto e oltre





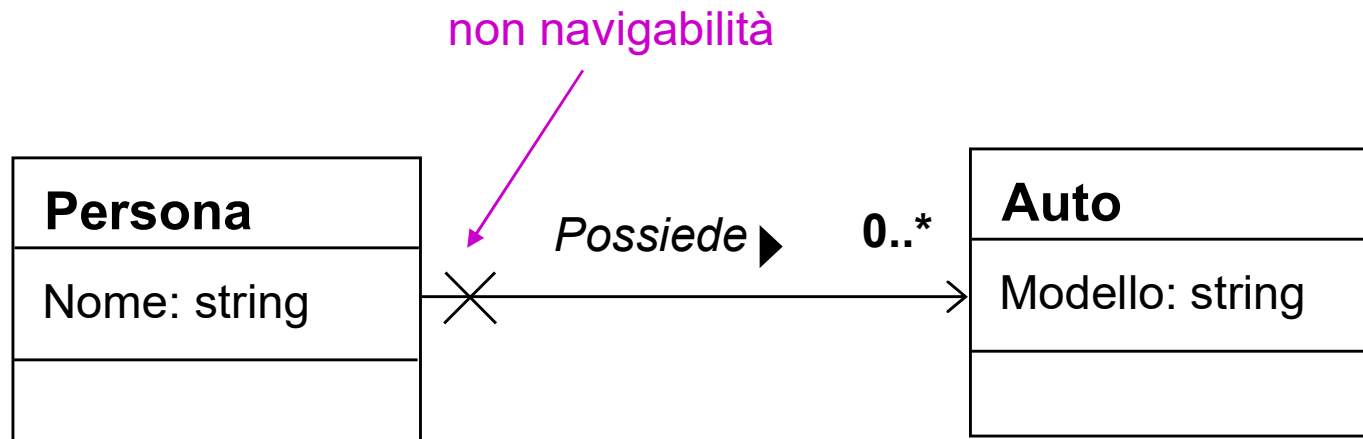
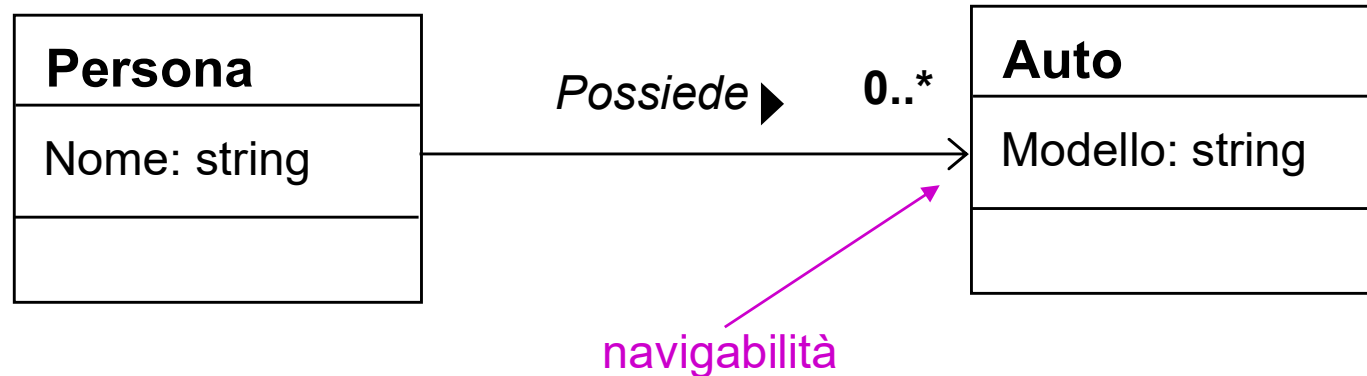
- Introduzione
- Classi
- **Associazioni**
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram



- Il class diagram indica che una persona può possedere un numero qualsiasi di auto
  - quanti sono i proprietari di una singola auto?
- Il diagramma non lo dice
  - la cardinalità della partecipazione di Persona all'associazione non è specificata



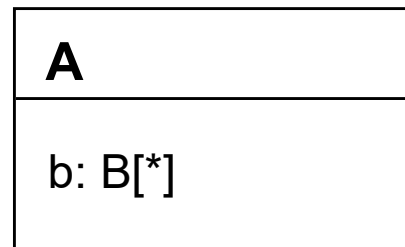
- Vogliamo indicare che l'associazione è navigabile in una sola direzione: dato un proprietario si trovano le sue auto, ma data un'auto non è mai possibile risalire al proprietario



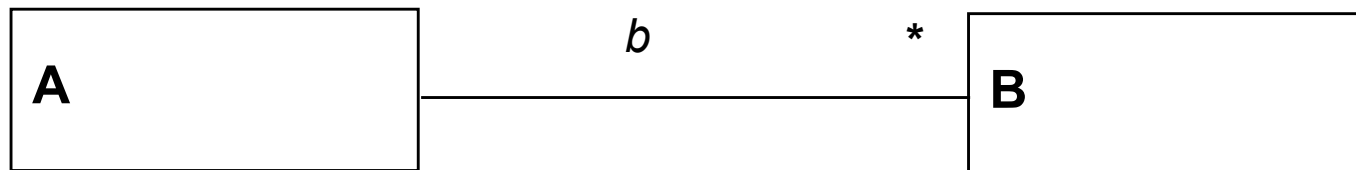


- Introduzione
- Classi
- Associazioni
  - Aggregazione/Composizione
  - Ereditarietà
  - Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- La notazione degli attributi può essere usata per indicare la terminazione di un'associazione posseduta da una classe (è un attributo, infatti)



equivale a





Introduzione

Classi

➤ Associazioni

Aggregazione/

Composizione

Ereditarietà

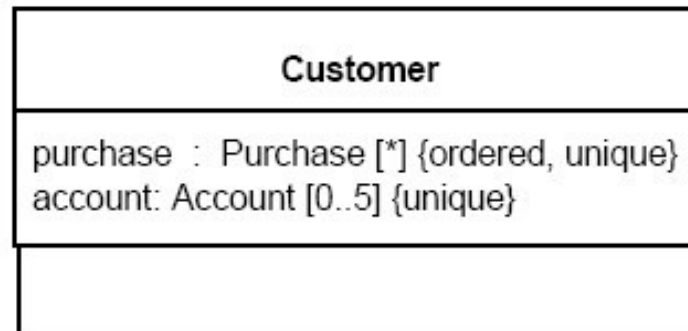
Vincoli

Interfacce

Ulteriori

caratteristiche

Object diagram

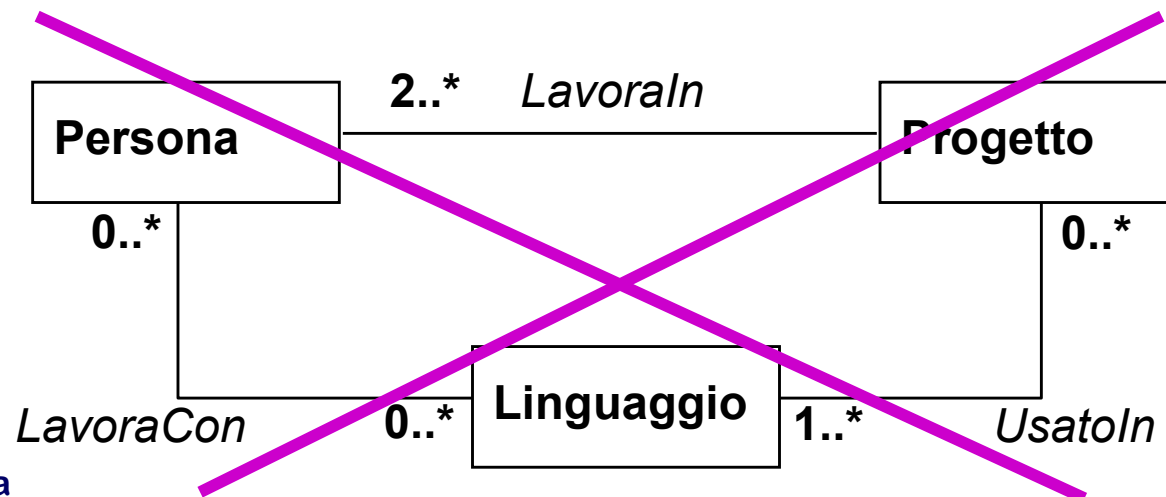
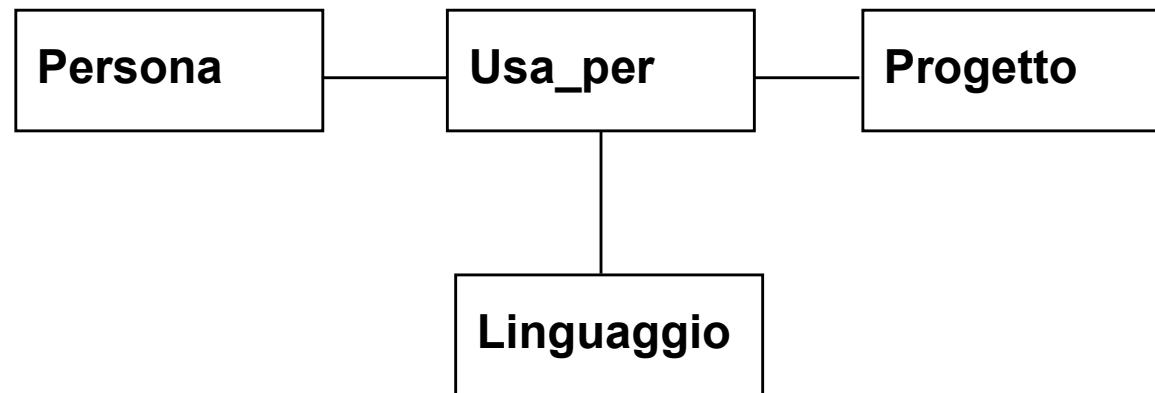


Molteplicità in una specifica testuale



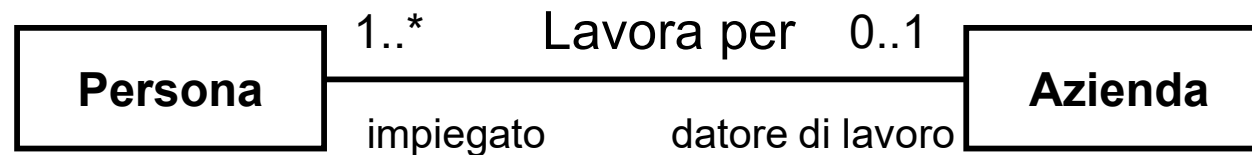


- La maggior parte degli strumenti CASE non supportano le relazioni ternarie
  - si introduce una classe che rappresenta la relazione





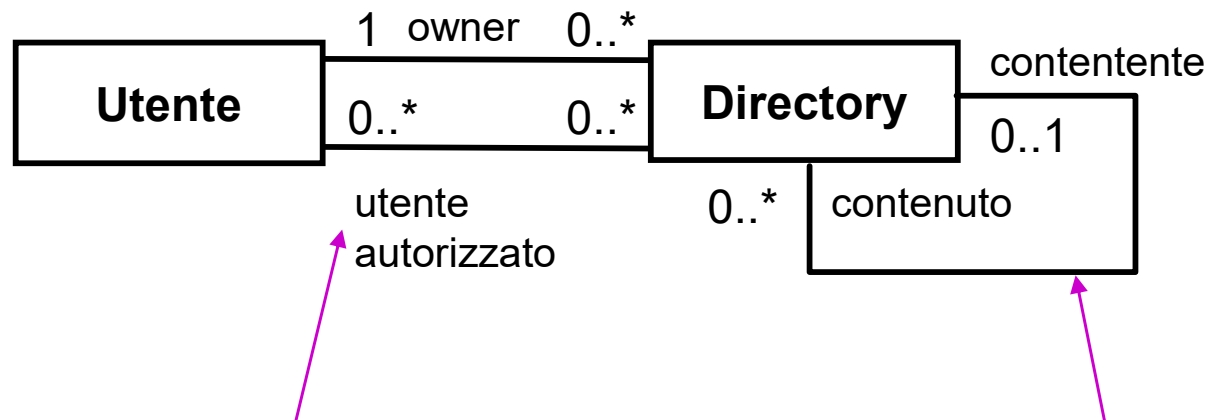
- Una classe può partecipare ad un'associazione con un ruolo specifico, che può essere indicato
- Utile/obbligatorio per
  - auto-associazioni
  - associazioni multiple tra due classi







- Quando le stesse classi sono coinvolte più volte dalla stessa associazione il ruolo diviene obbligatorio



Qui il ruolo serve a distinguere due associazioni diverse tra le stesse due classi

Qui il ruolo è obbligatorio

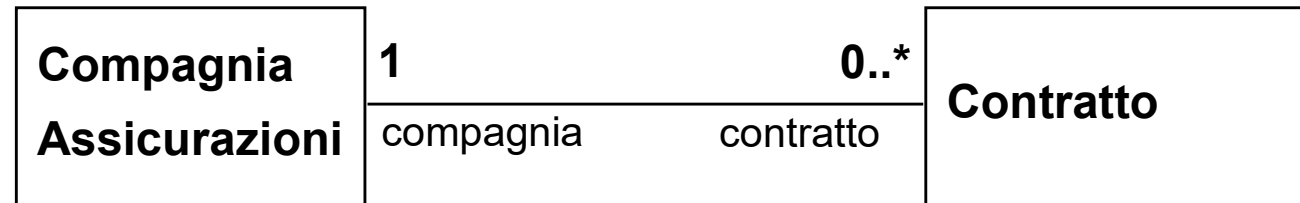


- Introduzione
- Classi
- Associazioni
  - Aggregazione/Composizione
  - Ereditarietà
  - Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- In generale non esiste un mapping diretto su un costrutto di un linguaggio di programmazione
  - possono essere implementate con attributi che contengono riferimenti (o puntatori) alle istanze delle classi associate
  - il riferimento a più istanze di una classe può essere modellato con liste, array, etc.
- Direzionalità
  - si può decidere di codificarla solo in una delle due direzioni (con un puntatore), perdendo però così la possibilità di percorrere il link nella direzione opposta
- Nota: un'associazione va modellata come tale in UML!!
  - nascondere le associazioni nelle classi (ad es. come attributi puntatori) porta alla costruzione di diagrammi contenenti dipendenze nascoste difficili da gestire



- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

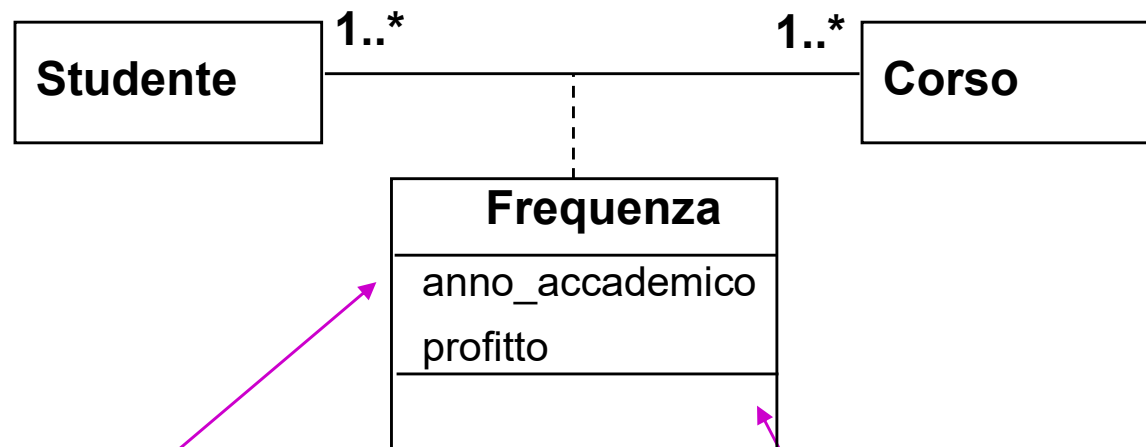


```
public class CompagniaAssicurazioni {
    private Vector contratti;
    /* In alternativa si può usare un array: Contratto[] contratti */
    ...
}

public class Contratto {
    private CompagniaAssicurazioni compagnia;
    ...
}
```



- In molti casi alcune proprietà sono proprie dell'associazione piuttosto che delle classi coinvolte



Questa è la notazione per indicare che un'associazione possiede alcuni attributi

Non si tratta di attributi dello studente perché cambiano da corso a corso.

