



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Diagramma delle classi

(Class Diagram)

Ingegneria Informatica -  
Ingegneria del software

Dott.ssa Silvia Bonfanti  
silvia.bonfanti@unibg.it

Università di Bergamo – sede  
Dalmine

26/10/2023

# Materiale

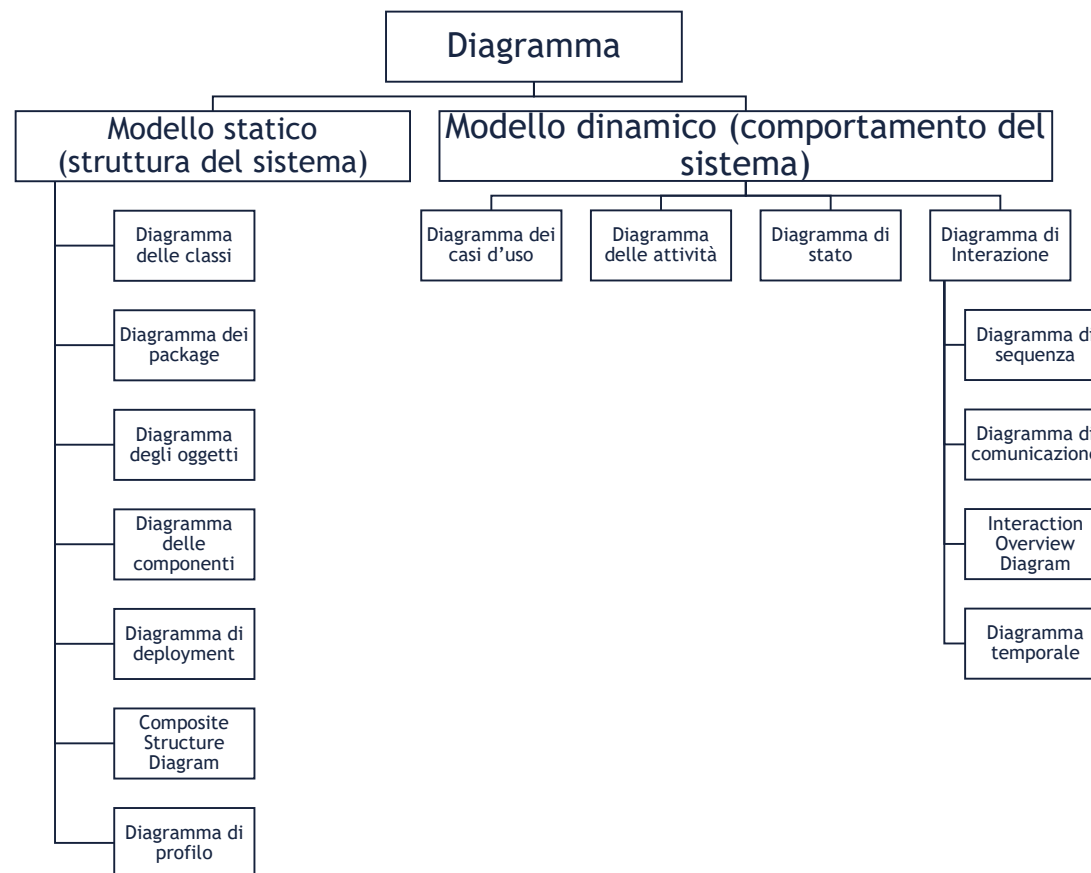
*Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel.  
UML@Classroom: An Introduction to Object-Oriented Modeling,  
Springer Verlag, 2015.*



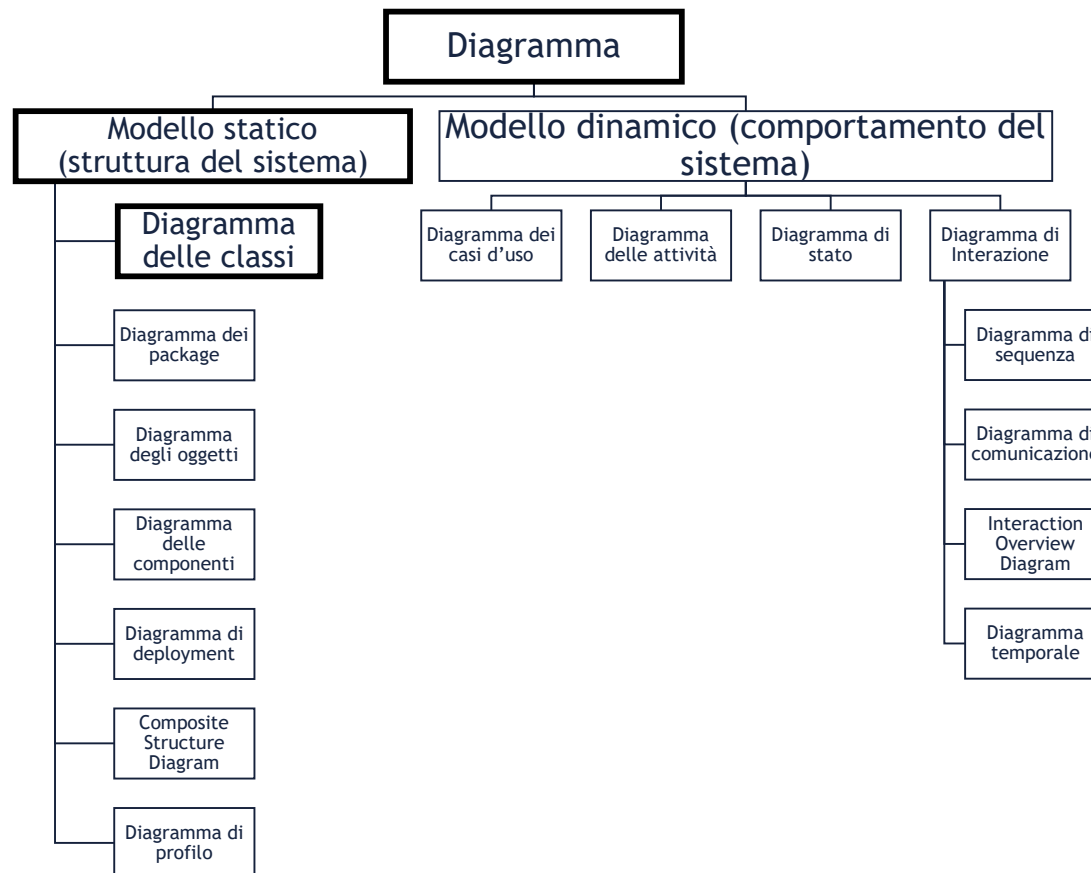
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Introduzione



# Introduzione



# Contenuto

- ☐ Oggetti
- ☐ Classi
- ☐ Attributi
- ☐ Operazioni
- ☐ Relazioni
  - ☐ Associazione binaria
  - ☐ Associazione N - ary
  - ☐ Classe Associativa
  - ☐ Aggregazione
  - ☐ Generalizzazione
- ☐ Creazione di un diagramma di classe
- ☐ Generazione di codice



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

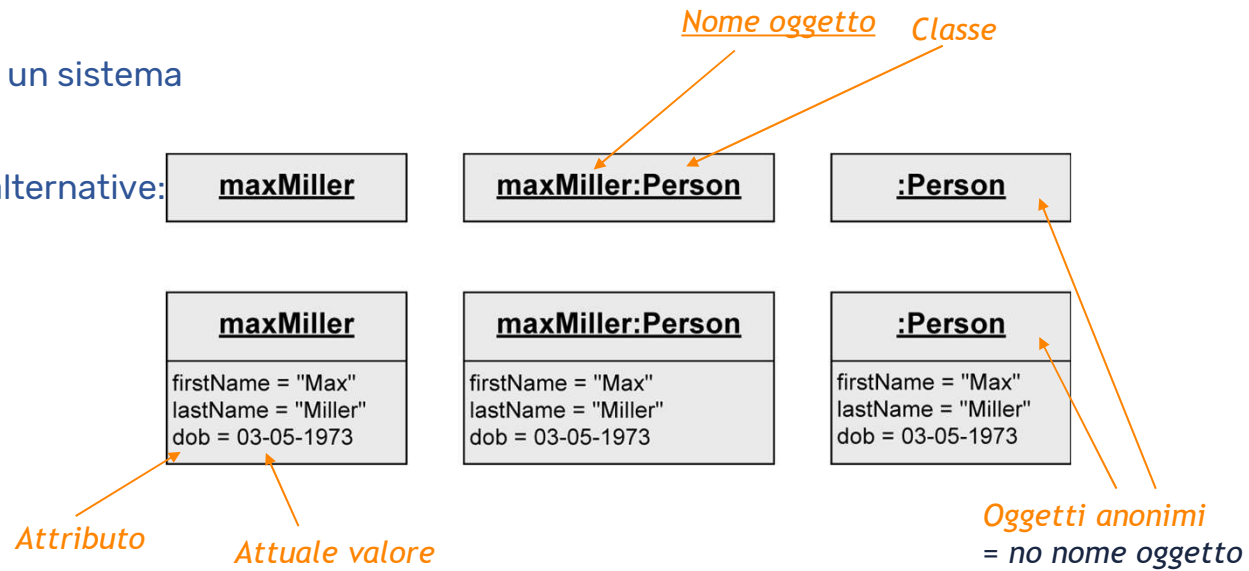
Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Oggetto

o:C

Individui di un sistema

Notazioni alternative:



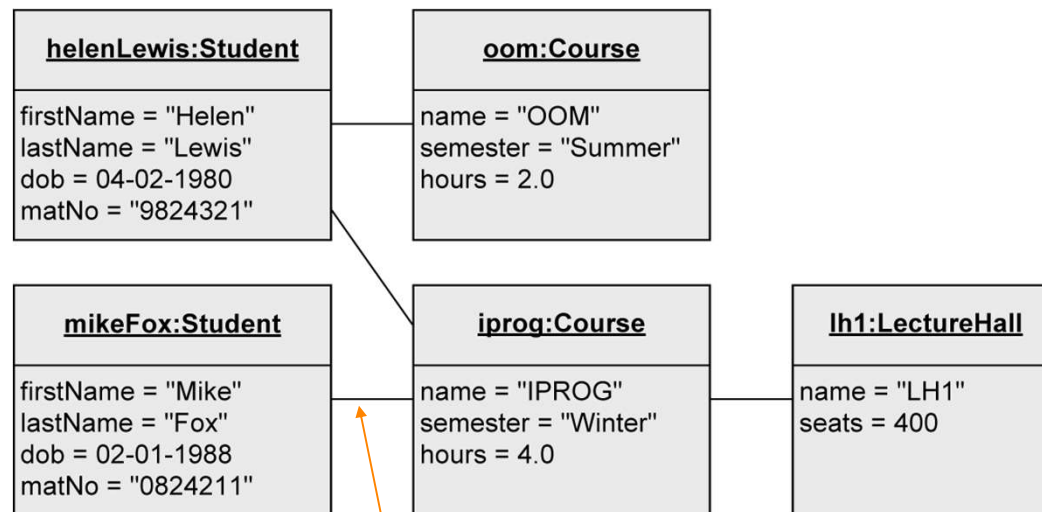
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Diagramma di oggetti

Oggetti di un sistema e loro relazioni (link)

Istantanea di oggetti in un momento specifico



*Legame generico*



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Dall'oggetto alla classe

Gli individui di un sistema hanno spesso caratteristiche e comportamenti identici  
Una classe è un piano di costruzione per un insieme di oggetti simili di un sistema

Gli oggetti sono istanze di classi

**Attributi:** caratteristiche strutturali di una classe

- Valore diverso per ogni istanza (= oggetto)

**Operazioni:** comportamento di una classe

- Identico per tutti gli oggetti di una classe  
→ non rappresentati nel diagramma degli oggetti

*Class*

Person
firstName: String lastName: String dob: Date

*Oggetto di quella classe*

<u>maxMiller:Person</u>
firstName = "Max" lastName = "Miller" dob = 03-05-1973



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione



# Classe

A

*Nome*

**Course**

*Attributi*

name: String  
semester: SemesterType  
hours: float

*Operazioni*

getCredits(): int  
getLecturer(): Lecturer  
getGPA(): float



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

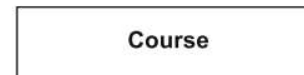
Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Livelli di dettaglio di una classe

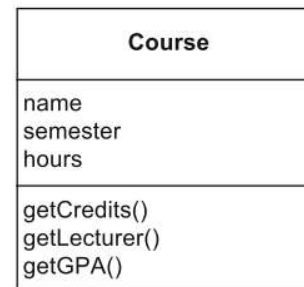
A: solo il nome

B: attributi e operazioni

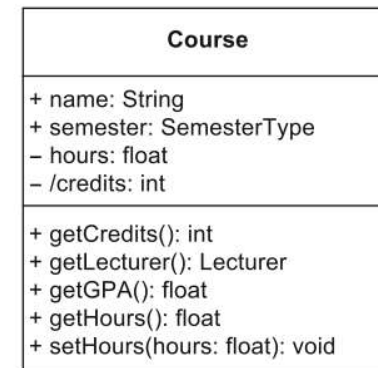
C: con i loro tipi e altre informazioni



(a)



(b)



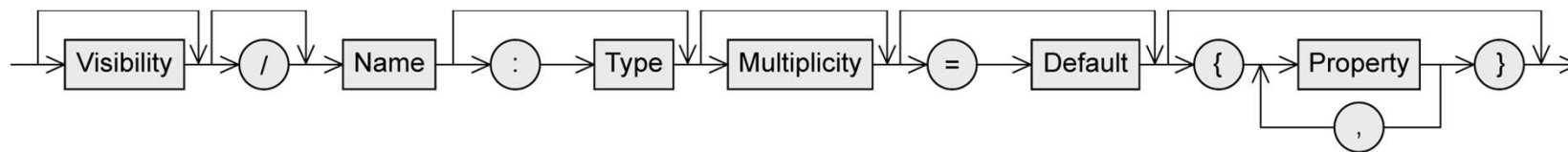
(c)



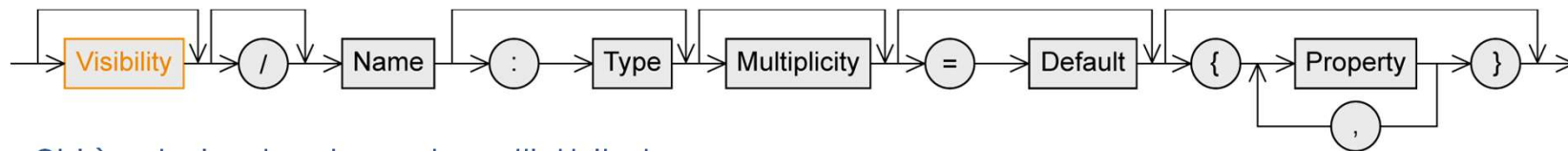
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Sintassi degli attributi



# Sintassi degli attributi - Visibilità



Chi è autorizzato ad accedere all'attributo

- + ... pubblico: tutti
- - ... privato: solo l'oggetto stesso
- # ... protetto: classe stessa e sottoclassi
- ~ ... pacchetto: classi che si trovano nello stesso pacchetto

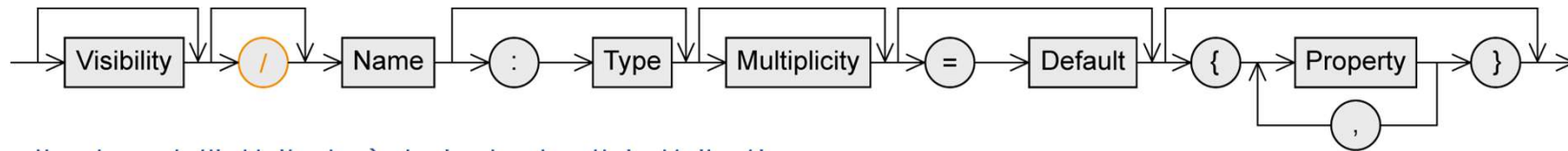
Person
+ firstName: String + lastName: String - dob: Date # address: String[1..*] {unique, ordered} - ssNo: String {readOnly} - /age: int - password: String = "pw123" - <u>personsNumber: int</u>



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Sintassi degli attributi - Attributo derivato

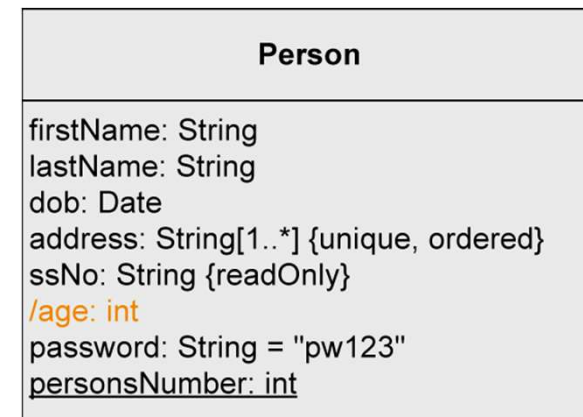


Il valore dell'attributo è derivato da altri attributi

Non fa parte dello stato

Esempio:

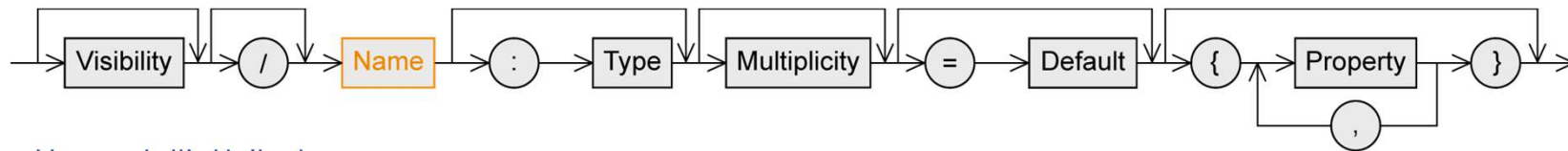
- età : calcolata dalla data di nascita



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Sintassi degli attributi - Nome

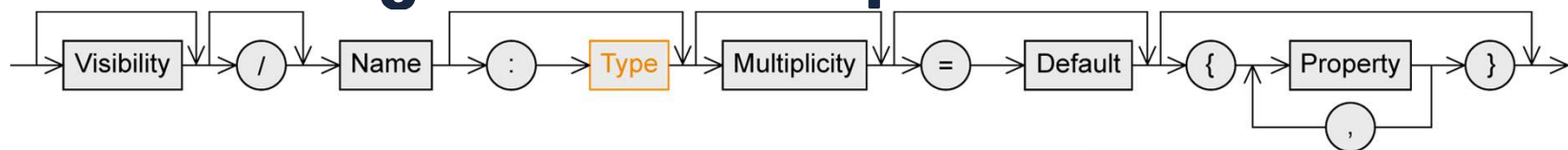


Nome dell'attributo

Person
<code>firstName: String</code> <code>lastName: String</code> <code>dob: Date</code> <code>address: String[1..*] {unique, ordered}</code> <code>ssNo: String {readOnly}</code> <code>/age: int</code> <code>password: String = "pw123"</code> <code><u>personsNumber: int</u></code>

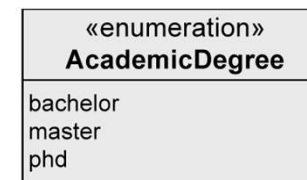
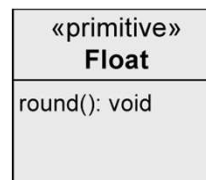
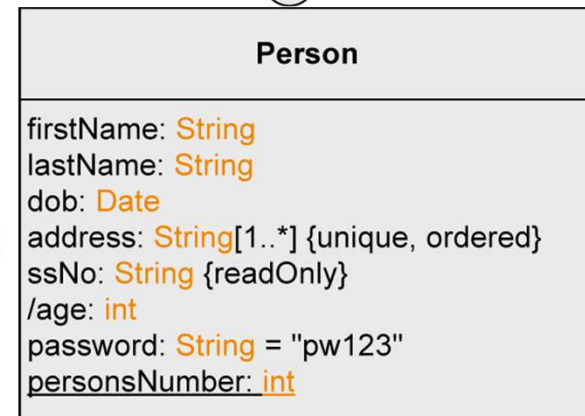


# Sintassi degli attributi - Tipo



## Tipo

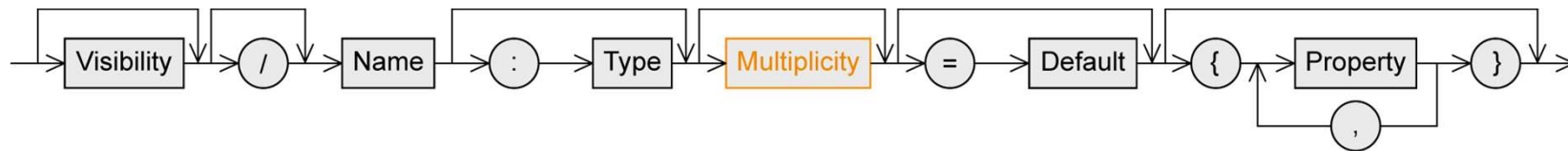
- Classi definite dall'utente
- Tipo di dati
  - Tipo di dati primitivo
    - Predefinito: Boolean, Integer, UnlimitedNatural , String
    - Definito dall'utente: « **primitivo** »
    - Tipo di dati composito: « **tipo di dati** »
  - Enumerazioni: « **enumerazione** »



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Sintassi degli attributi - Molteplicità

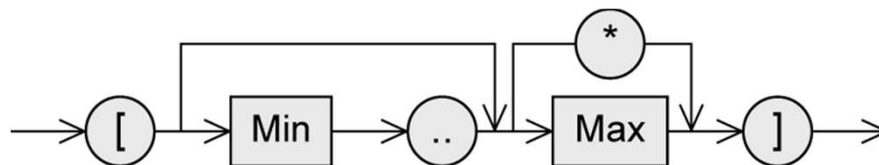


Numero di valori che un attributo può contenere

Valore predefinito: 1

Notazione: [ min..max ]

- nessun limite superiore: [\*] o [0..\*]



Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>

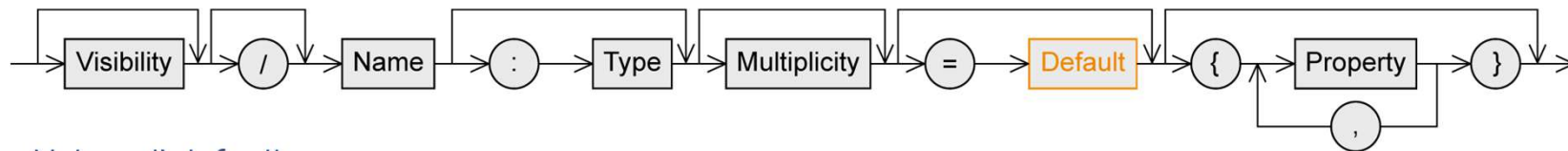


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione



# Sintassi degli attributi – Valore predefinito



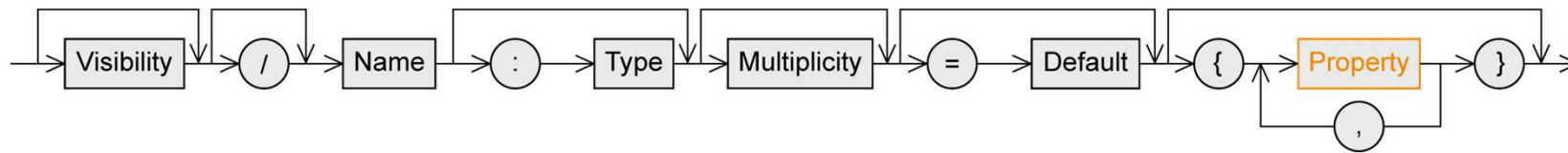
## Valore di default

- Utilizzato se il valore dell'attributo non è impostato in modo esplicito dall'utente

Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>



# Sintassi degli attributi – Proprietà



## Proprietà predefinite

- {readOnly} ... il valore non può essere modificato
- {unique} ... non sono consentiti duplicati
- {non-unique} ... duplicati consentiti
- {ordered} ... ordine fisso dei valori
- {unordered} ... nessun ordine fisso dei valori

## Specifiche degli attributi

- **Set:** {unordered, unique}
- **Multi-set:** {unordered, non-unique}
- **Set ordinato:** {ordered, unique}
- **Elenco:** {ordered, non-unique}

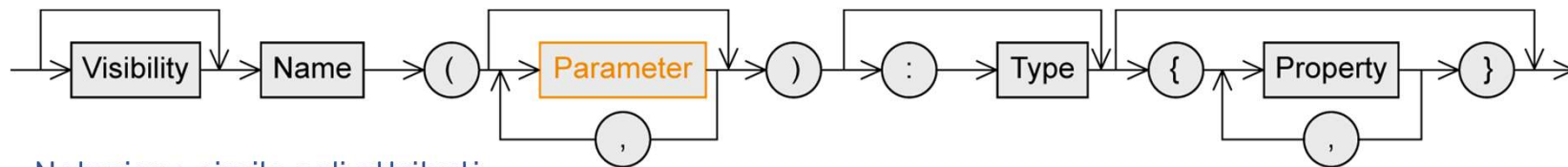
Person
firstName: String lastName: String dob: Date address: String[1..*] {unique, ordered} ssNo: String {readOnly} /age: int password: String = "pw123" <u>personsNumber: int</u>