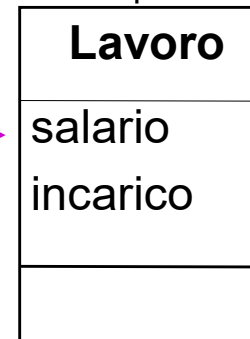
Né sono attributi del corso (ad es. ogni corso è frequentato da studenti diversi in anni diversi)



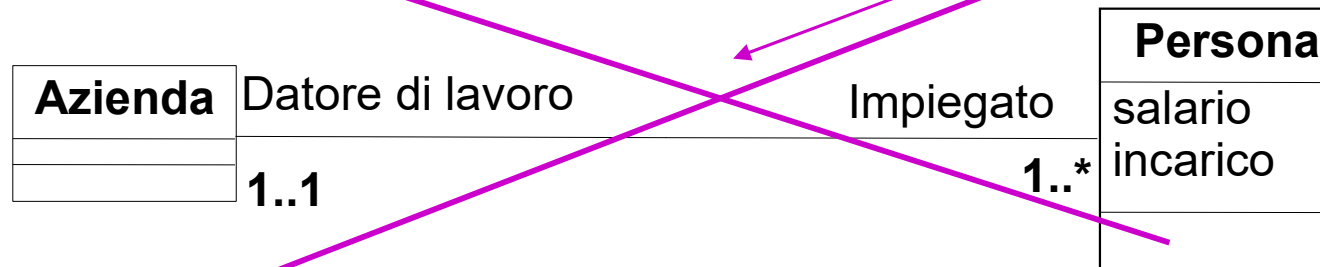
- Se l'associazione è 1-a-molti gli attributi dell'associazione si possono modellare come attributi della classe



Modello consigliato

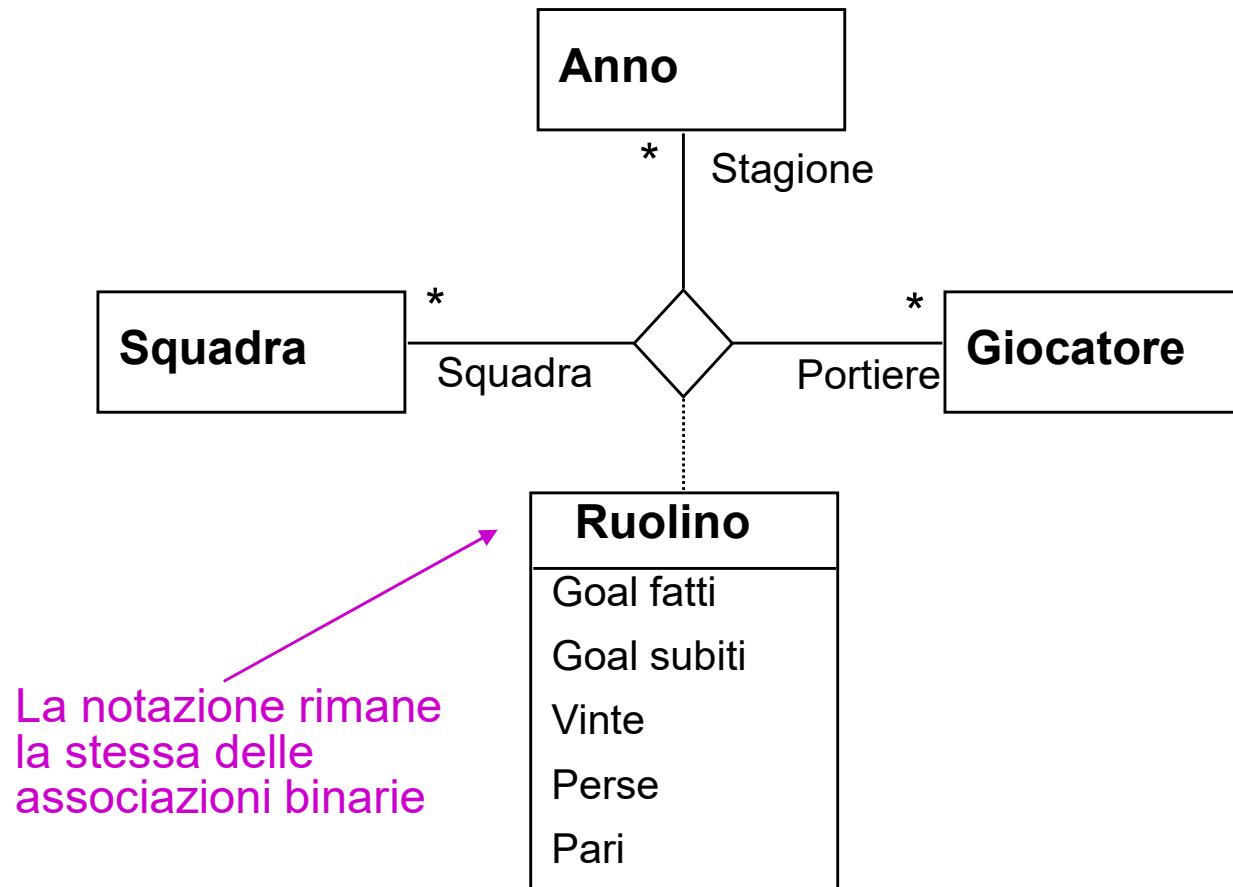


Modello sconsigliato



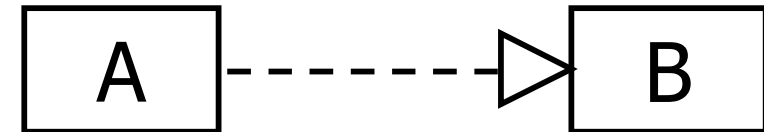
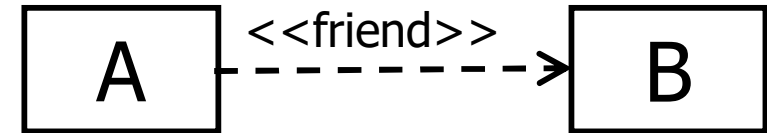


- Per associazioni n-arie la notazione rimane la stessa delle associazioni binarie





- Classi friend
  - A è amico di B
- Dipendenza
  - A dipende da B
- Realizzazione/Raffinamento
  - A raffina B
- Altri stereotipi
  - Call
  - Instantiate
  - ...





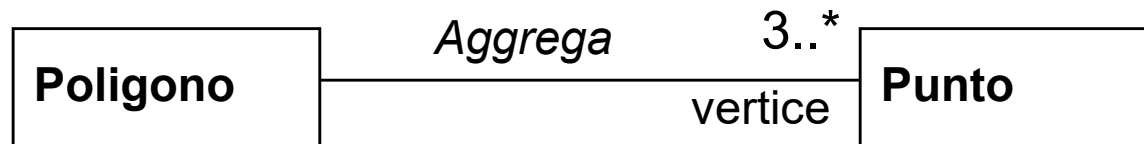
- Le aggregazioni sono una forma particolare di associazione
  - una classe aggrega oggetti di altre classi

Il rombo “vuoto” si legge “è un insieme di”



Essendo un tipo particolare di associazione è sempre possibile specificare la cardinalità

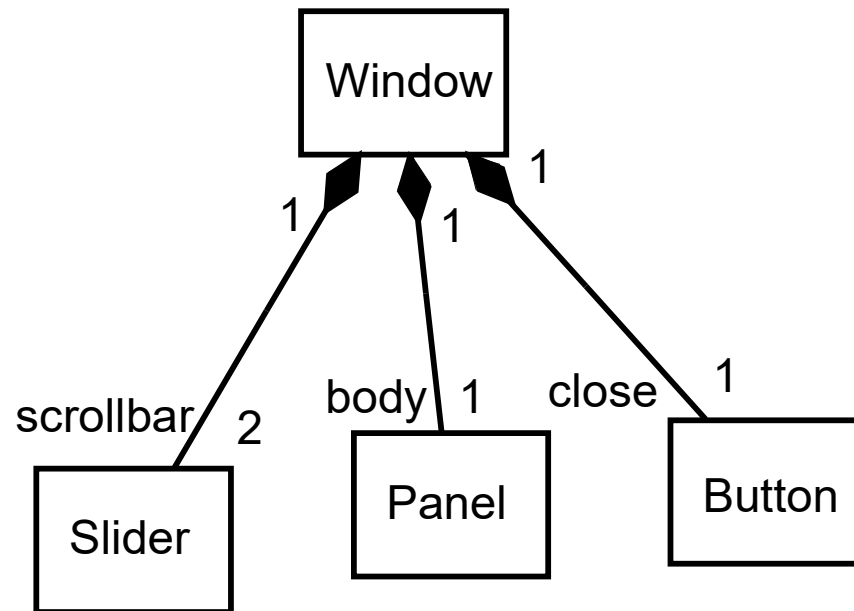
- Modello equivalente che usa una notazione convenzionale







- Una relazione di composizione è un'aggregazione forte
  - le parti componenti non esistono senza il contenitore
  - la proprietà da parte del contenente è esclusiva
    - la molteplicità dal lato dell'aggregato non può essere  $>1$
    - può essere qualsiasi per gli elementi componenti



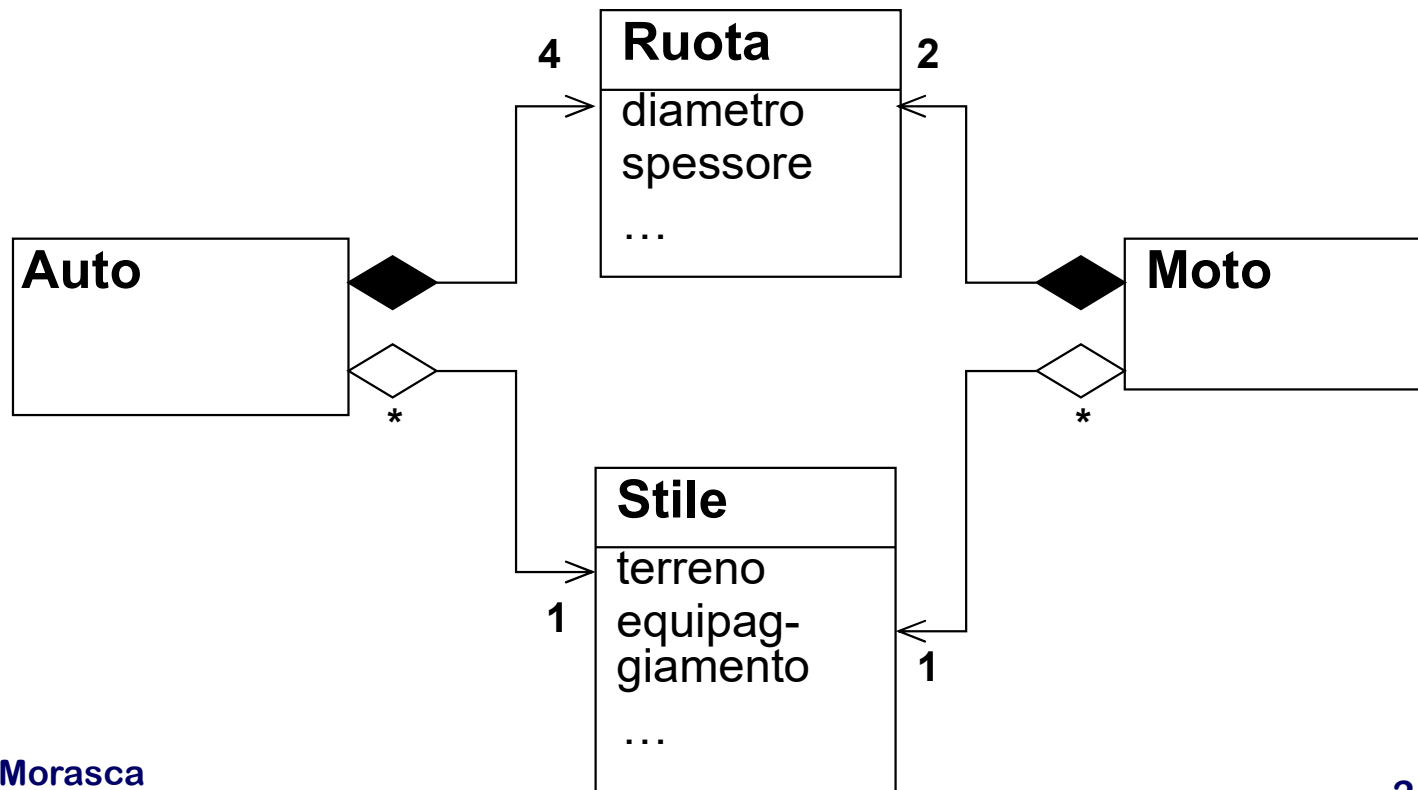


# Aggregazione vs. composizione

## UML – Class Diagram

- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- Un'istanza della classe Ruota non può far parte sia di un'Auto sia di una Moto. La cancellazione di un'Auto viene estesa a ogni sua Ruota, ma non allo Stile associato
- Un'istanza della classe Stile può, invece, essere condivisa tra un'Auto e una Moto





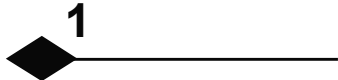
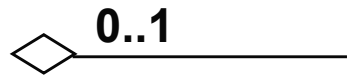

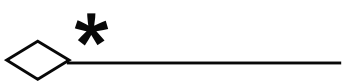
- Introduzione
- Classi
- Associazioni
- Aggregazione/  
Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram

- **Transitiva**
  - se un PC contiene un'unità base e questa contiene la CPU, allora il PC contiene la CPU
- **Asimmetrica**
  - se un PC contiene un'unità base, questo non implica che l'unità base contenga il PC
- **Propagazione degli effetti (solo per la composizione)**
  - se copio un documento composto di capitoli, devo copiare anche i singoli capitoli
  - se distruggo un PC, distruggo anche tutte le sue componenti



- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

### Dipendenza delle parti dall'oggetto composto

Proprietà	dipendente	indipendente
esclusiva		
condivisa		

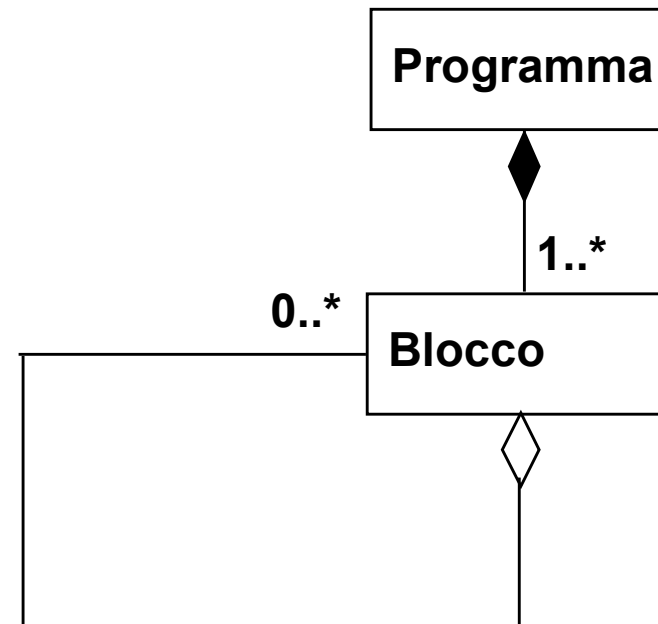
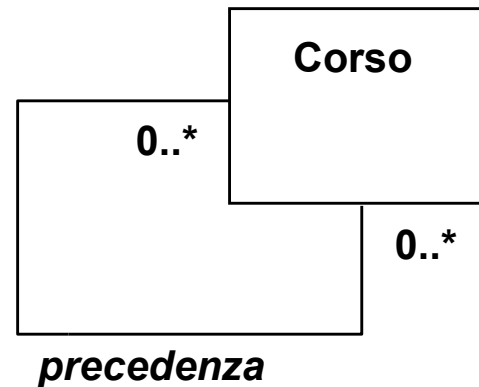


- Introduzione
- Classi
- Associazioni
- Aggregazione/  
Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram

- L'aggregazione è un caso particolare di associazione. Come distinguerle?
  - Part of (contenimento fisico vs. semplice riferimento)?
    - le parti sono fisicamente contenute nell'oggetto composito
    - l'associazione comporta il semplice mantenimento di riferimenti tra oggetti
  - transitiva ed asimmetrica?
  - c'è propagazione di attributi?
  - operazioni sull'intero sono automaticamente applicate alle parti?
  - ciclo di vita
    - l'oggetto composito deve gestire la creazione e la distruzione delle parti
    - non c'è dipendenza esistenziale tra le parti nell'associazione

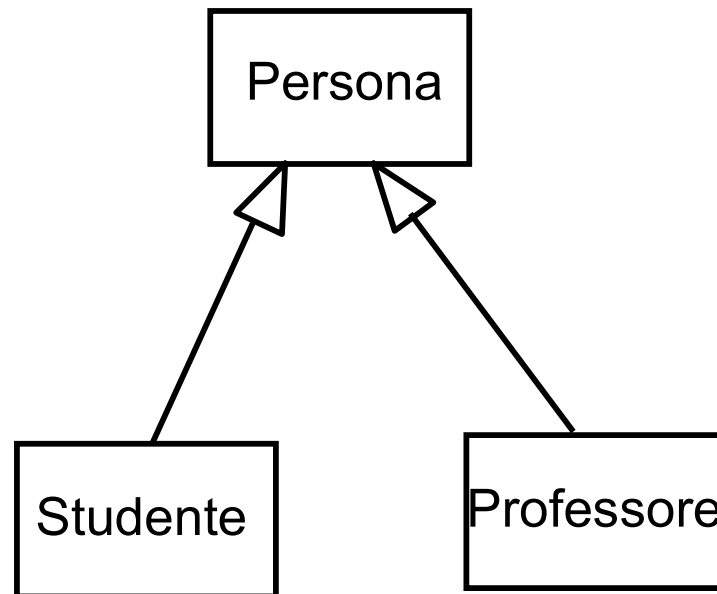


- Un'associazione è riflessiva se coinvolge oggetti della stessa classe
- Indica che oggetti multipli della stessa classe sono in relazione fra loro





- Esplicita eventuali comportamenti comuni
- È possibile dover:
  - aggiungere nuove proprietà alle classi
  - ridefinire/modificare operazioni esistenti

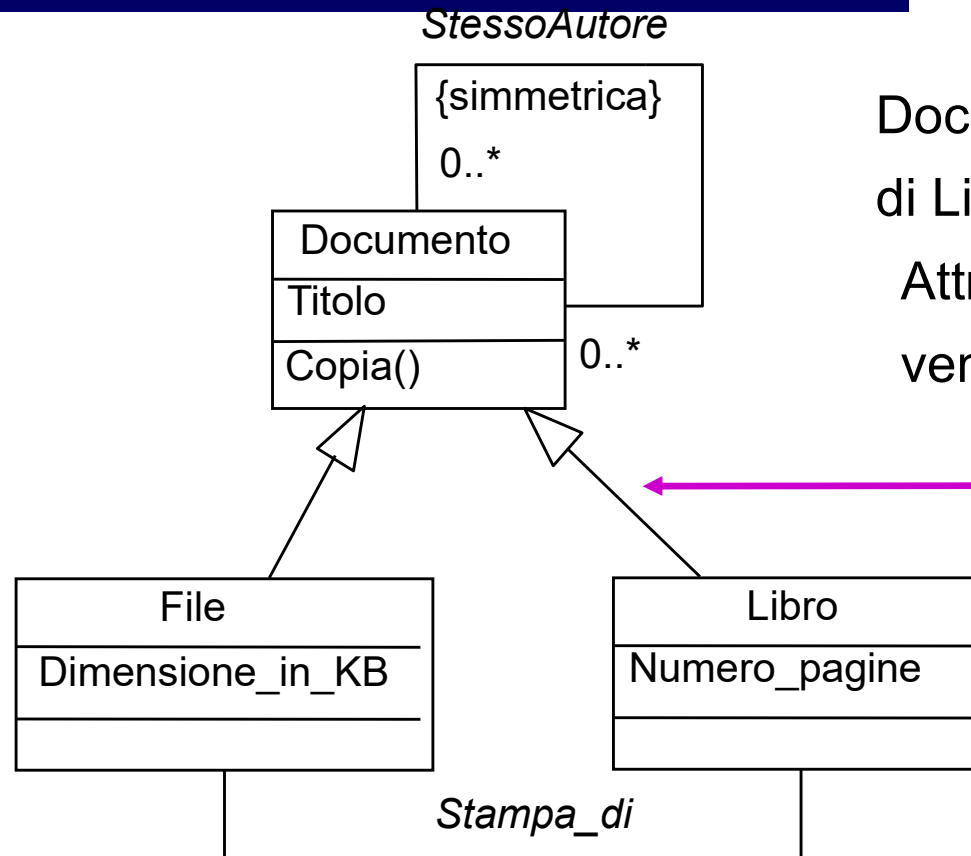




# Ereditarietà (Generalizzazione)

## UML – Class Diagram

- Introduzione
- Classi
- **Associazioni**
- Aggregazione/Composizione
- **Ereditarietà**
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram



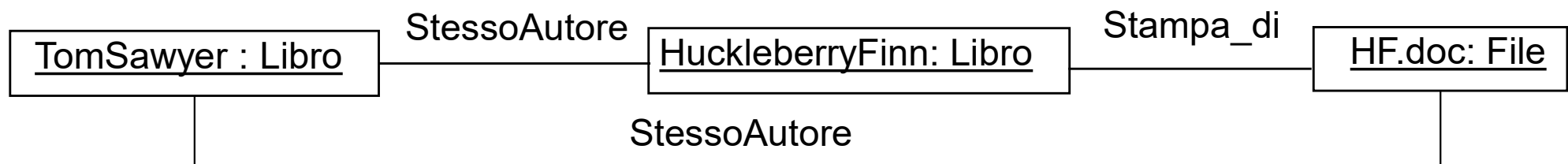
Documento è una generalizzazione di Libro e di File

Attributi, operazioni e associazioni vengono ereditati dalle sottoclassi

generalizzazione

File e libro sono sottoclassi di documento

### Esempio di Object Diagram



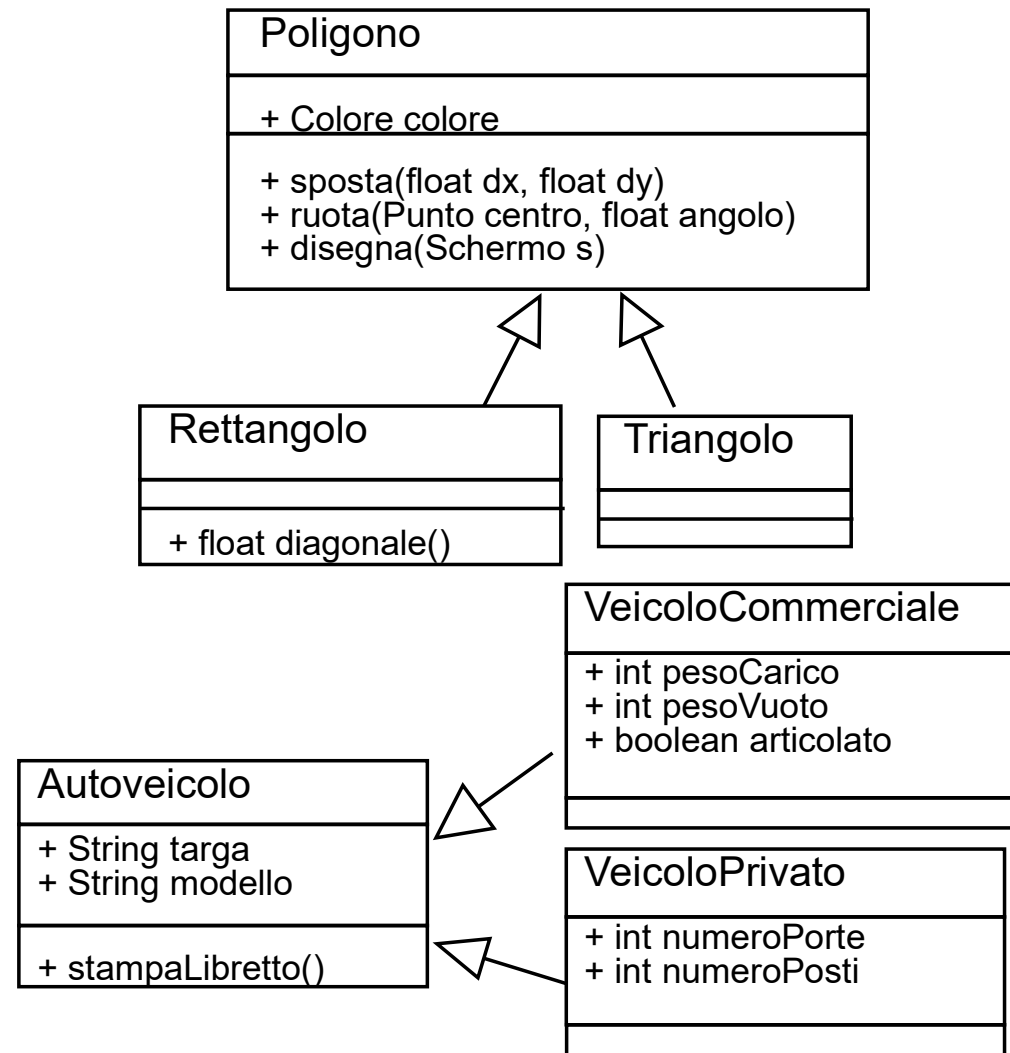




# Ereditarietà (Generalizzazione)

## UML – Class Diagram

- Introduzione
- Classi
- **Associazioni**
- Aggregazione/  
Composizione
- **Ereditarietà**
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram





# Aggregazione/composizione vs generalizzazione

## UML – Class Diagram

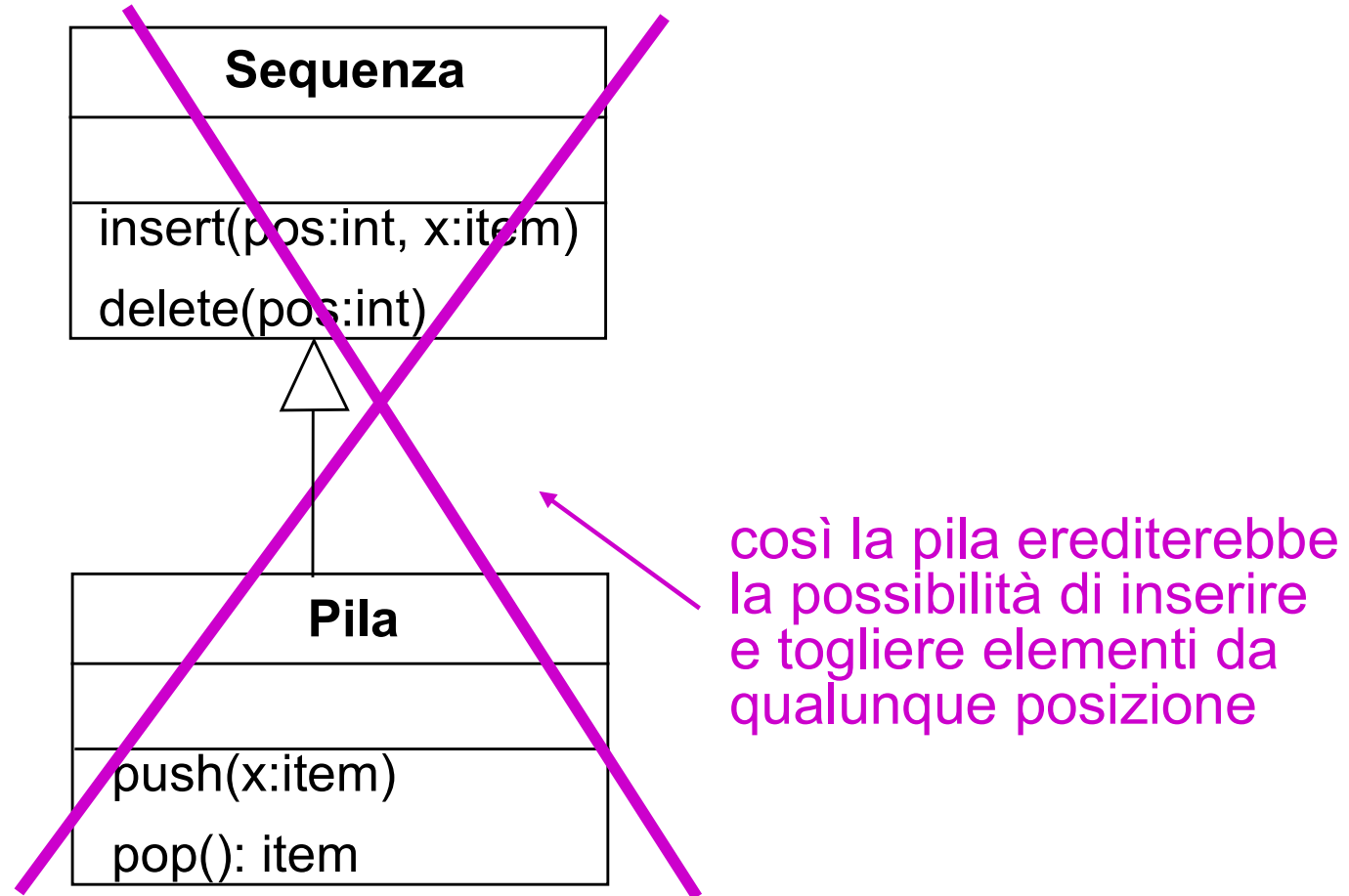
- Introduzione
- Classi
- **Associazioni**
  - Aggregazione/  
Composizione
- **Ereditarietà**
  - Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram

- L'aggregazione riguarda oggetti, la generalizzazione classi
- "part-of" vs "is-a"
- "and-relationship" vs "or-relationship"