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Sintassi dell'operazione - Parametri

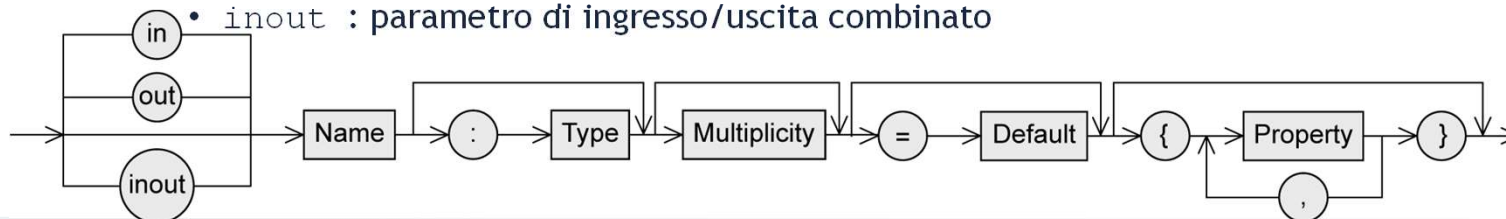


Notazione simile agli attributi

Direzione del parametro

- `in ...` parametro di input
  - Quando viene utilizzata l'operazione, da questo parametro è previsto un valore
- `out ...` parametro di uscita
  - Dopo l'esecuzione dell'operazione, il parametro ha assunto un nuovo valore
- `inout` : parametro di ingresso/uscita combinato

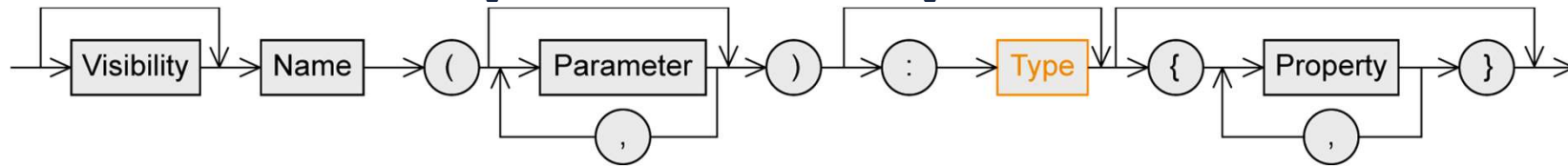
Person
...
+ getName(out fn: String, out ln: String): void
+ updateLastName(newName: String): boolean
+ getPersonsNumber(): int



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Sintassi dell'operazione - Tipo



Tipo di valore restituito

Person
...
getName(out fn: String, out ln: String): void updateLastName(newName: String): boolean getPersonsNumber(): int



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Variabile di classe e operazione di classe

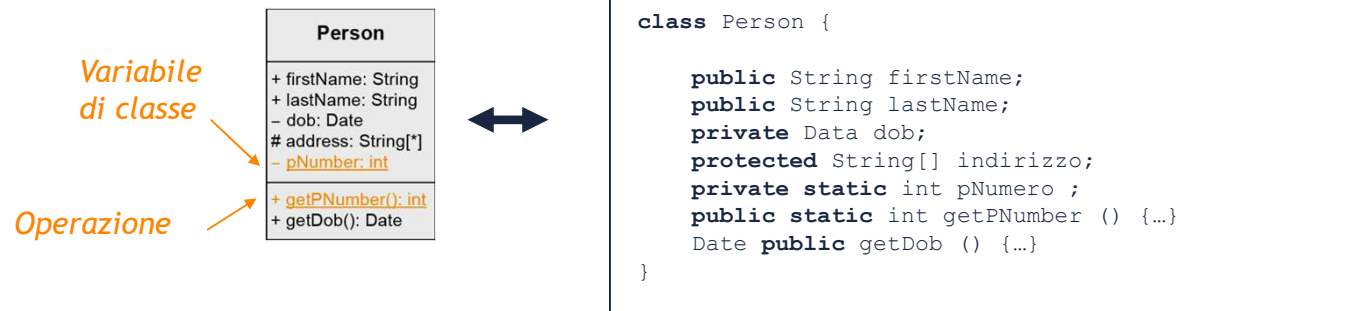
Variabile di classe (= attributo di classe, attributo statico)

- Definito solo una volta per classe, cioè condiviso da tutte le istanze della classe
- Ad esempio contatori per il numero di istanze di una classe, costanti, ecc.

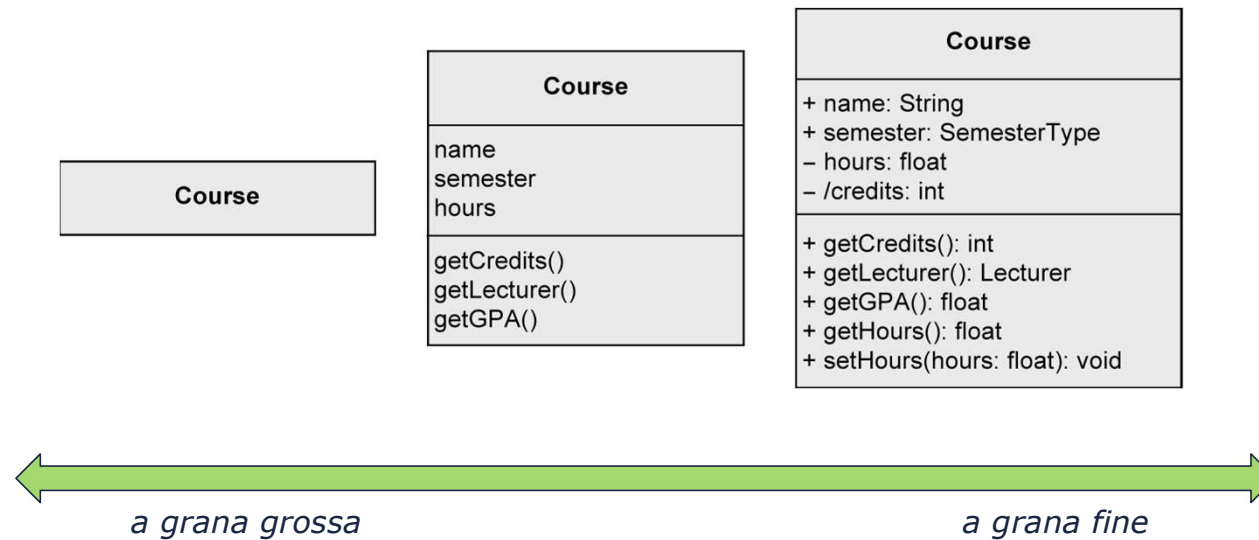
Operazione di classe (= operazione statica)

- Può essere utilizzato se non è stata creata alcuna istanza della classe corrispondente
- Es. costruttori, operazioni di conteggio, matematica. funzioni (sin(x)), ecc.

Notazione: sottolineatura del nome della variabile di classe / operazione di classe

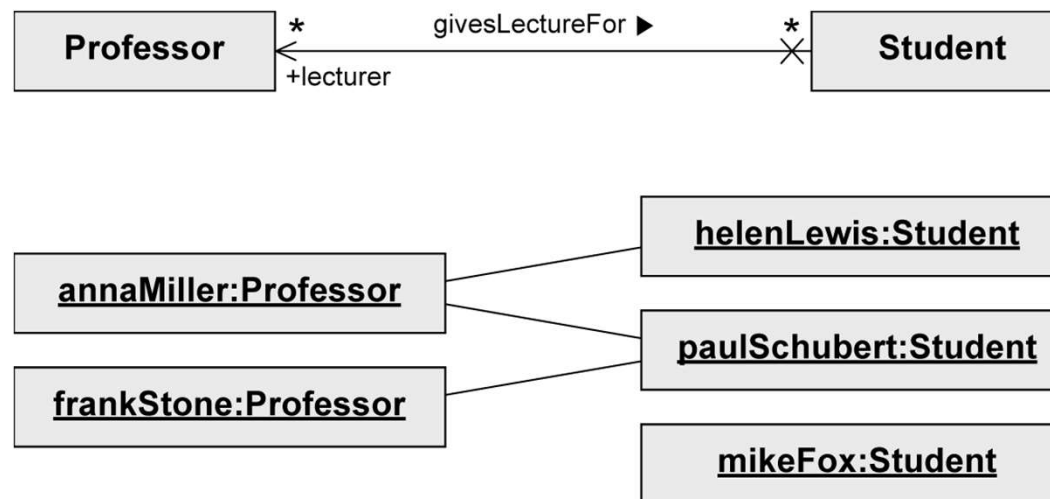


## Specificazione delle classi: diversi livelli di dettaglio



# Associazioni

Modella le possibili relazioni tra istanze di classi

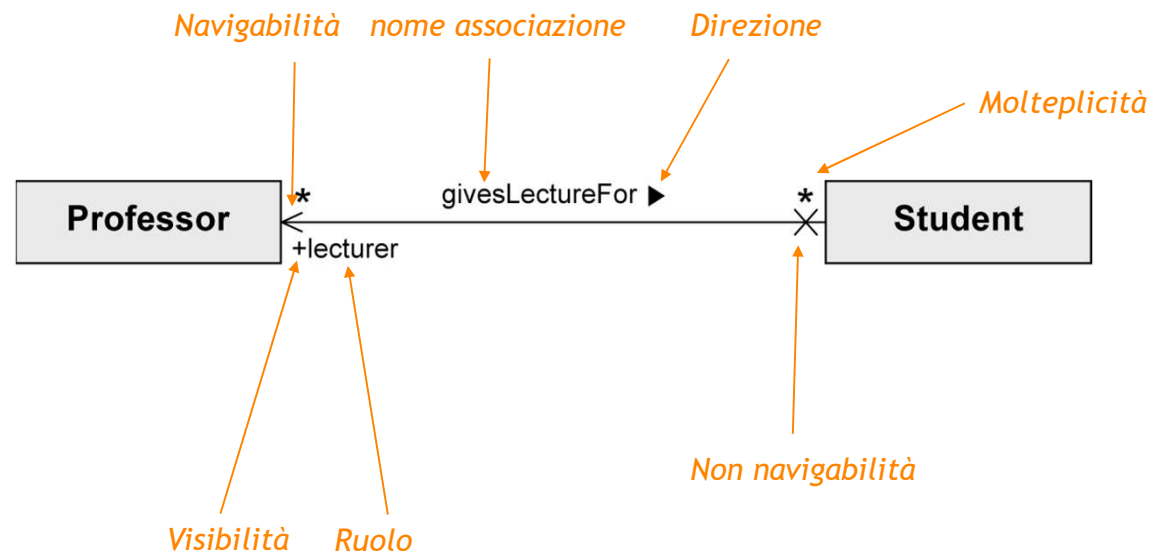
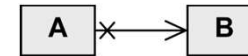


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Associazione Binaria

Collega istanze di due classi tra loro



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione



# Associazione Binaria - Navigabilità

**Navigabilità:** un oggetto conosce i suoi oggetti partner e può quindi accedere ai loro attributi e operazioni visibili

- Indicato dalla punta della freccia aperta

**Non navigabilità**

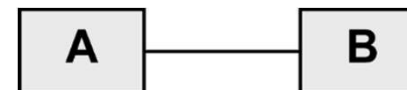
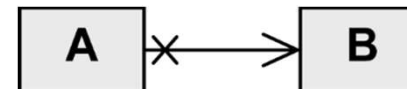
- Indicato da croce

**Esempio:**

- **A** può accedere agli attributi visibili e alle operazioni di **B**
- **B** non può accedere ad alcun attributo e operazione di **A**

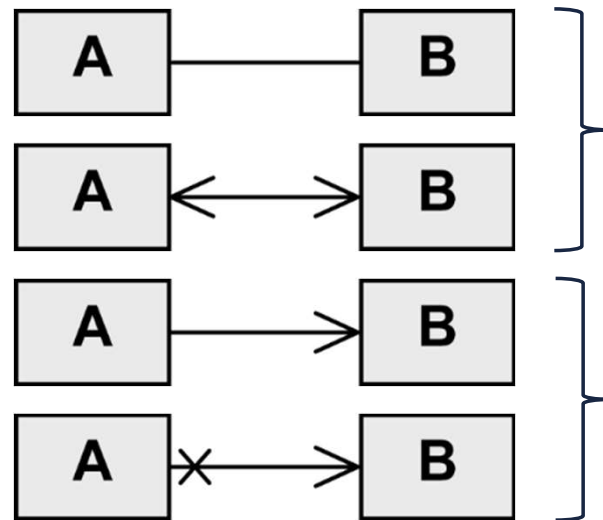
**Navigabilità indefinita**

- Si presume la navigabilità bidirezionale

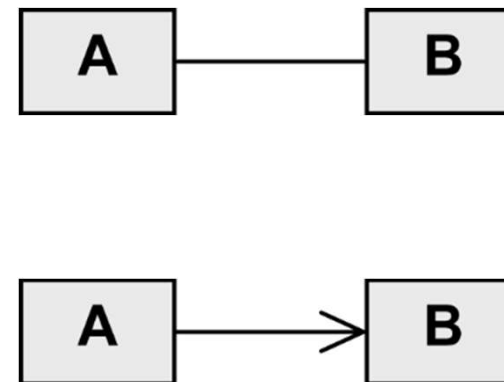


# Navigabilità: standard UML e best practice

*Standard UML*

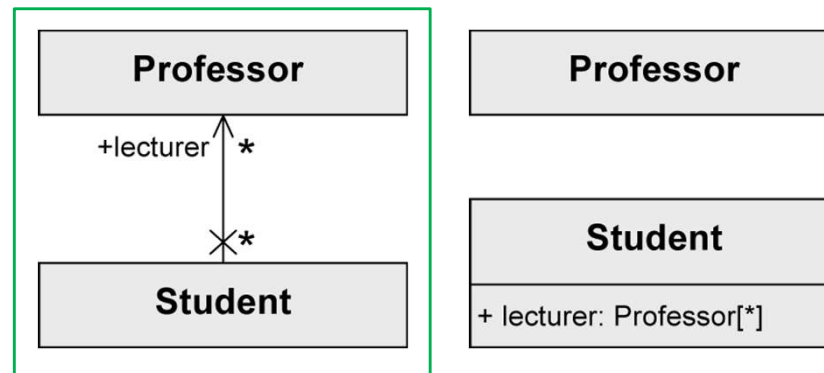


*Best practice*



# Associazione binaria come attributo

*Preferibile*



Notazione simile a Java:

```
class Professore{...}  
  
class Studente{  
    public Professore[] lecturer;  
    ...  
}
```

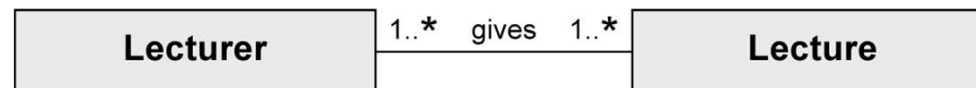
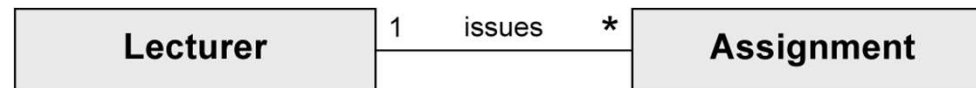


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

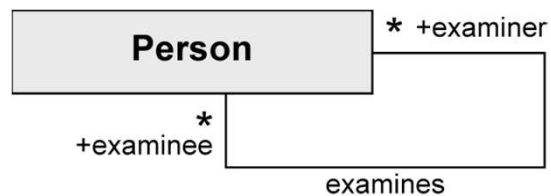
Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Associazione binaria – Molteplicità e ruolo

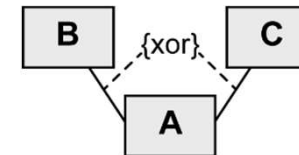
**Molteplicità:** numero di oggetti che possono essere associati esattamente a un oggetto del lato opposto



**Ruolo:** descrive il modo in cui un oggetto è coinvolto in una relazione di associazione

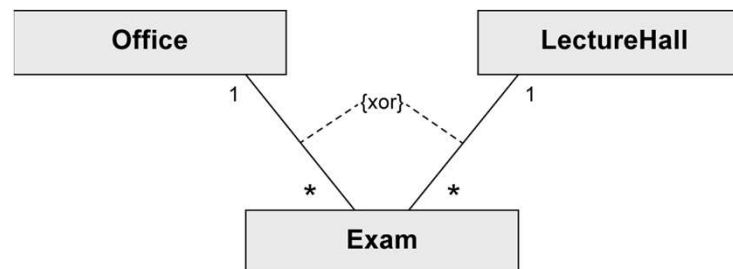


# Associazione binaria – vincolo



vincolo “o esclusivo” (xor).

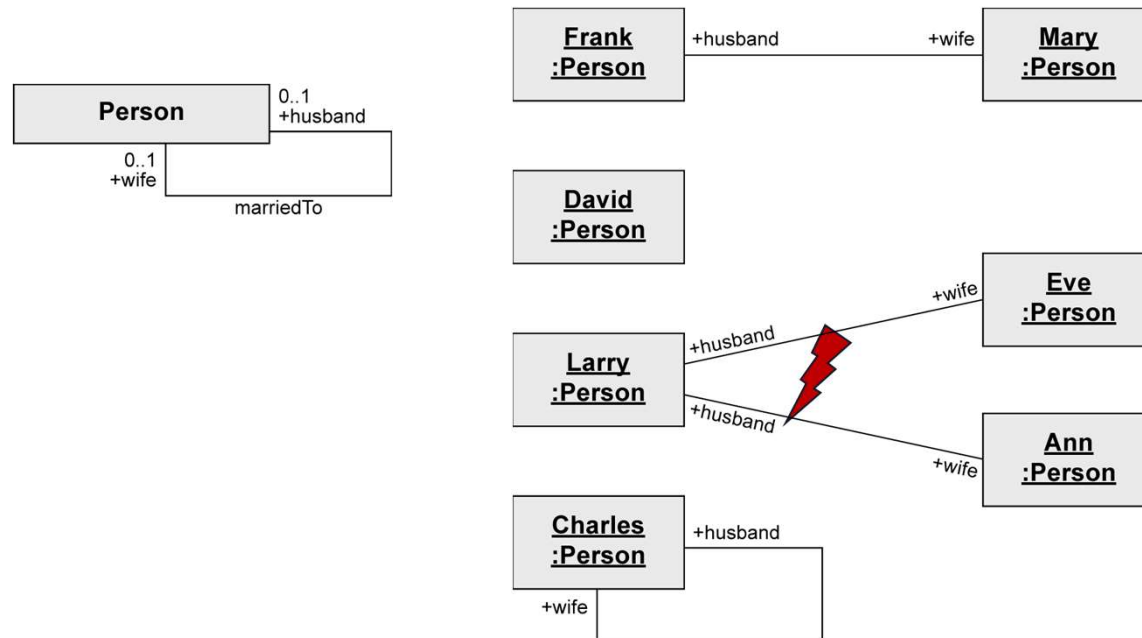
Un oggetto di classe **A** deve essere associato ad un oggetto di classe **B** o ad un oggetto di classe **C** ma non ad entrambi.



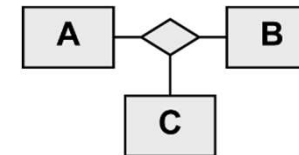
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

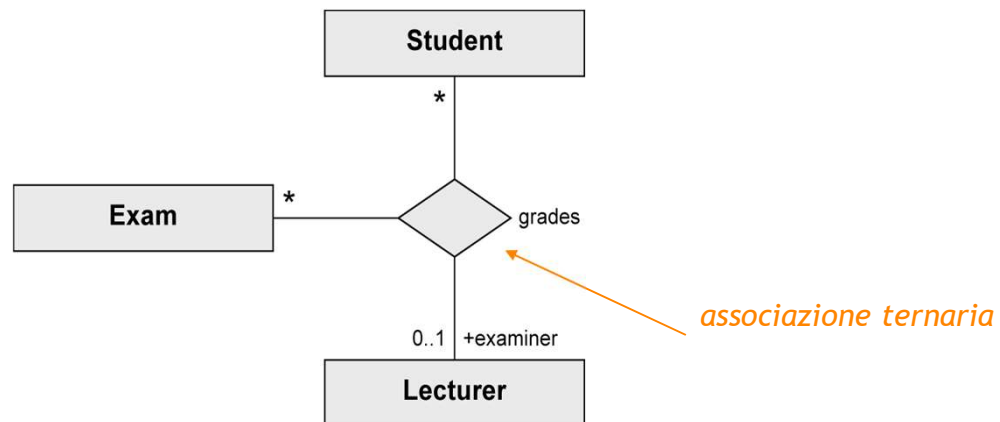
# Associazione unaria - Esempio



## n- Associazione (1/2)



Nella relazione sono coinvolti più di due oggetti partner.  
Nessuna direzione di navigazione



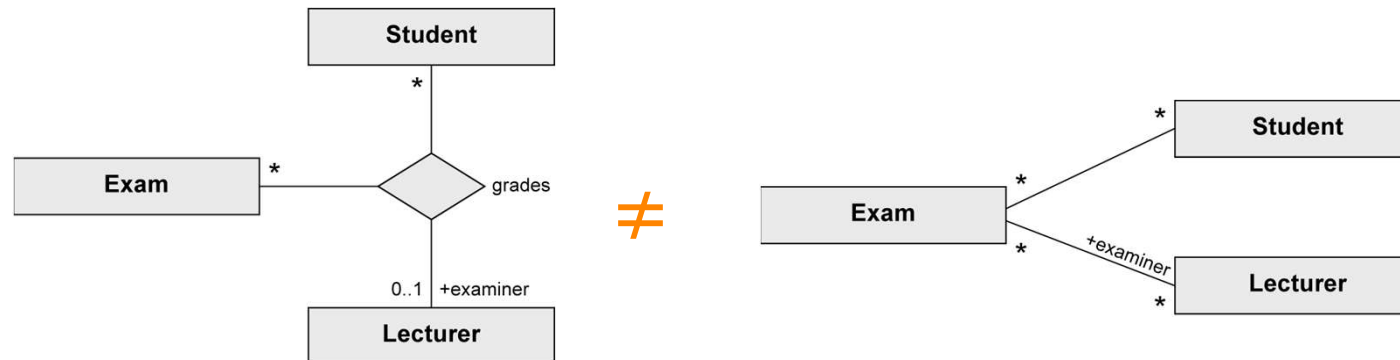
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

## n- Associazione (2/2)

Esempio

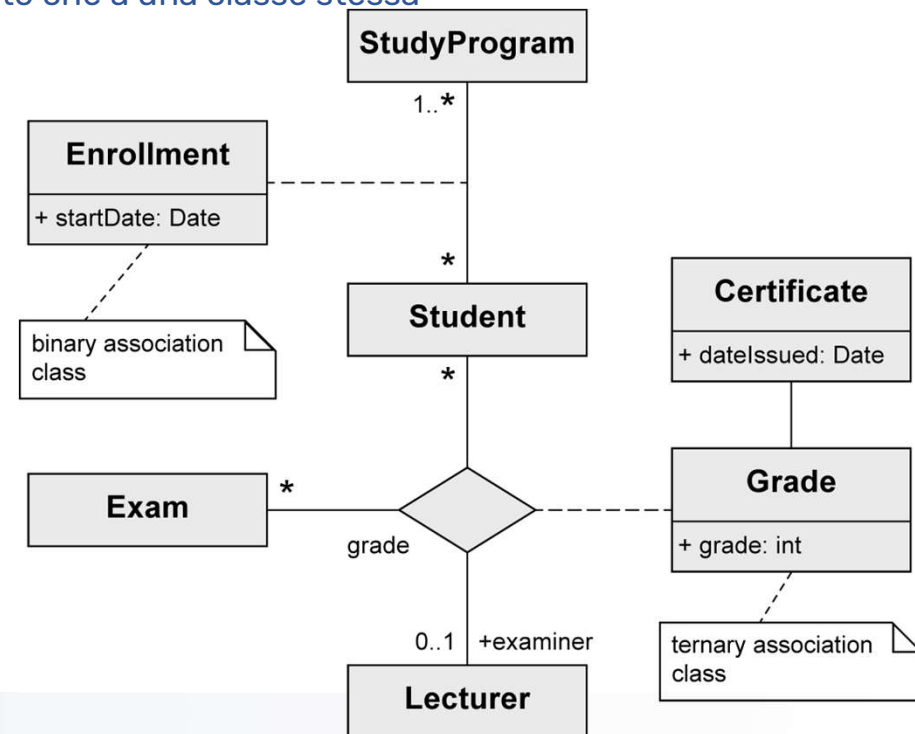
- ( **Studente** , **Esame** )  $\rightarrow$  ( **Docente** )
  - Uno studente sostiene un esame con uno o nessun docente
- ( **Esame** , **Docente** )  $\rightarrow$  ( **Studente** )
  - Un esame con un docente può essere sostenuto da un numero qualsiasi di studenti
- ( **Studente** , **Docente** )  $\rightarrow$  ( **Esame** )
  - Uno studente può essere valutato da un **docente** per qualsiasi numero di esami