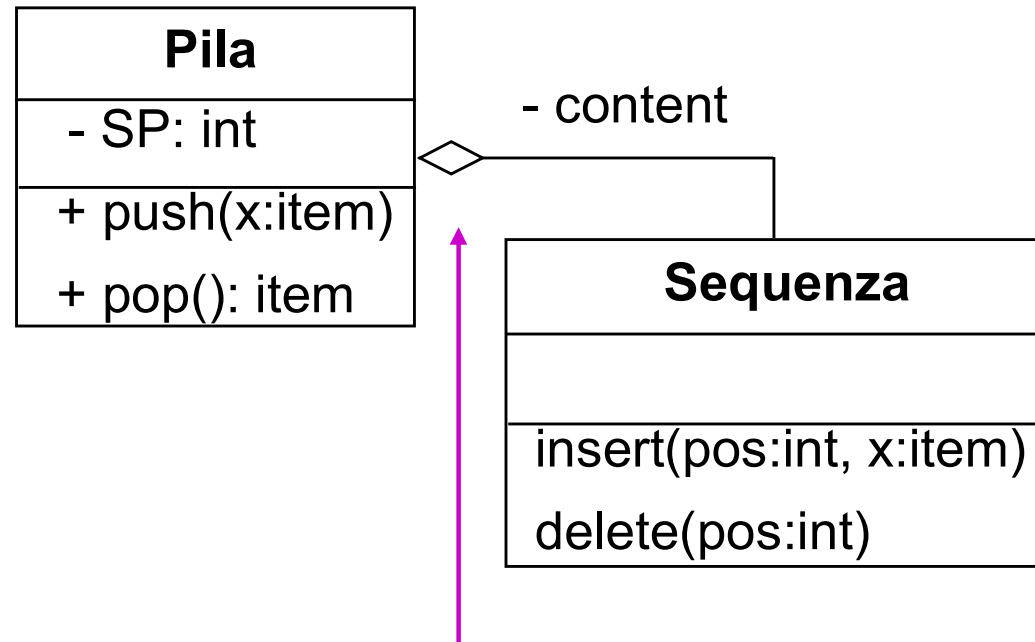
- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- Devo definire la classe pila (col solito significato)
- Dispongo di una classe Sequenza, che vorrei riutilizzare





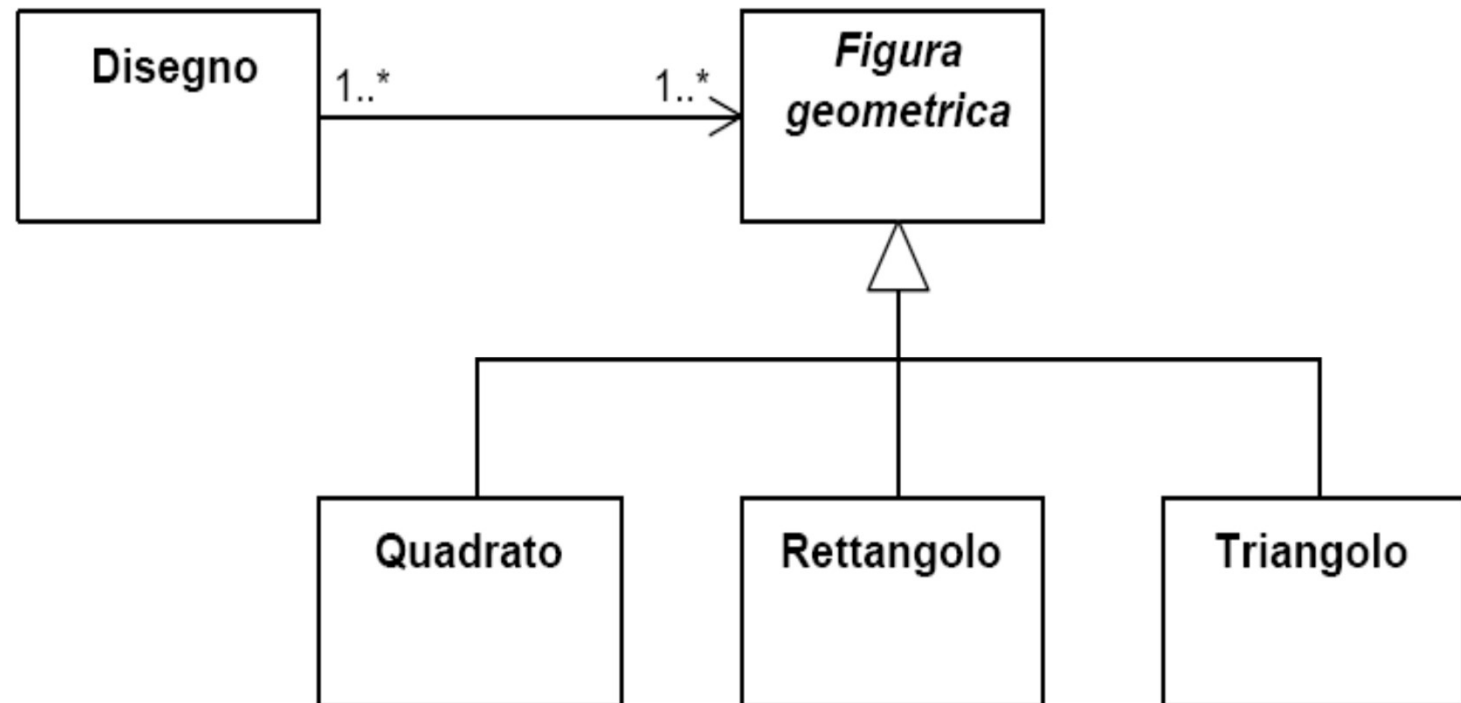
- Introduzione
- Classi
- **Associazioni**
  - Aggregazione/Composizione
- **Ereditarietà**
  - Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram



così la pila contiene  
(tipicamente nella  
parte privata) una  
sequenza di elementi

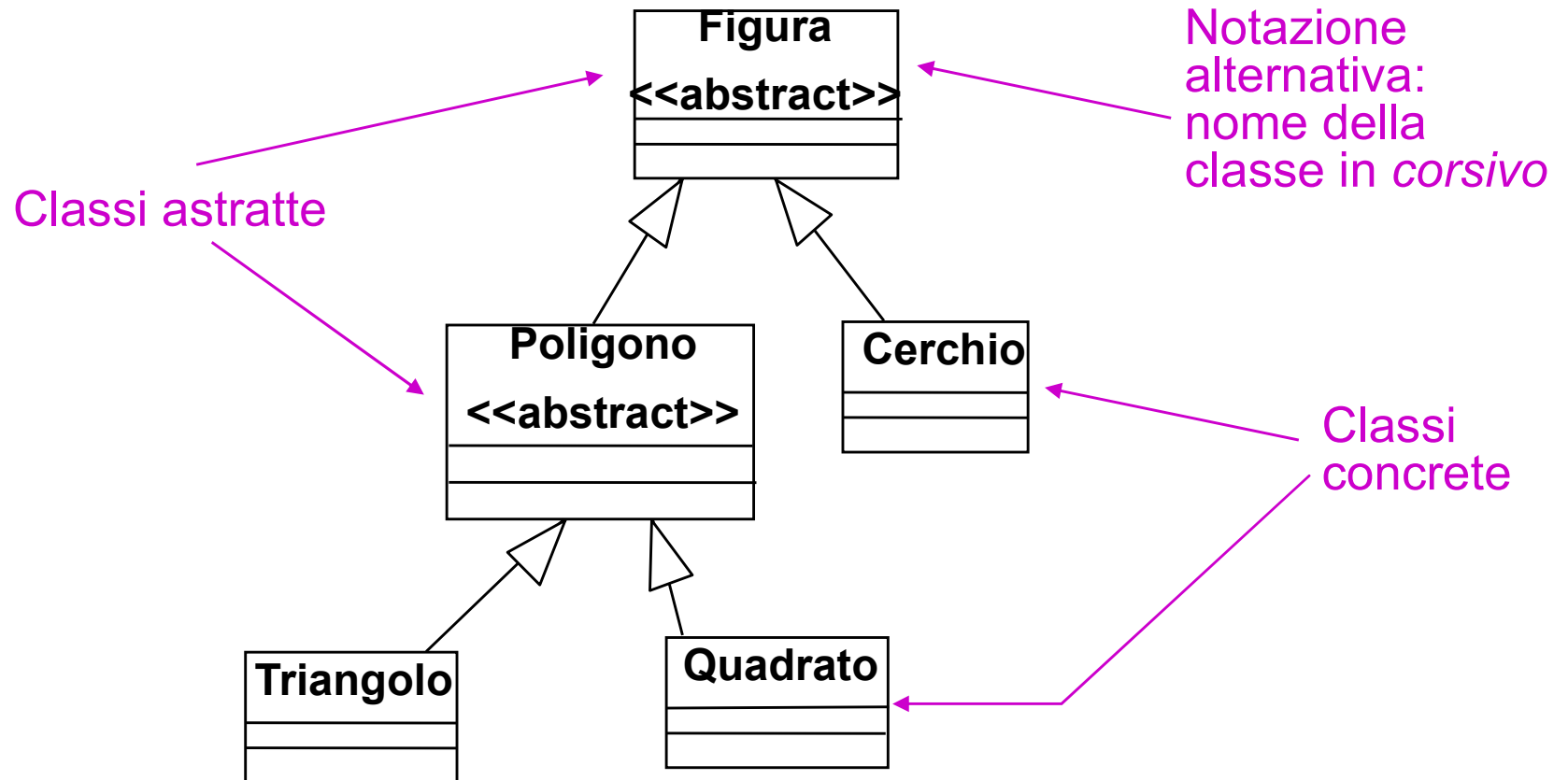


- Una classe astratta è una classe che non può essere direttamente istanziata
  - può avere sottoclassi concrete o astratte, ma almeno una concreta





- Introduzione
- Classi
- Associazioni
- Aggregazione/  
Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram



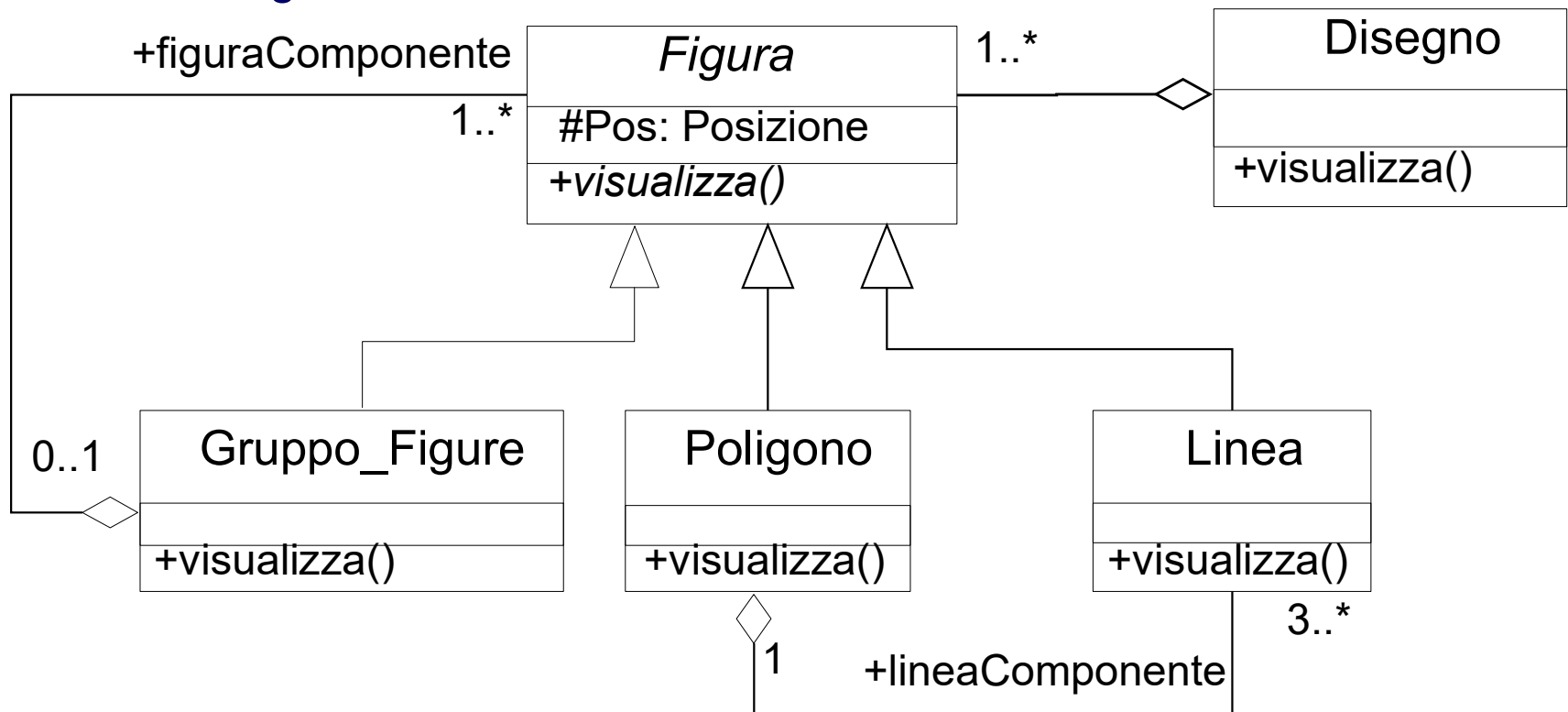


- Introduzione
- Classi
- Associazioni
- Aggregazione/  
Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram

- Il termine “astratto” viene anche utilizzato per descrivere un’operazione per la quale non è stata definita una implementazione
  - le operazioni astratte di una classe si rappresentano scrivendo il nome in corsivo
- Una classe astratta può avere operazioni concrete
- Tipicamente vengono usate
  - per mettere a fattor comune un'astrazione di un certo tipo
  - per favorire il riuso



- Un disegno è composto di figure. Non importa come sono fatte. Il disegno sa solo che per disegnare (visualizza) se stesso può chiamare il metodo visualizza delle figure
  - aggiungendo una sottoclasse a figura il codice di gestione dei disegni non cambia







Introduzione  
Classi

➤ **Associazioni**

Aggregazione/  
Composizione

➤ **Ereditarietà**

Vincoli

Interfacce

Ulteriori

caratteristiche

Object diagram

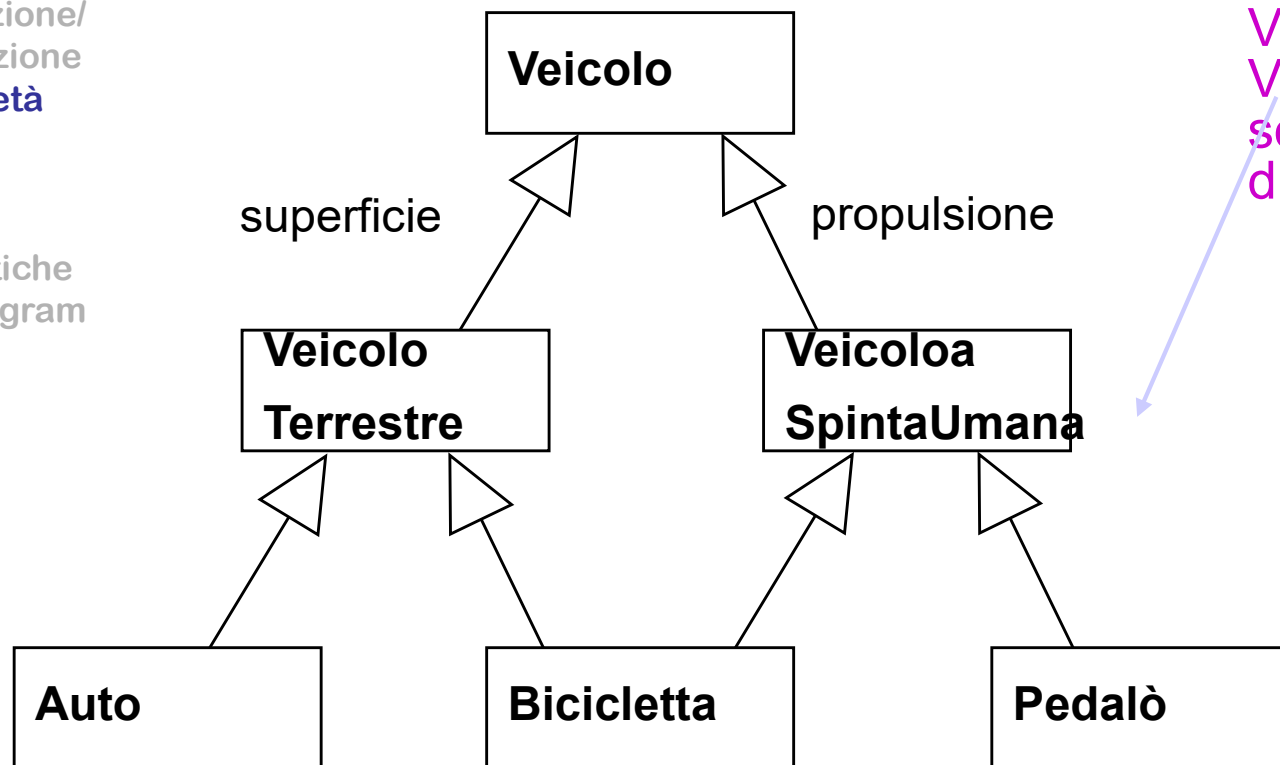
```
public abstract class Figura {
    public abstract void draw();
    protected Pos position;
}

public class Gruppo_Figura extends Figura {
    public FiguraVector figuraPart; /* Un Vector contenente solo
        istanze di Figura */
    public void draw() {
        for(int i=0; i< figuraPart.size(), i++)
            figuraPart.get(i).draw();
    }
}

public class Poligono extends Figura {
    public LineaVector lineaPart;
    public void draw() {
        for(int i=0; i< figuraPart.size(), i++)
            lineaPart.get(i).draw();
    }
}
```



- Introduzione
- Classi
- Associazioni
  - Aggregazione/Composizione
- Ereditarietà
  - Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram



VeicoloTerrestre e VeicoloSpintaUmana sono sottoclassi non disgiunte di Veicolo

Bicicletta eredita sia da VeicoloTerrestre, sia da VeicoloSpintaUmana



- Introduzione
- Classi
- Associazioni
- Aggregazione/  
Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori  
caratteristiche
- Object diagram

- Alcuni attributi, associazioni o operazioni possono essere ereditati più volte da una certa classe
  - Problemi
    - Ridondanza: ci può essere spreco di memoria a causa di campi o legami duplicati
    - Contraddizione: possono arrivare varianti diverse della stessa operazione, attributo o associazione
    - Multiple inheritance accidentale
      - Esempio: definisco Assistente come sottoclasse di Studente e MembroDiFacoltà
      - Cosa succede se un professore dell'Insubria è studente alla Bocconi ? Non per questo diventa un assistente



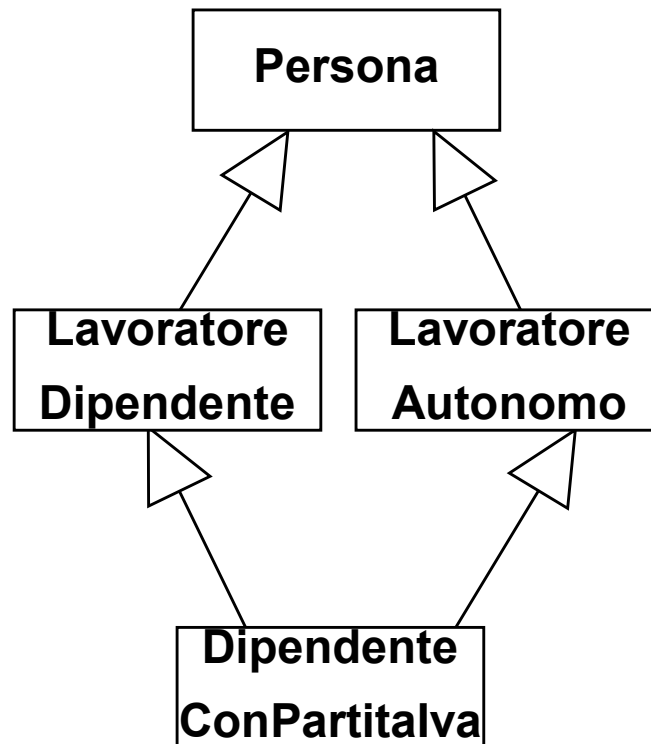
- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- Soluzioni (combinabili)
  - semplificare il modello
  - usare una classificazione nidificata
  - usare l'eredità multipla, ma inibire l'eredità di particolari attributi o operazioni (bloccare con gli strumenti del linguaggio l'eredità di ciò di cui non si necessita)
  - usare la Delega invece della relazione classe/sottoclasse
- Per evitare eredità multipla accidentale
- Per anticipare problemi di implementazione (non tutti i linguaggi supportano l'ereditarietà multipla)

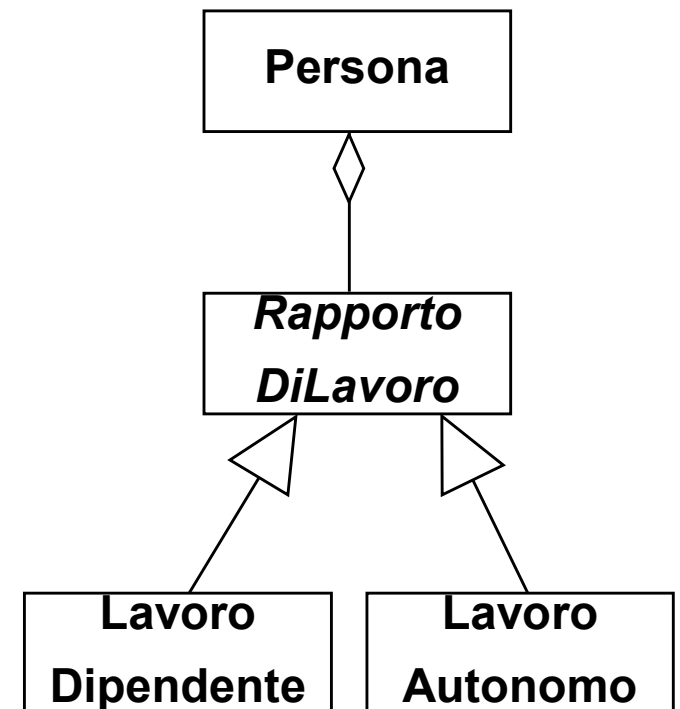


- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- La delega consiste nel "delegare" una entità apposita (es. **RapportoDiLavoro**) a svolgere particolari operazioni.



Senza delega (eredità multipla)

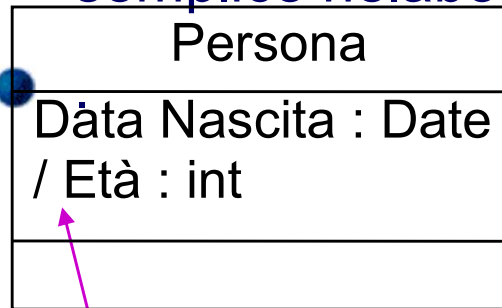


Con delega (eredità semplice)



- Introduzione
- Classi
- Associazioni
  - Aggregazione/Composizione
  - Ereditarietà
- Vincoli
  - Interfacce
  - Ulteriori caratteristiche
  - Object diagram

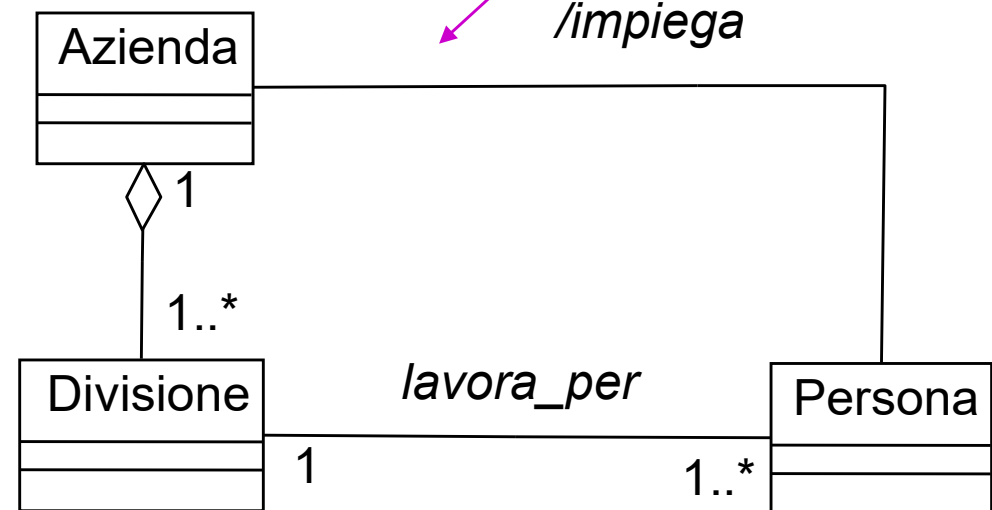
- Alcune associazioni e alcuni attributi sono in realtà una semplice rielaborazione di altri valori e/o legami



{Età = Data Corrente - Data Nascita}

Gli attributi i cui valori sono calcolati in base agli altri sono identificati da una barra ( / )

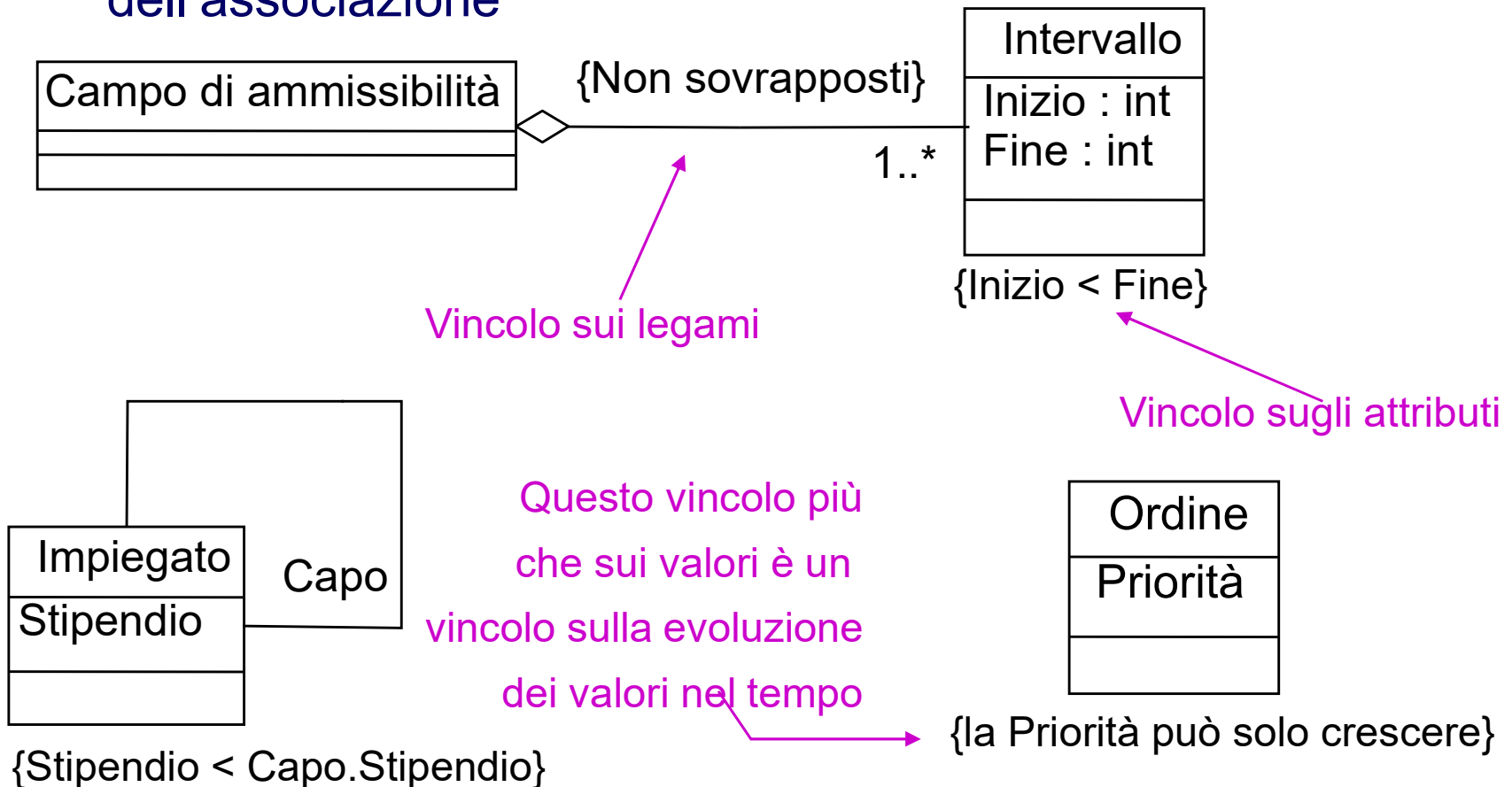
Le associazioni i cui legami sono calcolabili in base ad altri legami o ai valori degli attributi sono identificati da una barra trasversale





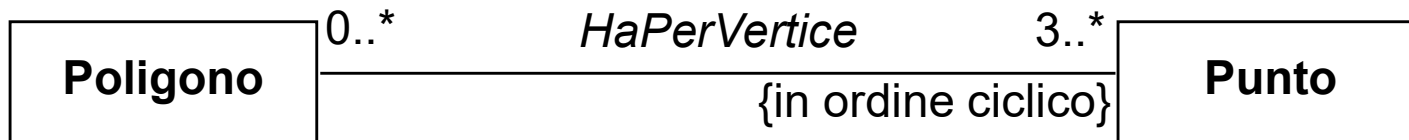
- Introduzione
- Classi
- Associazioni
- Aggregazione/Composizione
- Ereditarietà
- **Vincoli**
- Interfacce
- Ulteriori caratteristiche
- Object diagram

- Si possono porre vincoli e restrizioni sui legami e sul valore degli attributi scrivendoli tra graffe a margine della classe o dell'associazione

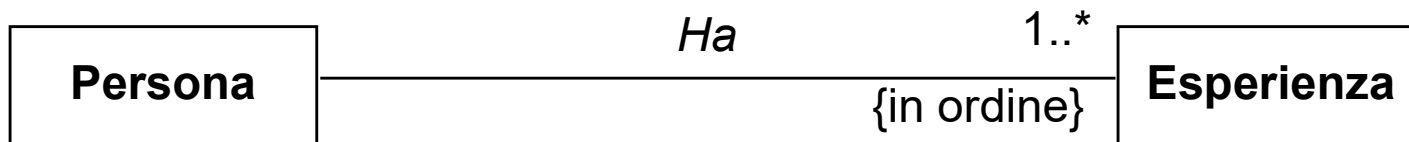




- In generale le associazioni "a molti" considerano gli oggetti legati ad uno come se fossero un insieme non ordinato
  - i vincoli permettono di fissare un certo tipo di ordinamento