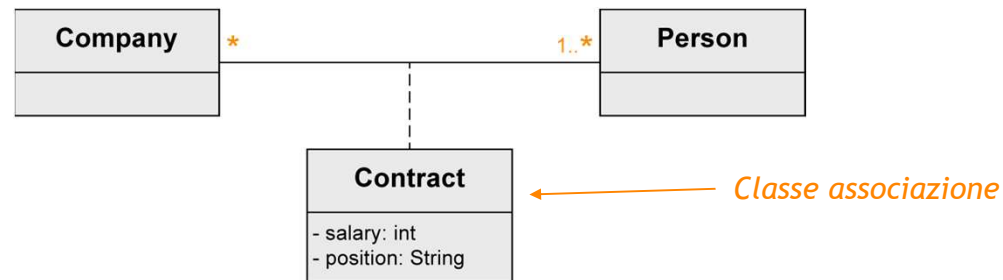
# Classe Associativa (Association classes)

Assegna attributi alla relazione tra classi piuttosto che a una classe stessa

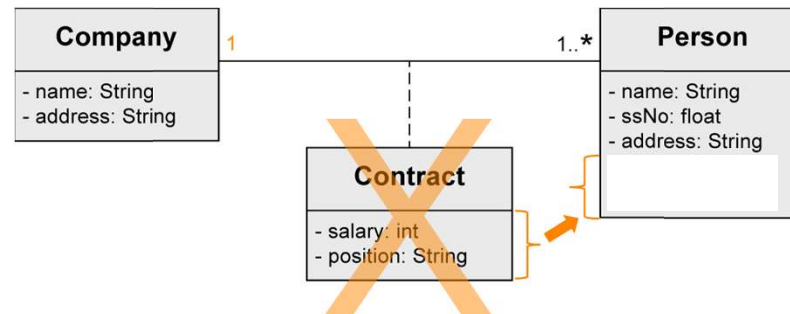


# Classe Associativa

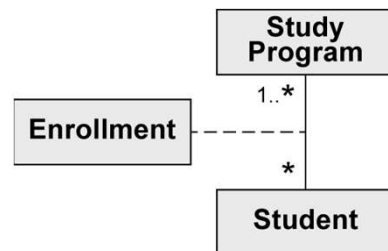
Necessario durante la modellazione di n:m Associazioni



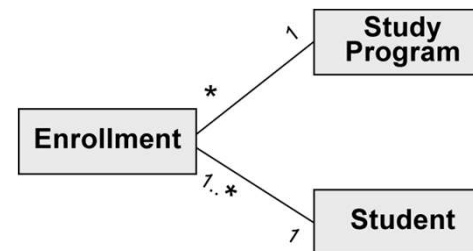
Con 1:1 o 1:n possibile ma non necessario



## Classe associativa vs. Classe normale



≠



Uno Student può iscriversi per uno particolare StudyProgram solo una volta

Uno Student può avere multiple Iscrizione per uno stesso StudyProgram



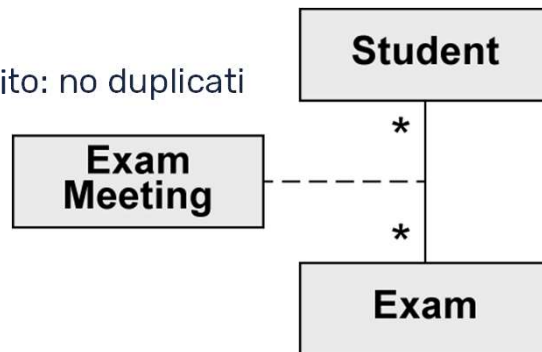
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

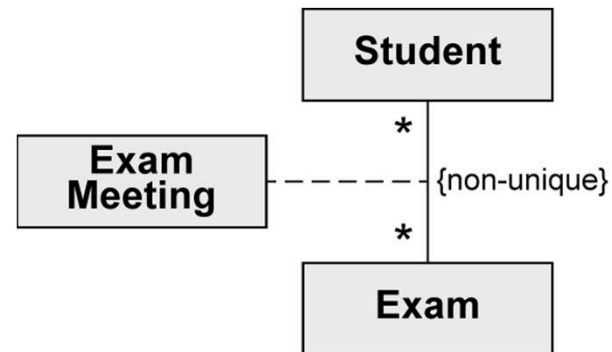
## Classe di associazione – unica / non unica (1/2)

- non univoco : duplicati permessi

Predefinito: no duplicati



A uno *Student* può essere concesso solo una riunione d'esame per un esame specifico **una volta**.



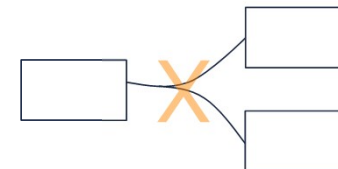
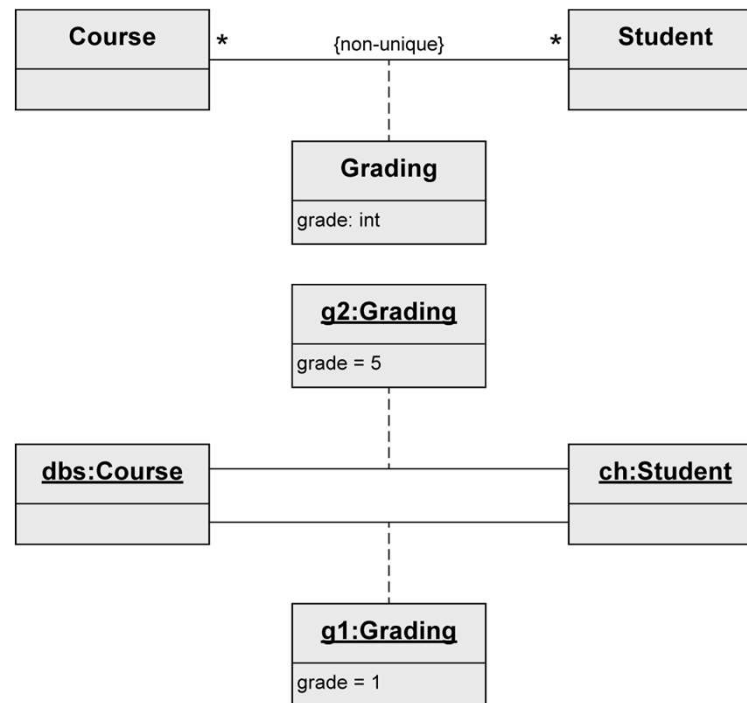
Uno *Student* può averne **più di uno** incontri d'esame per un esame specifico.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Classe di associazione – unica / non unica (2/2)



# Aggregazione

Forma speciale di associazione

Usato per esprimere che una classe fa parte di un'altra classe

Proprietà dell'associazione di aggregazione:

- Transitivo : se  $B$  fa parte di  $A$  e  $C$  fa parte di  $B$  , anche  $C$  fa parte di  $A$
- Asimmetrico : non è possibile che  $A$  faccia parte di  $B$  e  $B$  far parte di  $A$  contemporaneamente.

Due tipi:

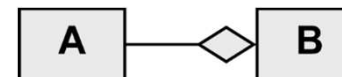
- Aggregazione condivisa
- Composizione



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Aggregazione condivisa



Esprime una debole appartenenza delle parti a un tutto

= Le parti esistono anche indipendentemente dal tutto

La molteplicità all'estremità dell'aggregazione può essere >1

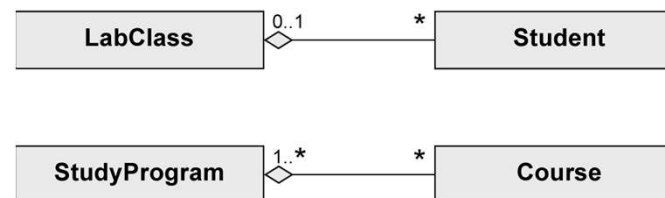
= Un elemento può far parte di più altri elementi contemporaneamente

Si estende su un grafo aciclico diretto

Sintassi: diamante vuoto alla fine dell'aggregazione

Esempio :

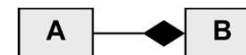
- **Studiante** fa parte di **LabClass**
- **Course** fa parte di un **StudyProgram**



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Composizione



Dipendenza di esistenza tra l'oggetto composto e le sue parti

Una parte può essere contenuta solo in al massimo un oggetto composto in un momento specifico

- Molteplicità alla fine dell'aggregazione max. 1
- -> Gli oggetti composti formano un albero

Se l'oggetto composto viene eliminato, vengono eliminate anche le sue parti.

Sintassi: diamante solido alla fine dell'aggregazione

Esempio: Beamer fa parte di LectureHall fa parte di Building



*Se il **Building** è cancellato ,  
il **LectureHall** viene anche cancellato*

*Il **Beamer** può esistere senza la  
**LectureHall** , ma se è contenuto nella  
**LectureHall** mentre è cancellato, il **Beamer**  
viene anche cancellato*



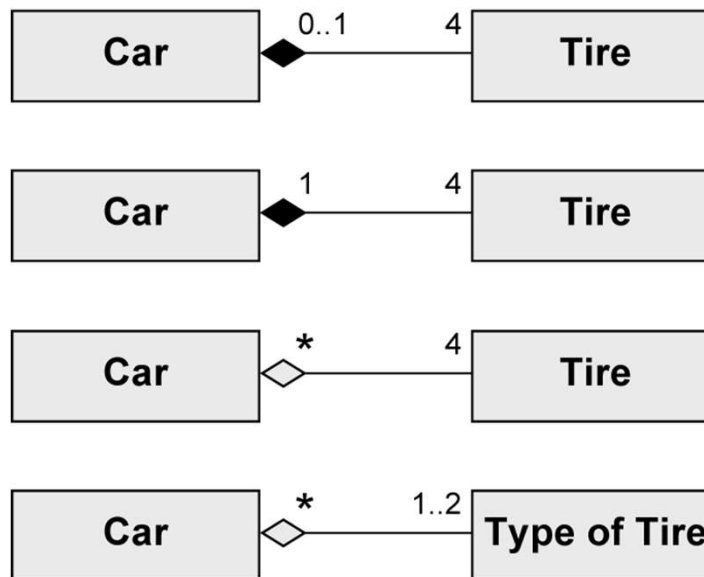
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione



# Aggregazione e composizione condivisa

Quale modello si applica?

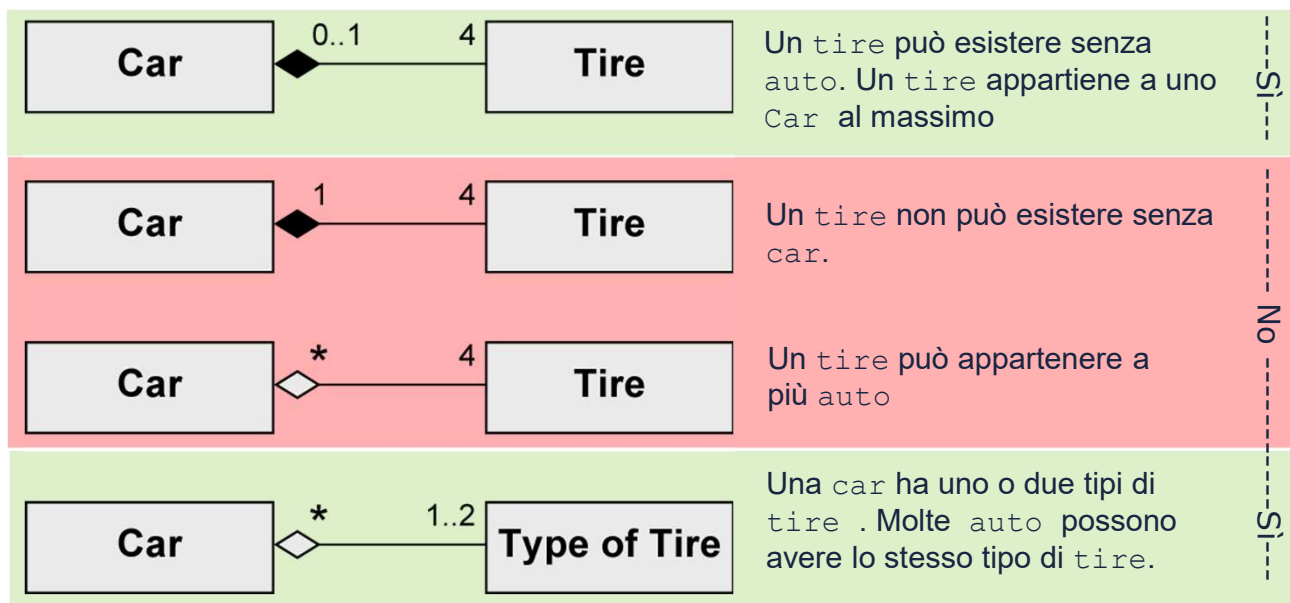


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Aggregazione e composizione condivisa

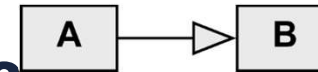
Quale modello si applica ?