La sequenza dei vertici lungo il perimetro



La sequenza cronologica nel curriculum



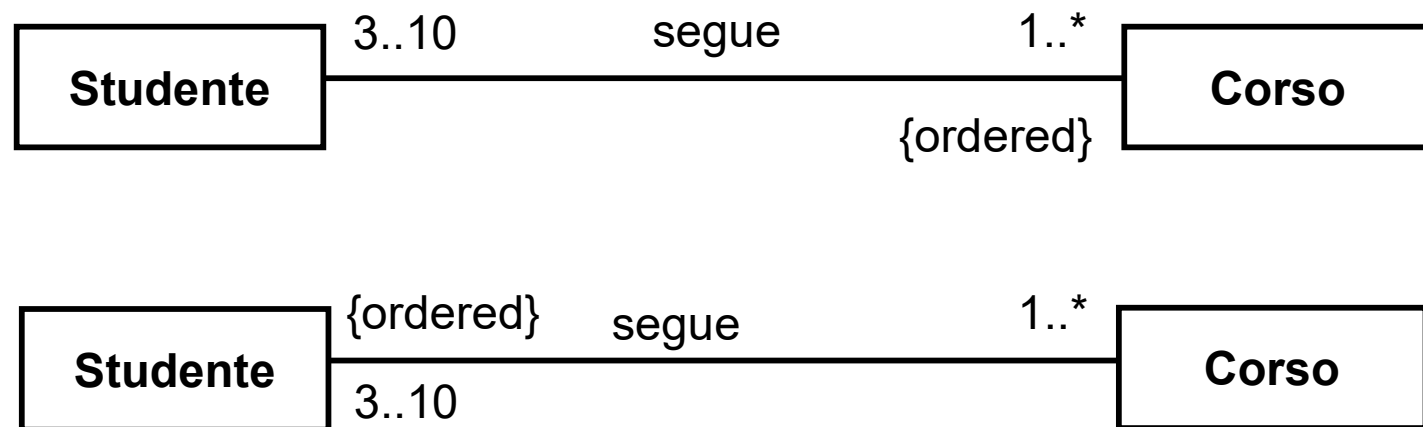


# Vincoli (associazioni singole)

## UML – Class Diagram

- Introduzione
- Classi
- Associazioni
  - Aggregazione/
  - Composizione
  - Ereditarietà
- **Vincoli**
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- I vincoli consentono di fissare il tipo di ordinamento degli oggetti dal lato “molti” di un’associazione



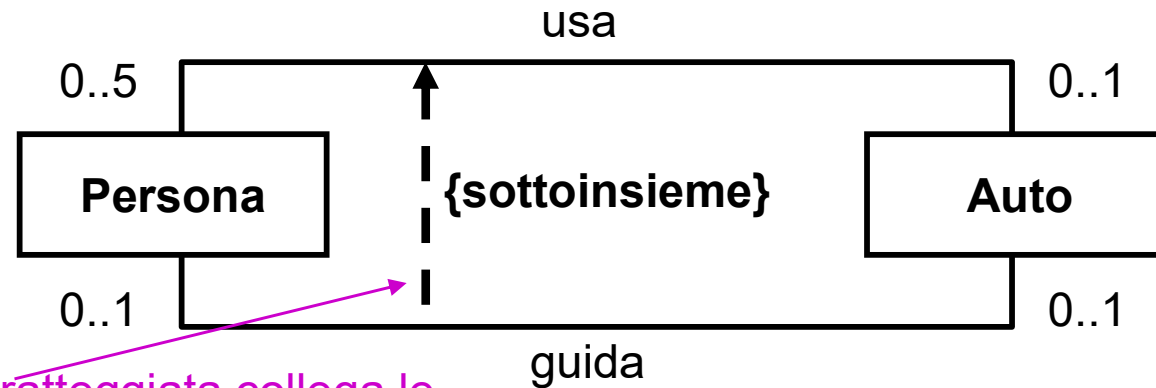


# Vincoli (associazioni multiple)

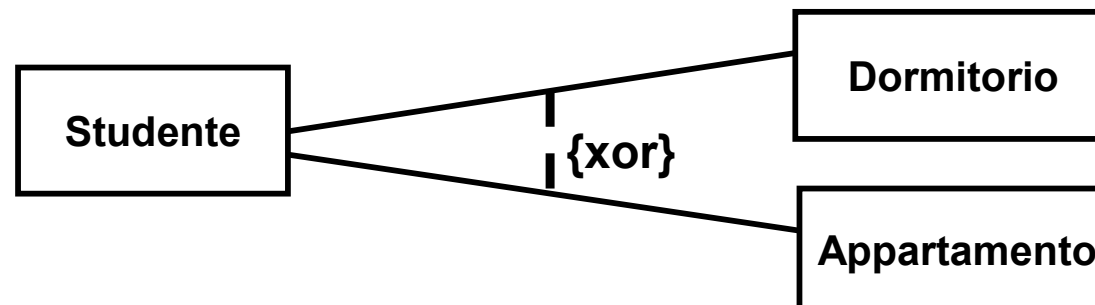
## UML – Class Diagram

- Introduzione
- Classi
- Associazioni
  - Aggregazione/
  - Composizione
  - Ereditarietà
- Vincoli
  - Interfacce
  - Ulteriori
  - caratteristiche
  - Object diagram

- Le associazioni possono avere vincoli che condizionano un'associazione rispetto ad un'altra



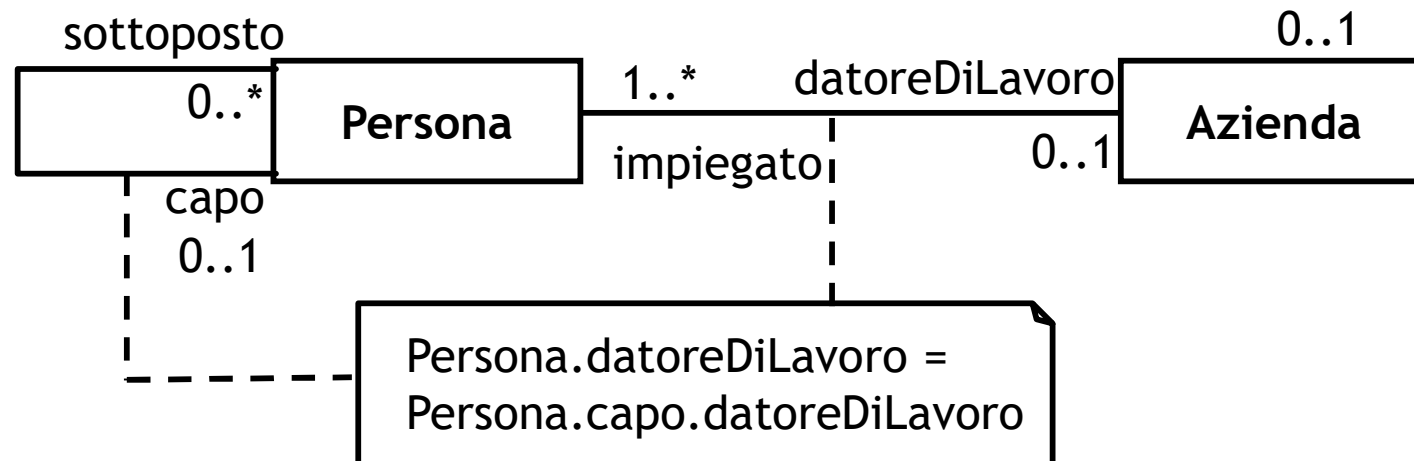
Una freccia tratteggiata collega le associazioni soggette al vincolo e un'etichetta fra parentesi graffe indica quale sia il vincolo





- Introduzione
- Classi
- Associazioni
  - Aggregazione/
  - Composizione
  - Ereditarietà
- **Vincoli**
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram

- Vincoli “implementativi” possono essere espressi come commenti particolari



# OCL



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

➤ **Interfacce**

Ulteriori

caratteristiche

Object diagram

- Un'interfaccia è la dichiarazione di un set di caratteristiche e obblighi pubblici
- Specifica un contratto, che deve essere rispettato da qualunque istanza di classificatore che realizzi l'interfaccia
  - gli obblighi specificati sono vincoli di diverso genere, come pre- e post-condizioni o specifiche di protocolli che impongono un ordine nelle interazioni
- La specifica dell'interfaccia deve essere implementata da un classificatore istanziabile avente caratteristiche pubbliche conformi all'interfaccia



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

➤ Interfacce

Ulteriori

caratteristiche

Object diagram

- Simile a una classe, ma con restrizioni:
  - tutte le operazioni sono pubbliche e astratte
    - non hanno implementazioni di default
  - tutti i dati sono costanti
- Una classe può sempre implementare più interfacce
- Due tipi di interfacce
  - Fornita: la classe che espone l'interfaccia fornisce le operazioni ivi dichiarate
  - Richiesta: la classe che espone l'interfaccia ha bisogno delle operazioni ivi dichiarate per poter svolgere i propri compiti
- Indica anche l'insieme minimo di “servizi” offerti o richiesti



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

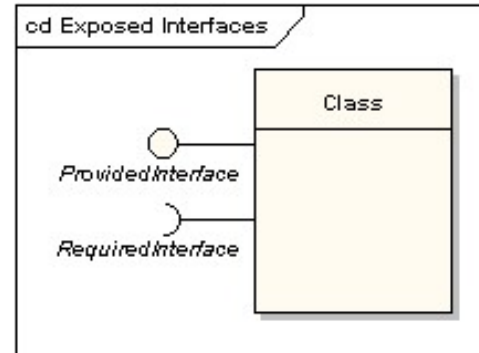
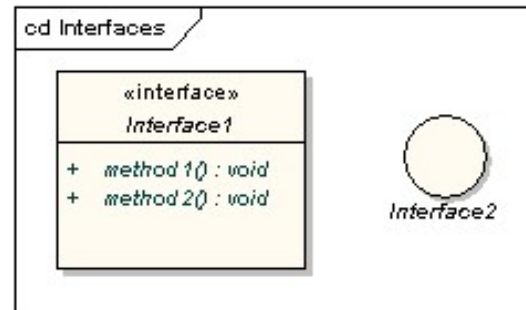
Vincoli

➤ **Interfacce**

Ulteriori

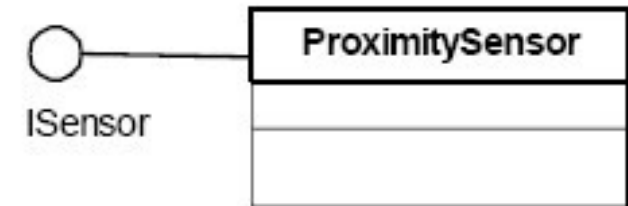
caratteristiche

Object diagram

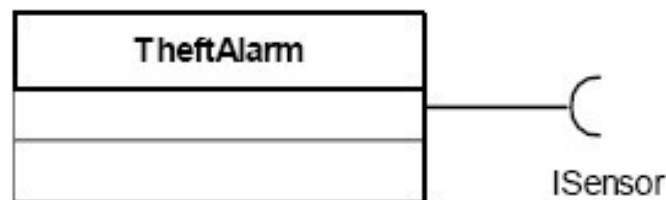




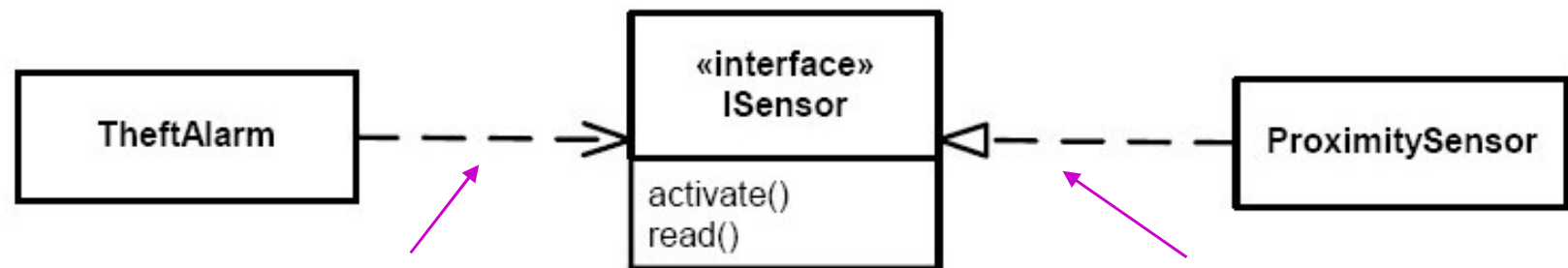
Introduzione  
Classi  
Associazioni  
Aggregazione/  
Composizione  
Ereditarietà  
Vincoli  
➤ **Interfacce**  
Ulteriori  
caratteristiche  
Object diagram



- **ISensor: interfaccia fornita da ProximitySensor**



- **ISensor: interfaccia richiesta da TheftAlarm**



Dipendenza

Realization  
relationship



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

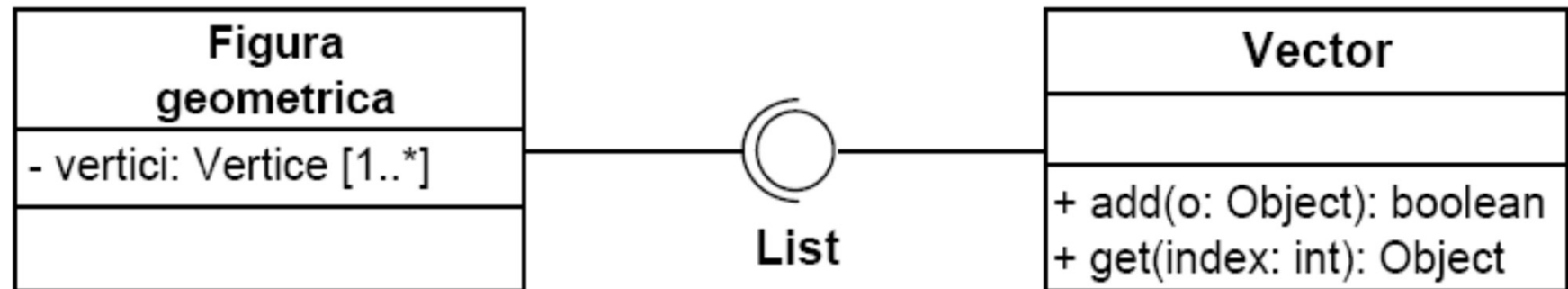
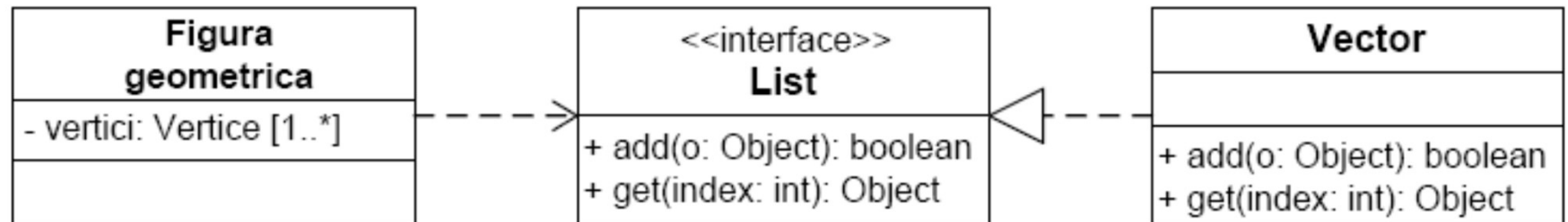
Vincoli