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Generalizzazione/spacializzazione



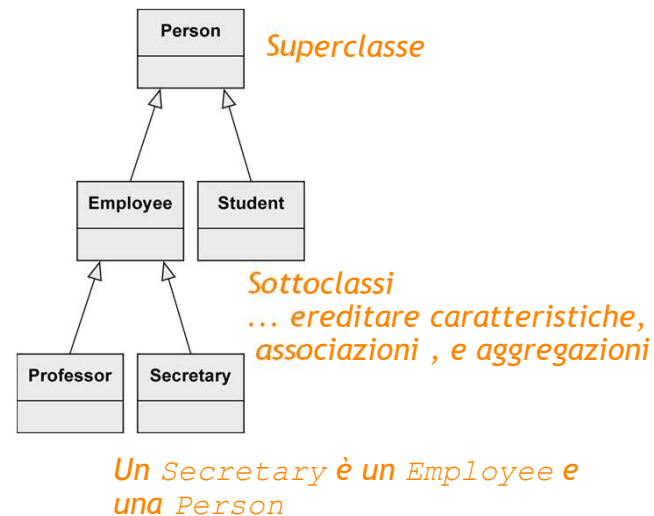
Le caratteristiche (attributi e operazioni), le associazioni e le aggregazioni specificate per una classe generale (superclasse) vengono *trasmesse* alle relative sottoclassi.

Ogni istanza di una sottoclasse è contemporaneamente un'istanza indiretta della superclasse.

La *sottoclasse* eredita tutte le caratteristiche, le associazioni e le aggregazioni della superclasse eccetto quelle private.

La *sottoclasse* può avere ulteriori caratteristiche, associazioni e aggregazioni.

Le generalizzazioni sono *transitive*.



# Generalizzazione – Classe astratta

Usato per evidenziare caratteristiche comuni delle loro sottoclassi.

Utilizzato per garantire che non vi siano istanze dirette della superclasse.

Solo le sue sottoclassi non astratte possono essere istanziate.

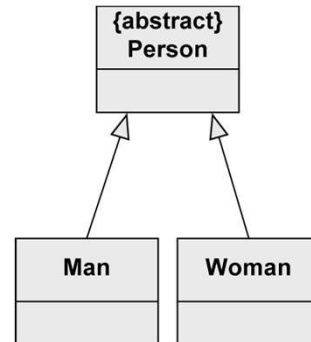
Utile nel contesto delle relazioni di generalizzazione.

Notazione: parola chiave **{abstract}** o nome della classe in corsivo.

**{abstract}**  
**A**

**{abstract}**  
**Person**

*Person*



*Nessun oggetto Person è possibile*

*Due tipi di Person : Man e Woman*



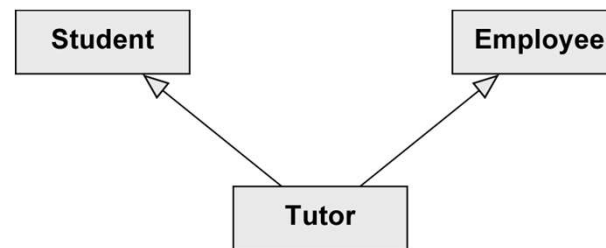
UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Generalizzazione – Ereditarietà multipla

UML consente l'ereditarietà multipla  
Una classe può avere più superclassi

Esempio:



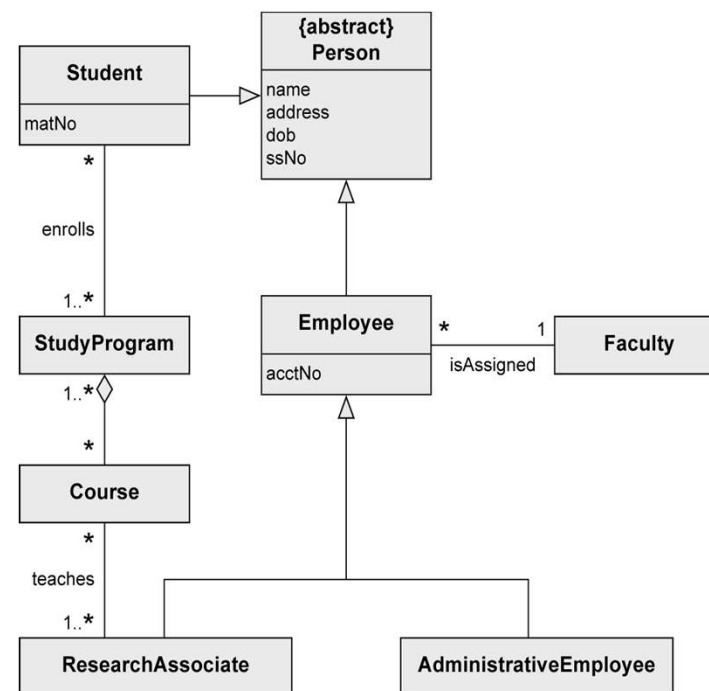
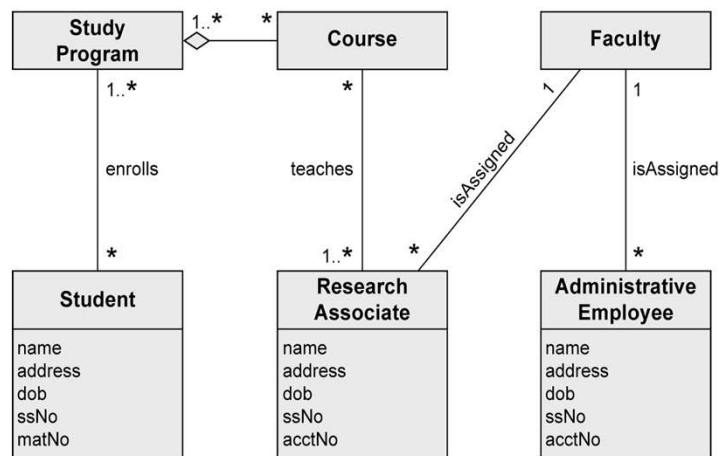
*Un Tutor è entrambi un Employee e uno Student*



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Con e senza generalizzazione



# Creazione di un diagramma di classe

Non è possibile estrarre completamente classi, attributi e associazioni da un testo in linguaggio naturale automaticamente.

Linee guida

- I nomi spesso indicano classi
- Gli aggettivi indicano i valori degli attributi
- I verbi indicano operazioni

**Esempio:** il sistema di gestione della biblioteca memorizza gli utenti con il loro ID univoco, nome e indirizzo, nonché i libri con il titolo, l'autore e il numero ISBN. Ann Foster vuole usare la biblioteca.



*Domanda : Cosa sai su Ann Foster?*



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Esempio – Sistema Informativo di Ateneo

Un'università è composta da più facoltà che sono composte da vari istituti. Ogni facoltà e ogni istituto ha un nome. Un indirizzo è noto per ogni istituto.

Ogni facoltà è guidata da un preside, che è un dipendente dell'università.

Il numero totale dei dipendenti è noto. I dipendenti hanno un numero di previdenza sociale, un nome e un indirizzo e-mail. C'è una distinzione tra personale di ricerca e personale amministrativo.

Gli assegnisti di ricerca sono assegnati ad almeno un istituto. Il campo di studio di ogni ricercatore associato è noto. Inoltre, i ricercatori associati possono essere coinvolti nei progetti per un certo numero di ore e sono noti il nome, la data di inizio e la data di fine dei progetti. Alcuni ricercatori associati tengono corsi. Poi sono chiamati docenti.

I corsi hanno un numero univoco (ID), un nome e una durata settimanale in ore.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione



## Esempio – Fase 1: Identificazione delle classi

*modello il sistema "Università"*

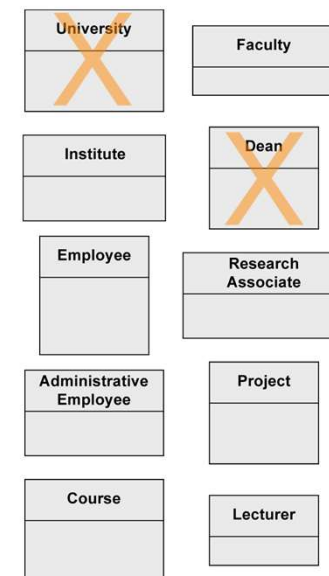
Un'università è composta da più facoltà che sono composte da vari istituti. Ogni facoltà e ogni istituto ha un nome. Un indirizzo è noto per ogni istituto.

Ogni facoltà è guidata da un preside (dean), che è un dipendente dell'università.

Il numero totale dei dipendenti è noto. I dipendenti hanno un numero di previdenza sociale, un nome e un indirizzo e-mail. C'è una distinzione tra personale di ricerca e personale amministrativo.

Gli assegnisti di ricerca sono assegnati ad almeno un istituto. Il campo di studio di ogni ricercatore associato è noto. Inoltre, i ricercatori associati possono essere coinvolti nei progetti per un certo numero di ore e sono noti il nome, la data di inizio e la data di fine dei progetti. Alcuni ricercatori associati tengono corsi. Poi sono chiamati docenti.

I corsi hanno un numero univoco (ID), un nome e una durata settimanale in ore.



*Dean non ha ulteriori attributi di qualunque altro dipendente*



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Esempio – Fase 2: Identificazione degli attributi

Un'università è composta da più facoltà che sono composte da vari istituti. Ogni facoltà e ogni istituto ha un nome. Un indirizzo è noto per ogni istituto.

Ogni facoltà è guidata da un preside, che è un dipendente dell'università.

Il numero totale dei dipendenti è noto. I dipendenti hanno un numero di previdenza sociale, un nome e un indirizzo e-mail. C'è una distinzione tra personale di ricerca e personale amministrativo.

Gli assegnisti di ricerca sono assegnati ad almeno un istituto. Il campo di studio di ogni ricercatore associato è noto. Inoltre, i ricercatori associati possono essere coinvolti nei progetti per un certo numero di ore e sono noti il nome, la data di inizio e la data di fine dei progetti. Alcuni ricercatori associati tengono corsi. Poi sono chiamati docenti.

I corsi hanno un numero univoco (ID), un nome e una durata settimanale in ore.