➤ **Interfacce**

Ulteriori

caratteristiche

Object diagram







Introduzione

Classi

Associazioni

Aggregazione/

Composizione

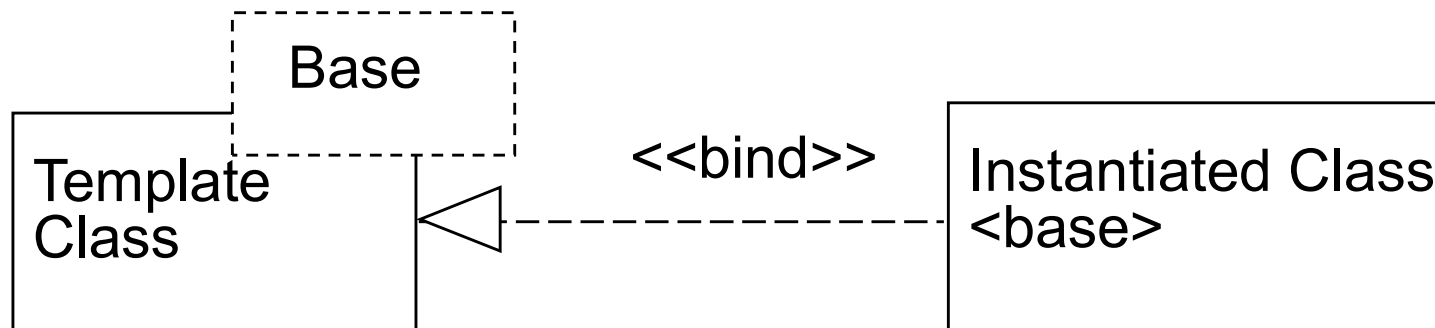
Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram

- Rappresenta una classe contenente uno o più tipi parametrici
- Definisce una famiglia di classi
- Non è una classe: per ottenere la classe occorre prima istanziare il parametro





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

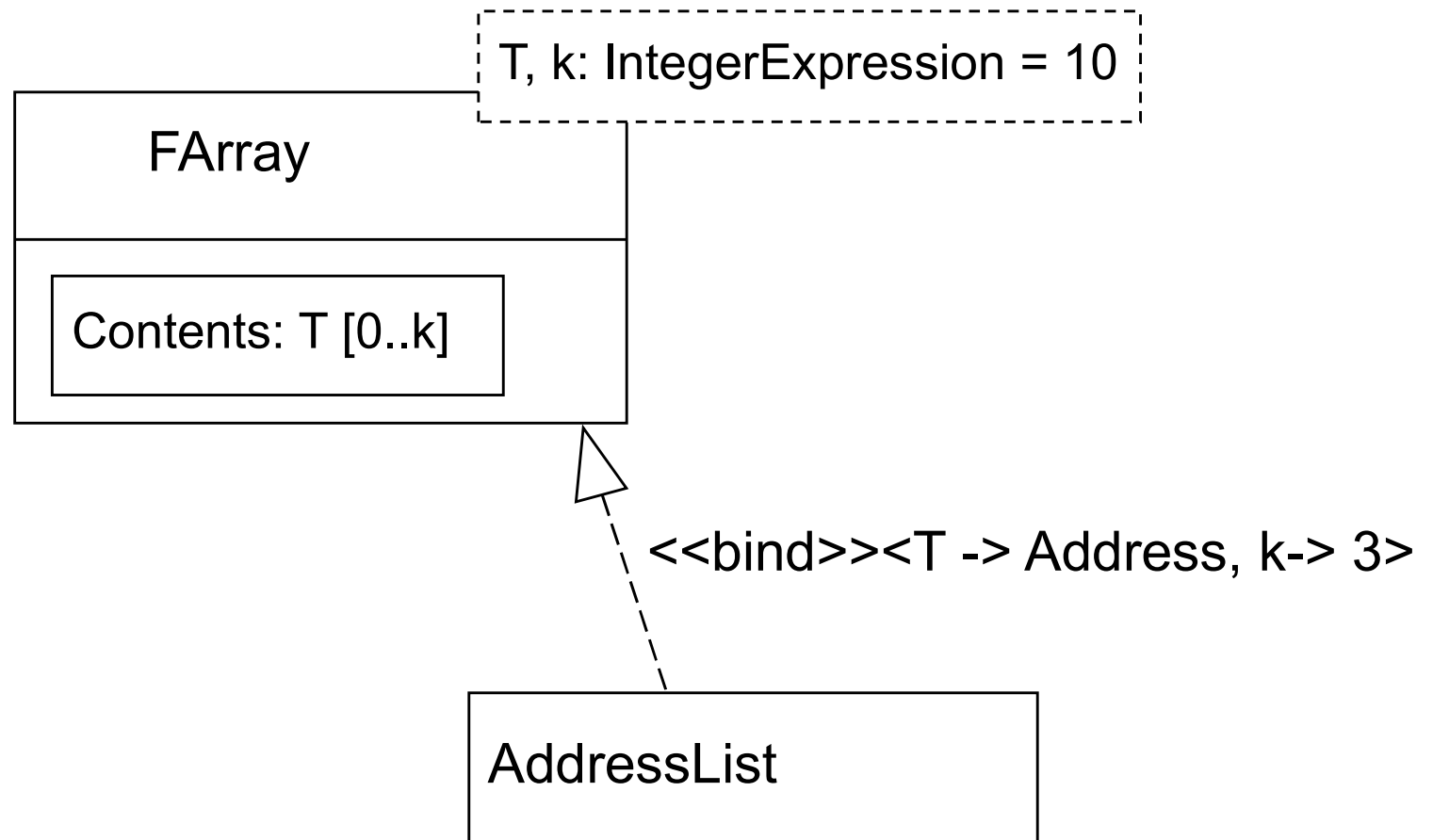
Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**

Object diagram

- La relazione collega una classe parametrica e la classe creata istanziando il parametro





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

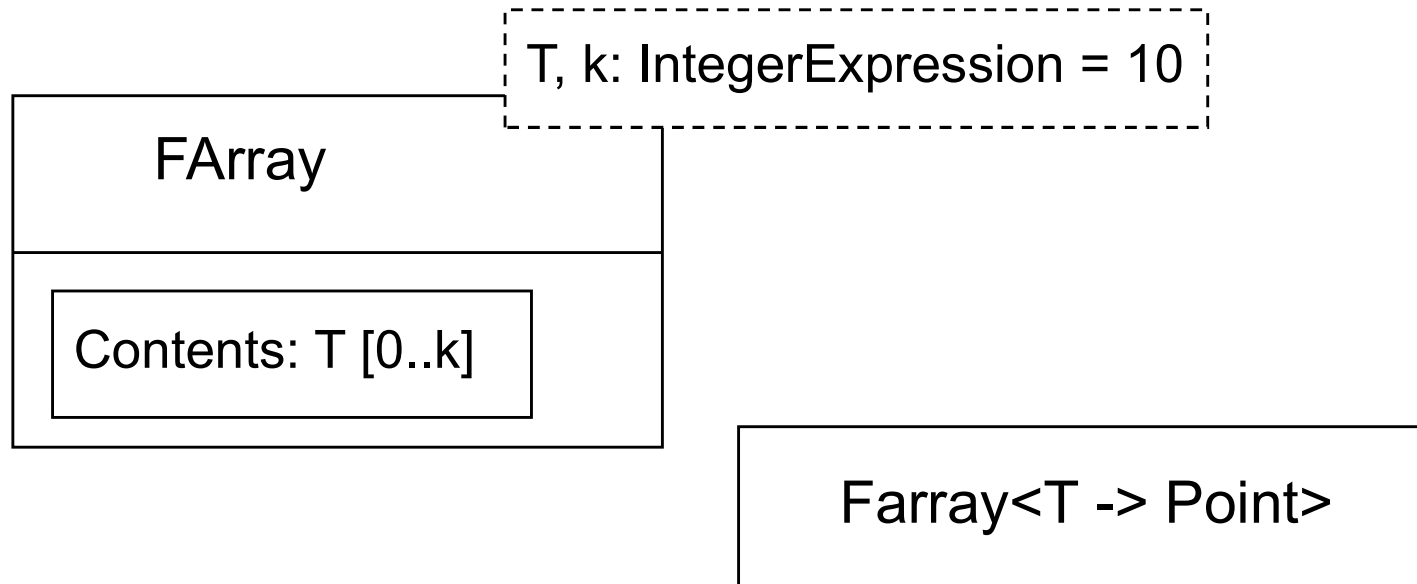
Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**

Object diagram

- La relazione collega una classe parametrica e la classe creata istanziando il parametro



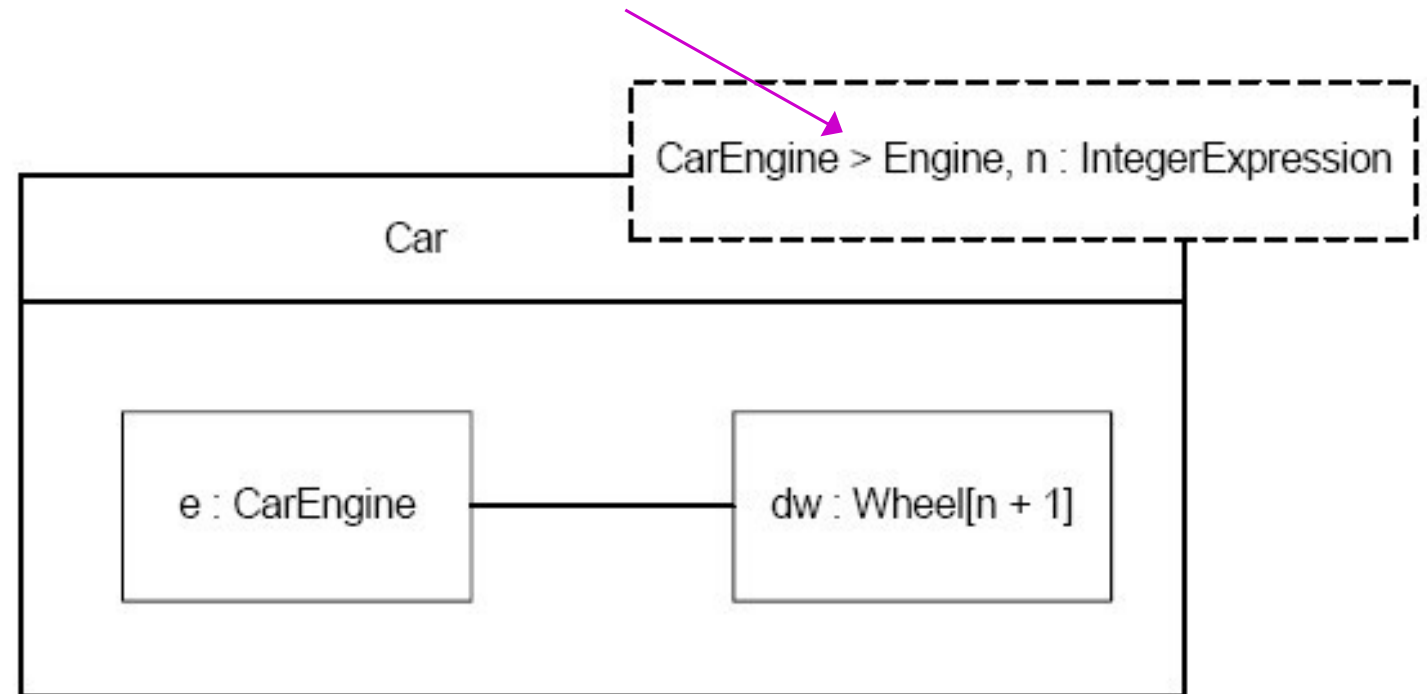
qui la relazione di binding è  
implicita (per k si usa il  
default)



# Classi parametriche: altro esempio

## UML – Class Diagram

La classe usata come parametro CarEngine deve essere compatibile con la classe Engine



DieselCar : Car<CarEngine -> DieselEngine, n -> 2>



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

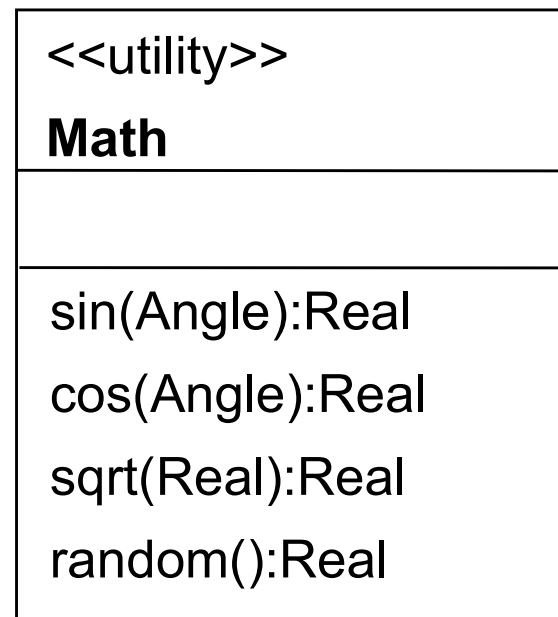
Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram

- Raggruppano variabili e procedure, che di fatto verranno viste come globali





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

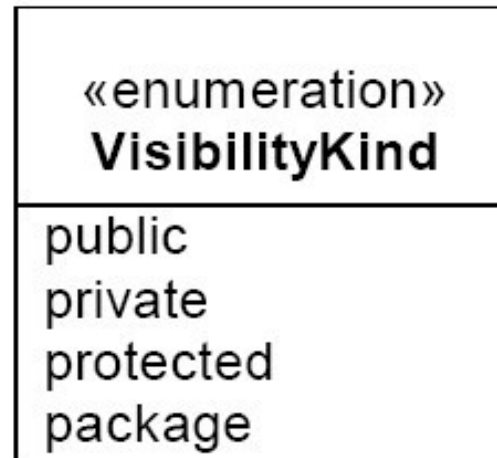
Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**

Object diagram





- Introduzione
- Classi
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- **Ulteriori**
- caratteristiche
- Object diagram

- Rappresentano uno dei meccanismi attraverso i quali è possibile estendere la notazione UML
- Vengono utilizzati per specializzare la semantica di elementi predefiniti in UML ( classi, associazioni, ecc.)
  - uno stereotipo rappresenta una sottoclasse di un elemento esistente (classe, associazione, package, use case ...) ed individua un particolare uso per l'elemento
  - un elemento stereotipato può essere utilizzato in tutte le situazioni in cui può essere usato l'elemento originale
- Uno stereotipo può essere descritto in modo testuale (<<StereotypeName>>) o grafico



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

➤ Ulteriori  
caratteristiche  
Object diagram

- Entity: classe passiva: le sue istanze non sono mai iniziatori di interazioni



`<<entity>>`

**EntityClass**

- Control: classe le cui istanze controllano le interazioni fra collezioni di altri oggetti.