Faculty
+ name: String

Institute
+ name: String
+ address: String

Employee
+ ssNo: int
+ name: String
+ email: String
+ <u>counter</u> : int

Research Associate
+ fieldOfStudy: String

Administrative Employee

Project
+ name: String
+ start: Date
+ end: Date

Course
+ name: String
+ id: int
+ hours: float

Lecturer



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Esempio – Fase 2: Identificazione delle relazioni (1/6)

Tre tipi di relazioni:

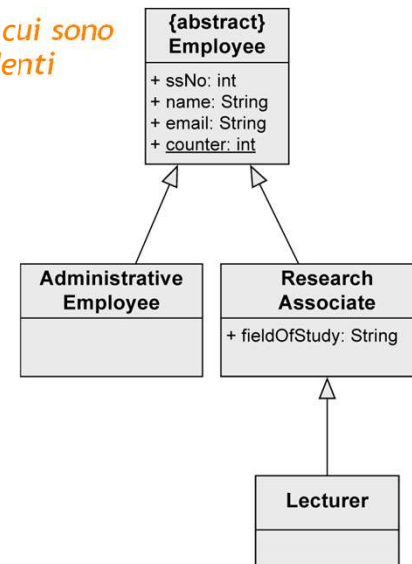
- Associazione
- Generalizzazione
- Aggregazione

Indicazione di una generalizzazione

*"C'è una distinzione tra personale di ricerca e personale amministrativo".*

*"Alcuni ricercatori associati tengono dei corsi. Nel caso sono chiamati lecturers".*

*Astratto, cioè non cui sono  
Altri tipi di dipendenti*

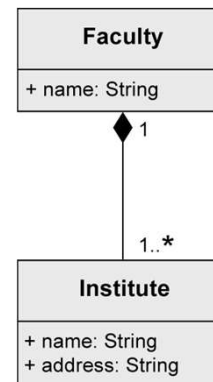


UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Esempio – Fase 2: Identificazione delle relazioni (2/6)

*"Un'università è composta da più facoltà che sono composte da vari istituti".*

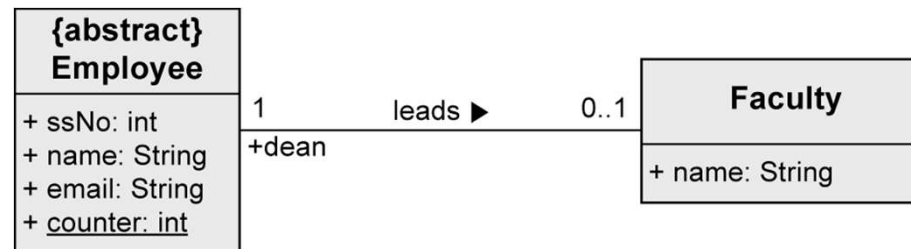


*Composizione a mostrare esistenza dipendenza*



## Esempio – Fase 2: Identificazione delle relazioni (3/6)

*“Ogni facoltà è guidata da un preside, che è un impiegato dell'università”*

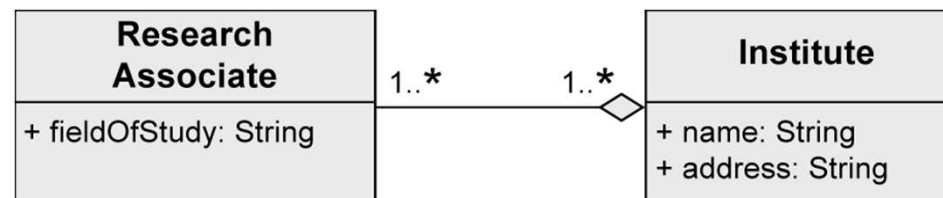


*Nella relazione leads, il  
Dipendente prende il ruolo di un dean (preside) .*



## Esempio – Fase 2: Identificazione delle relazioni (4/6)

"Gli associati di ricerca sono assegnati ad almeno un istituto."

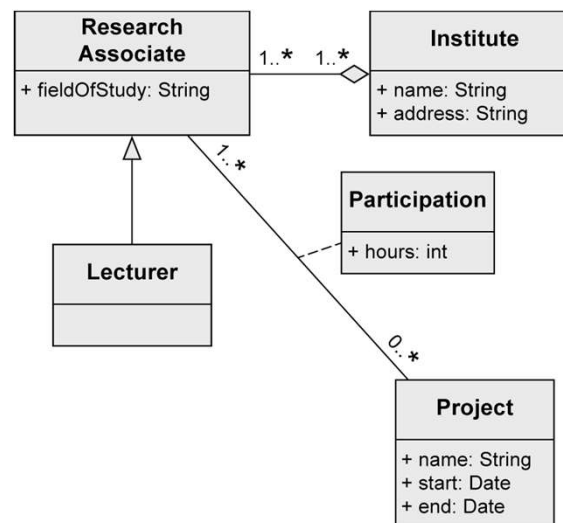


*Aggregazione Condivisa per mostrare che Associati di ricerca sono parte di un Istituto , ma non c'è dipendenza di esistenza*



## Esempio – Fase 2: Identificazione delle relazioni (5/6)

*"Inoltre, i ricercatori associati possono essere coinvolti nei progetti per un certo numero di ore".*

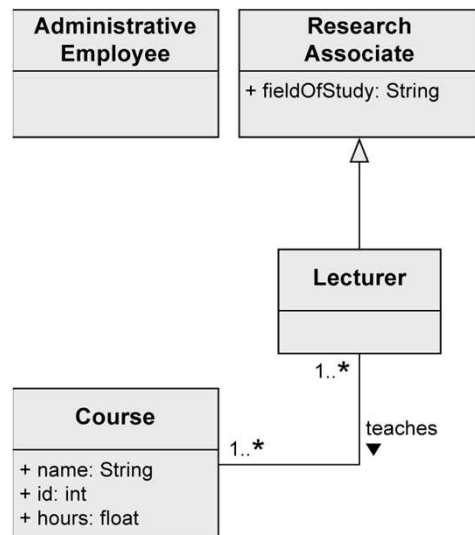


*La classe associazione consente di memorizzare il numero di ore per ogni singolo Progetto di ogni singolo ResearchAssociate*



## Esempio – Fase 2: Identificazione delle relazioni (6/6)

*“Alcuni ricercatori associati tengono dei corsi. In quel caso sono chiamati docenti”.*

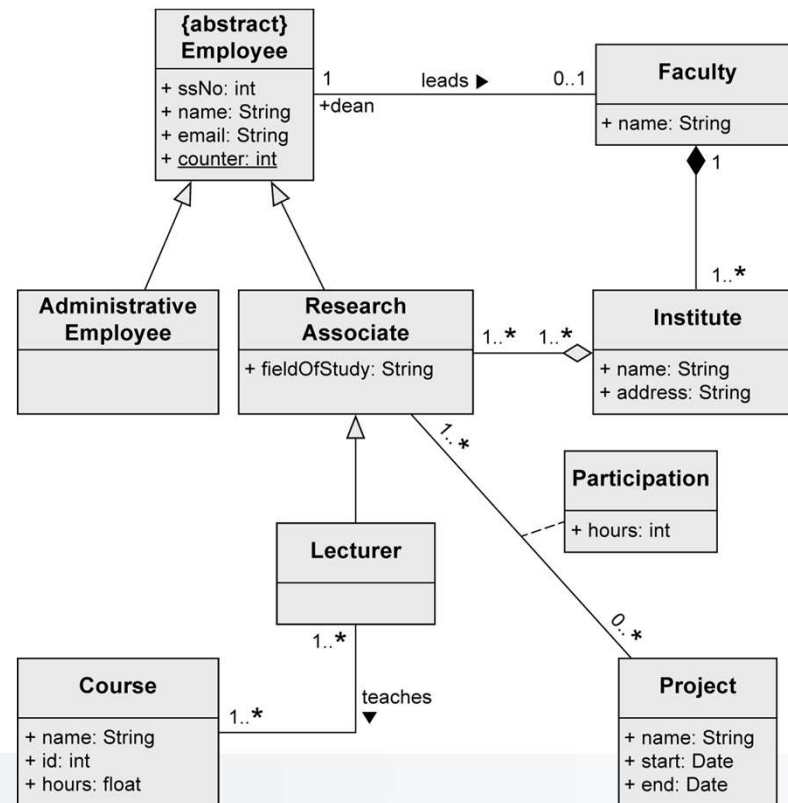


*Il Lecturer eredita tutte le caratteristiche, associazioni e aggregazioni da ResearchAssociate. Inoltre, un Lecturer ha un'associazione insegna al Corso.*





## Esempio: diagramma di classe completo



# Generazione di codice

I diagrammi di classe vengono spesso creati con l'intenzione di implementare gli elementi modellati in un linguaggio di programmazione orientato agli oggetti.

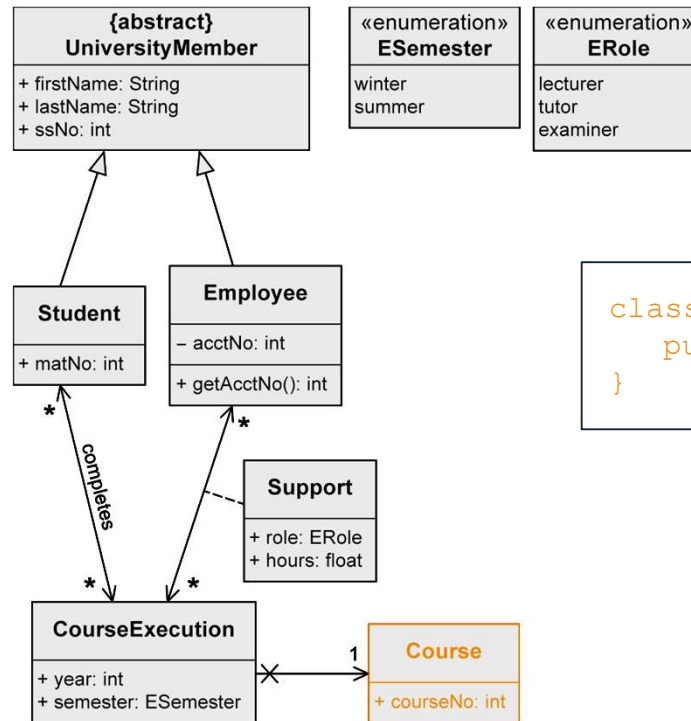
Spesso la traduzione è semiautomatica e richiede solo un minimo intervento manuale.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

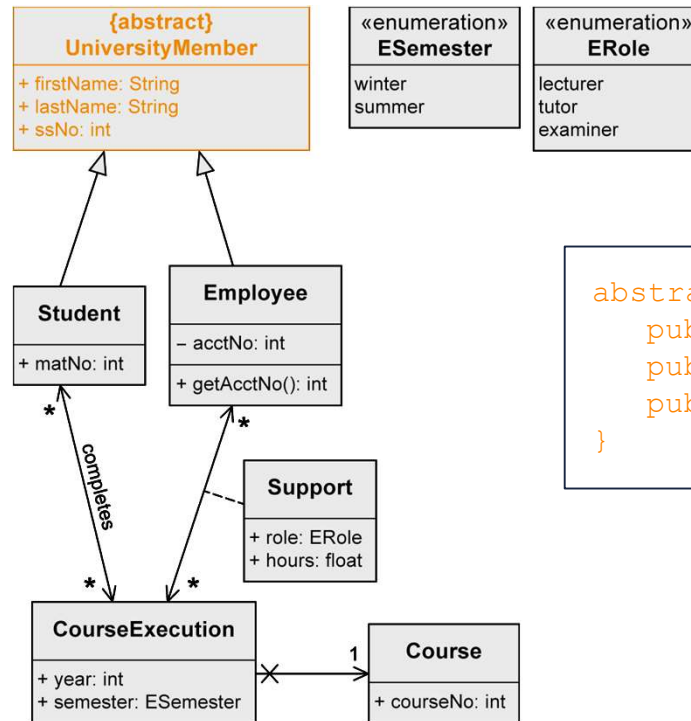
## Generazione di codice – Esempio (1/6)



```
class Course {
    public int courseNo;
}
```



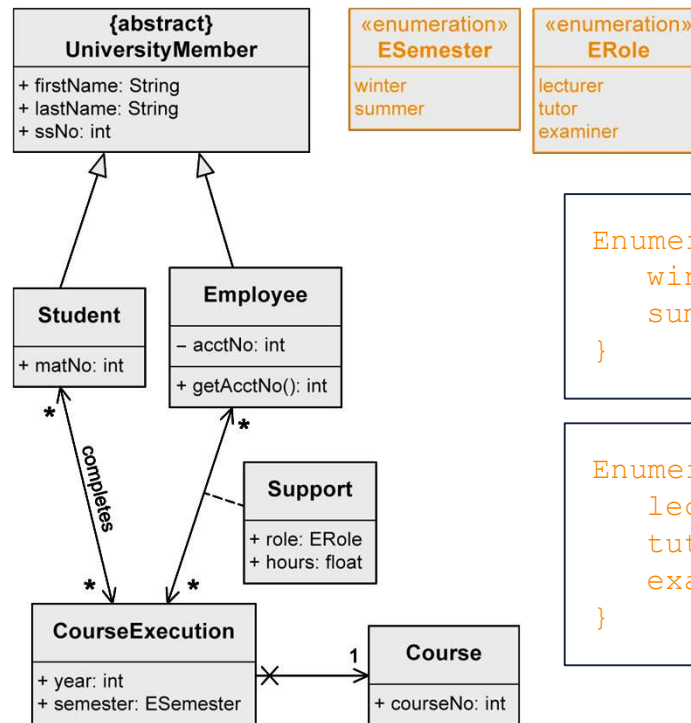
## Generazione di codice – Esempio (2/6)



```
abstract class UniversityMember {
    public String firstName;
    public String lastName;
    public int ssNo;
}
```