`<<control>>`

**ControlClass**

- Boundary: classe posta alla “periferia” del sistema (comunque all’interno del sistema)

- svolge il ruolo di “Intermediario” tra gli attori e le altre parti del sistema



`<<boundary>>`

**BoundaryClass**





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram

- Uno degli usi degli stereotipi consiste nella definizione di “profili” o “estensioni” di UML, cioè specializzazioni del linguaggio per modellare particolari domini applicativi
- Ad es. esiste una estensione per modellare business process



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

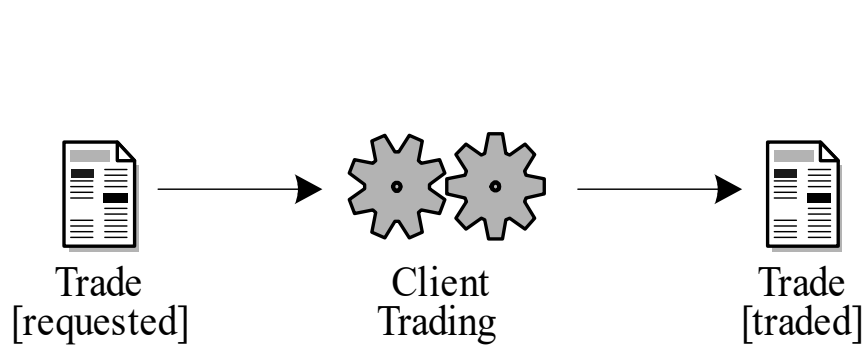
Ereditarietà

Vincoli

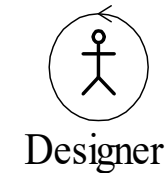
Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram

### ● Icone per rappresentare entità ed azioni



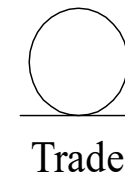
«worker»  
Administrator



«internal worker»  
Designer



«case worker»  
OrderEntry



«entity»  
Trade



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori**  
caratteristiche  
Object diagram

- Relazione che indica una dipendenza di varia natura tra elementi di un modello UML
  - si può avere dipendenza tra classi, packages, use cases, etc.
- Individua una connessione “semantica” tra due elementi, uno dei quali è dipendente dall’altro
  - una modifica nell’elemento indipendente ha effetti su quello dipendente



Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

➤ Ulteriori  
caratteristiche  
Object diagram

- Alcune relazioni di dipendenza (tra classi) esplicitamente citate nella specifica UML:

`<<instantiates>>`

- Un metodo di una classe crea istanze di un'altra classe

`<<calls>>`

- Indica che un metodo di una classe chiama un metodo di un'altra classe

`<<friend>>`

- Indica la possibilità di accesso al contenuto di un'altra classe indipendentemente dalla visibilità prevista dalla classe target

`<<usage>>`

- Indica che un elemento richiede la presenza di un altro per il suo corretto funzionamento (comprende `<<calls>>`, `<<instantiates>>`)



Introduzione

Classi

Associazioni

Aggregazione/

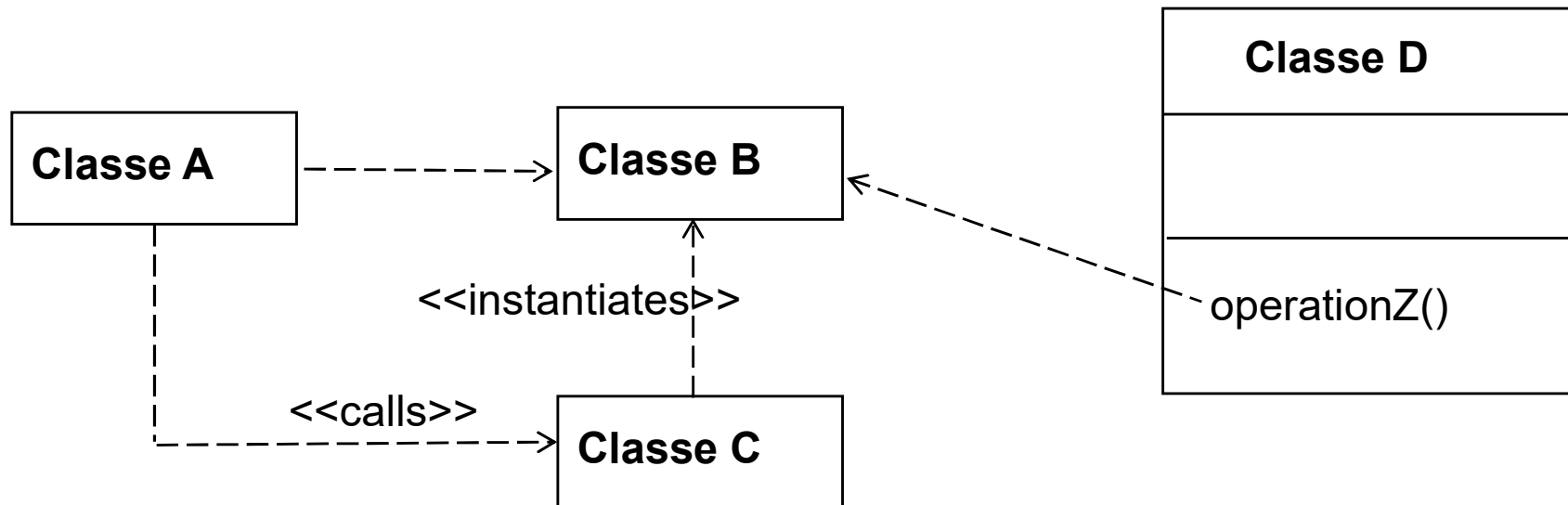
Composizione

Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram





Introduzione

Classi

Associazioni

Aggregazione/

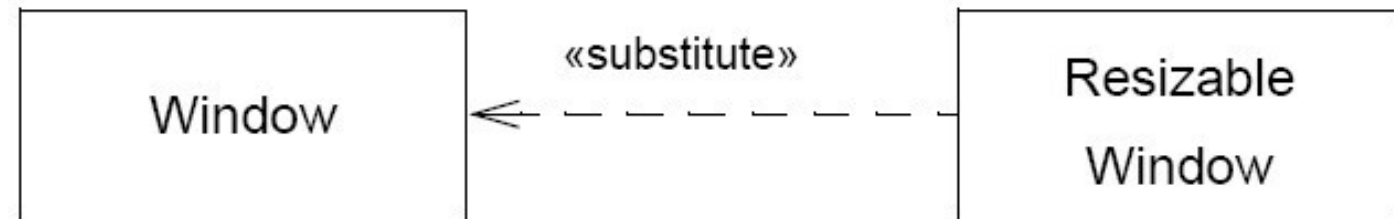
Composizione

Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

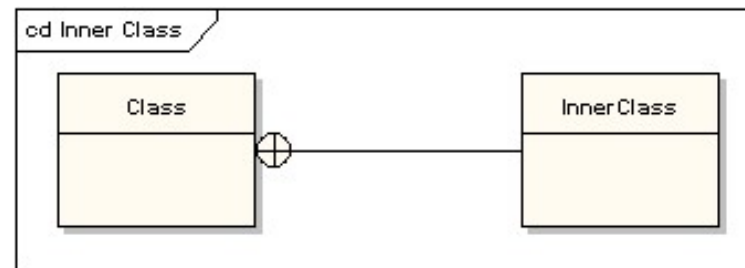
Ereditarietà

Vincoli

Interfacce

➤ **Ulteriori  
caratteristiche**  
Object diagram

- Un nesting mostra l'annidamento tra due elementi





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

Ulteriori

caratteristiche

➤ Object diagram

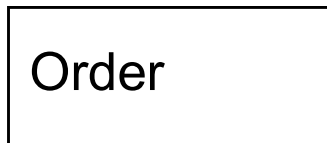
- Descrivono singole istanze di classi (oggetti) e associazioni (links) rappresentate in un particolare class diagram
- Adatti a descrivere esempi o situazioni specifiche (punti di vista o “fotografie” delle istanze esistenti in un certo istante di tempo)
- Si usano durante l’analisi e il progetto, per
  - capire la struttura di oggetti complessi
  - esplicitare la struttura di oggetti complessi
  - presentare “immagini” del sistema



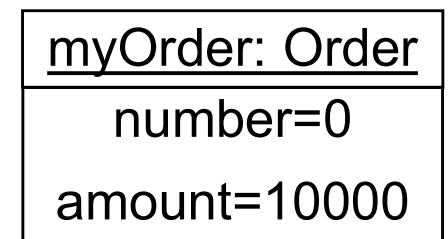
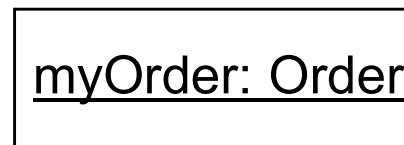
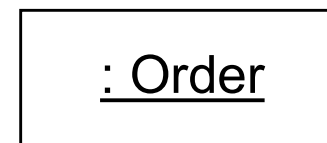


- Scopo delle istanze è solitamente di fornire degli esempi di oggetti, oppure evidenziare oggetti di particolare rilevanza

### Classe

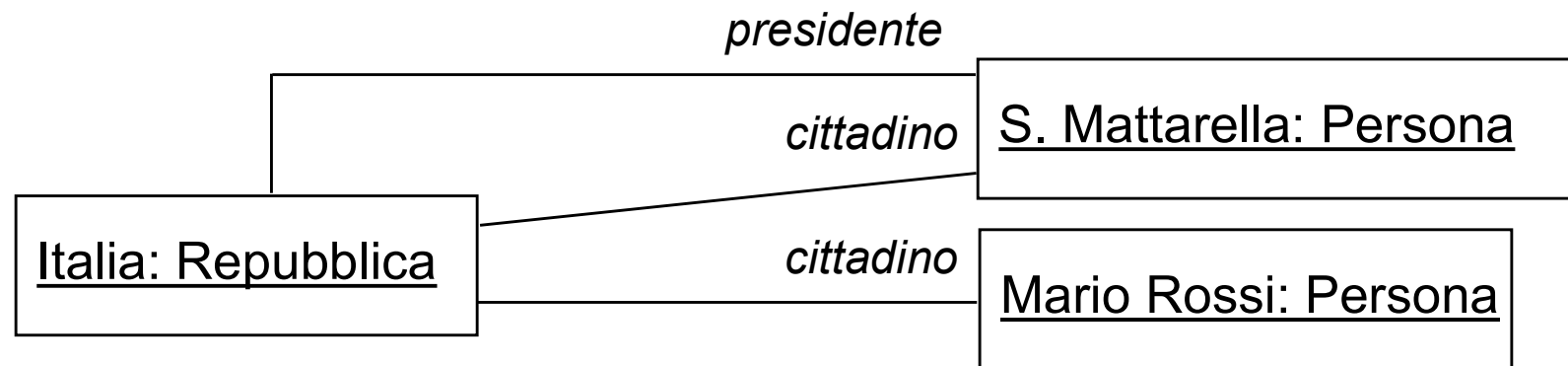


### Istanze



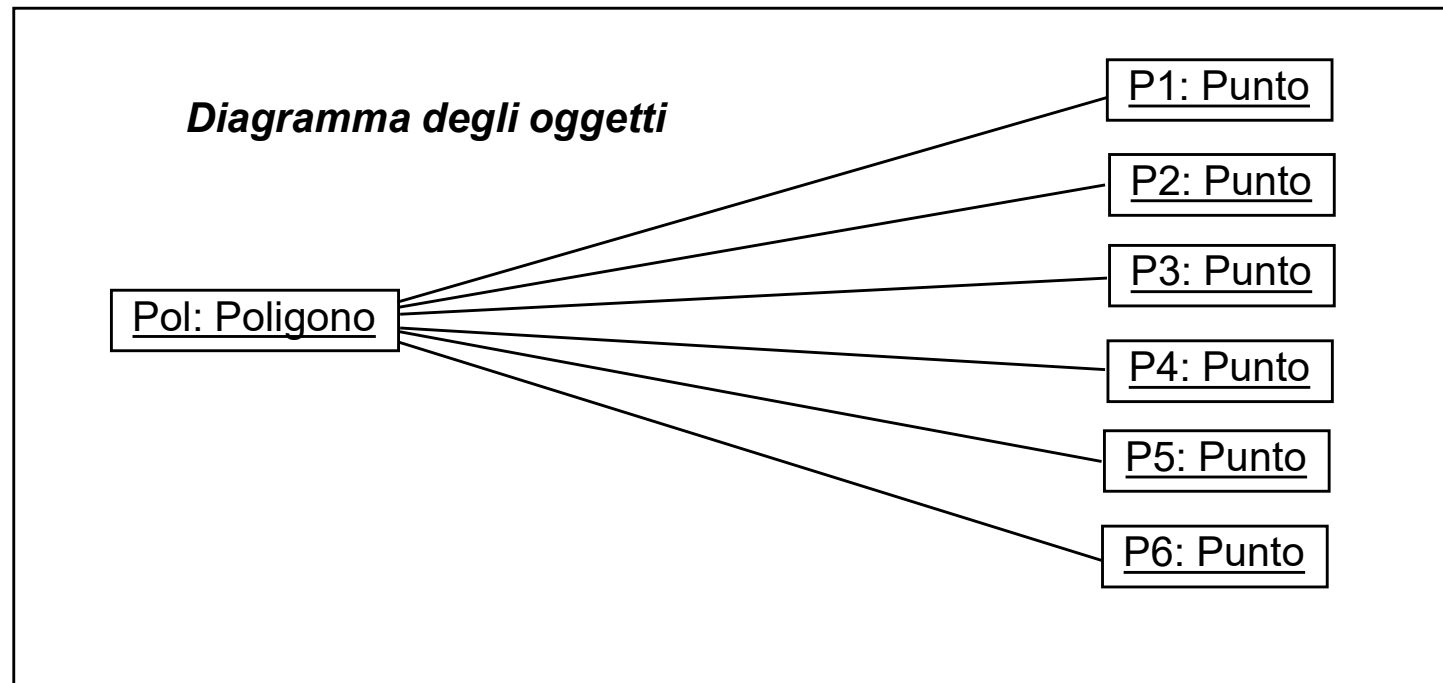
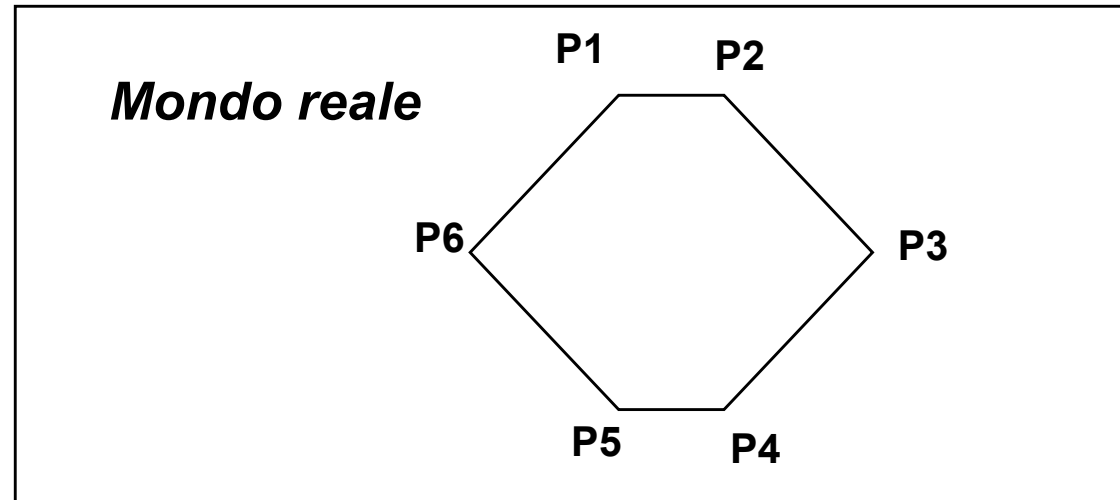


- Un legame (link) è una connessione fisica o concettuale fra due istanze
- Mentre un'associazione connette due classi, un link connette due oggetti
  - un link è un'istanza di un'associazione





- Introduzione
- Classi
- Associazioni
- Aggregazione/
- Composizione
- Ereditarietà
- Vincoli
- Interfacce
- Ulteriori
- caratteristiche
- Object diagram





Introduzione

Classi

Associazioni

Aggregazione/

Composizione

Ereditarietà

Vincoli

Interfacce

Ulteriori

caratteristiche

➤ Object diagram

- Se si fosse scelta una molteplicità 4, l'object diagram non sarebbe corretto