## Generazione di codice – Esempio (3/6)



```

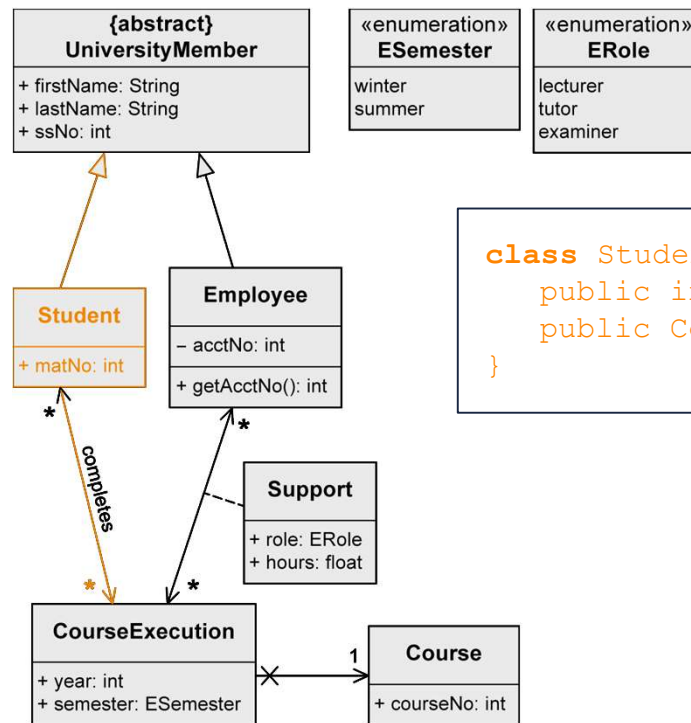
Enumeration ESemester {
    winter,
    summer
}
    
```

```

Enumeration ERole {
    lecturer,
    tutor,
    examiner
}
    
```



## Generazione di codice – Esempio (4/6)



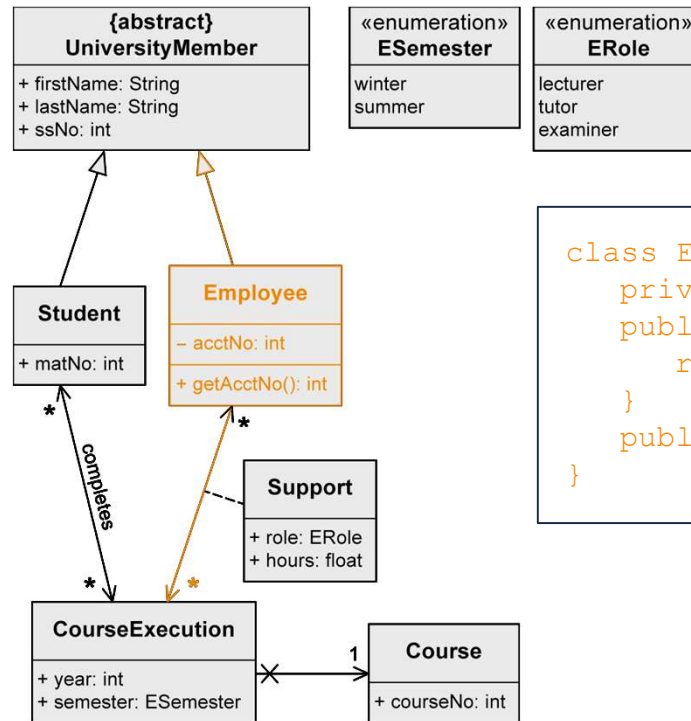
```

class Student extends UniversityMember {
    public int matNo;
    public CourseExecution [] completedCourses;
}
    
```

È possibile utilizzare List/Set al posto degli array: dipende dal generatore



## Generazione di codice – Esempio (5/6)

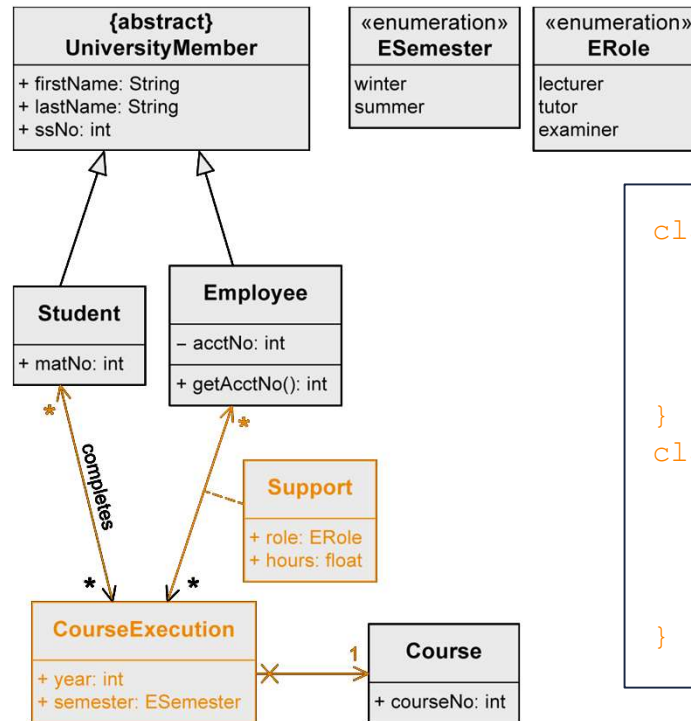


```

class Employee extends UniversityMember {
    private int acctNo;
    public int getAcctNo () {
        return acctNo;
    }
    public CourseExecution [] courseExecutions;
}
    
```



## Generazione di codice – Esempio (6/6)



```

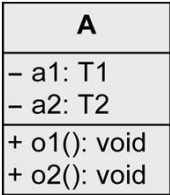
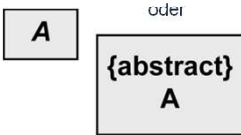
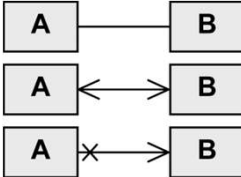
class CourseExecution {
    public int year;
    public ESemester semester;
    public Student [] student;
    public Course course;
}

class Support {
    public Erole role;
    public float hours;
    public Empolyee employee;
    public CourseExecution courseExecution;
}
    
```



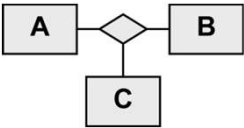
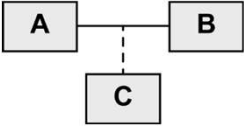
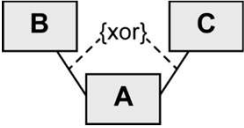


## Elementi di notazione (1/3)

Nome	Notazione	Descrizione
Classe		Descrizione della struttura e del comportamento di un insieme di oggetti
Classe astratta		Classe che non può essere istanziata
Associazione		Rapporto tra classi: navigabilità non specificata, navigabile in entrambe le direzioni, non navigabile in una direzione



## Elementi di notazione (2/3)

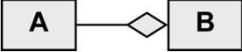

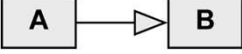

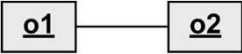
Nome	Notazione	Descrizione
n- associazione _		Relazione tra n (qui 3) classi
Classe di associazione		Descrizione più dettagliata di un'associazione
<b>xor</b> relazione		Un oggetto di <b>c</b> è in relazione con un oggetto di <b>A</b> o con un oggetto di <b>B</b> ma non con entrambi



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Elementi di notazione (3/3)

Nome	Notazione	Descrizione
Aggregazione condivisa		Relazione parti-tutto ( <b>A</b> è parte di <b>B</b> )
Forte aggregazione = composizione		Relazione parti-intero dipendente dall'esistenza ( <b>A</b> è parte di <b>B</b> )
Generalizzazione		Relazione di eredità ( <b>A</b> eredita da <b>B</b> )
Oggetto		Istanza di una classe
Collegamento		Relazione tra oggetti



# ESERCIZI



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

# Esercizio: Rivendita auto usate



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Rivendita auto usate

Progettiamo un'applicazione per la gestione di una rivendita di auto usate. Un automezzo viene identificato dalla targa, dal numero di telaio e da un certo numero di caratteristiche specifiche, quali il colore, la cilindrata, il tipo di carburante e gli optional. Ogni automezzo è caratterizzato da una "carta di identità" che definisce l'anno d'immatricolazione, il numero di chilometri e la data dell'ultima revisione. Il sistema gestisce anche camion e van, che si differenziano dalle automobili per la capacità di carico (quintali o persone). Il sistema cataloga anche i clienti, con le solite caratteristiche: nome, cognome, indirizzo e codice fiscale. Un cliente può stipulare uno o più contratti per acquistare uno o più automezzi. Ogni contratto deve avere una data di stipula, un ammontare, una data di inizio validità ed eventuali dilazioni di pagamento pattuite tra le parti. Il sistema deve anche gestire lo storico dei diversi automezzi, cioè la storia del mezzo che contiene tutti i passaggi di proprietà noti al rivenditore.



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'Informazione e della Produzione

## Rivendita auto usate: identificazione delle classi

Progettiamo un'applicazione per la gestione di una rivendita di auto usate. Un **automezzo** viene identificato dalla targa, dal numero di telaio e da un certo numero di caratteristiche specifiche, quali il colore, la cilindrata, il tipo di carburante e gli **optional**. Ogni automezzo è caratterizzato da una "**carta di identità**" che definisce l'anno d'immatricolazione, il numero di chilometri e la data dell'ultima revisione. Il sistema gestisce anche **camion** e **van**, che si differenziano dalle automobili per la capacità di carico (quintali o persone). Il sistema cataloga anche i **clienti**, con le solite caratteristiche: nome, cognome, **indirizzo** e codice fiscale. Un cliente può stipulare uno o più **contratti** per acquistare uno o più automezzi. Ogni contratto deve avere una data di stipula, un ammontare, una data di inizio validità ed eventuali **dilazioni di pagamento** pattuite tra le parti. Il sistema deve anche gestire lo storico dei diversi automezzi, cioè la storia del mezzo che contiene tutti i passaggi di proprietà noti al rivenditore.



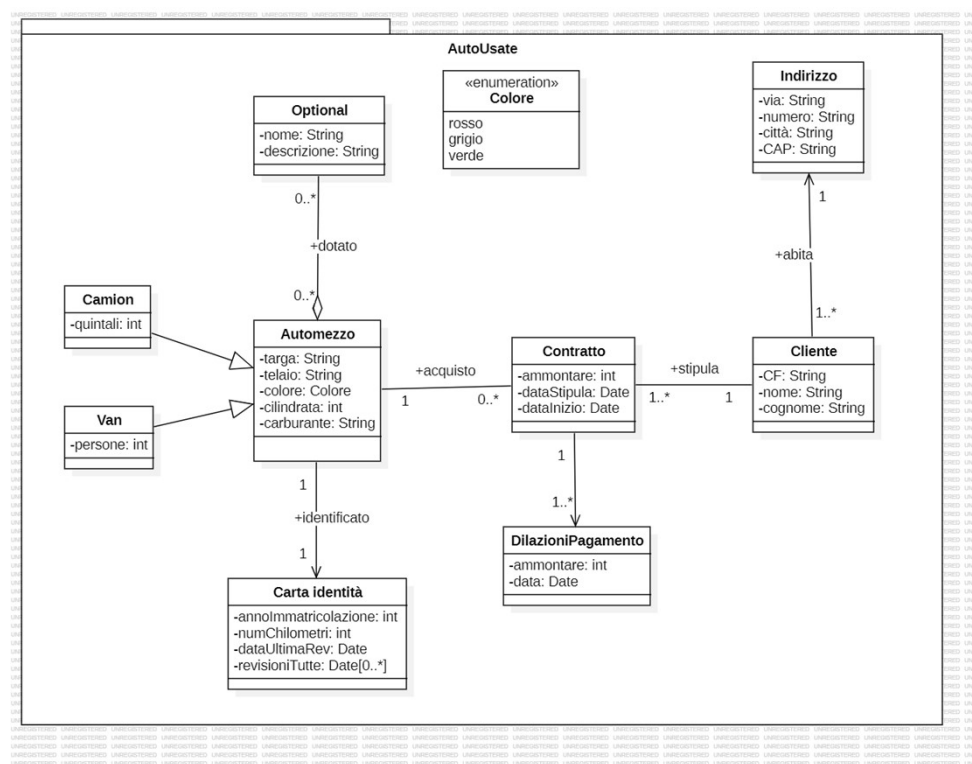
## Rivendita auto usate: identificazione degli attributi

Progettiamo un'applicazione per la gestione di una rivendita di auto usate. Un **automezzo** viene identificato dalla **targa**, dal **numero di telaio** e da un certo numero di caratteristiche specifiche, quali il **colore**, la **cilindrata**, il **tipo di carburante** e gli **optional**. Ogni automezzo è caratterizzato da una "**carta di identità**" che definisce l'anno d'immatricolazione, il numero di chilometri, la **data dell'ultima revisione** e di tutte le **precedenti**. Il sistema gestisce anche **camion** e **van**, che si differenziano dalle automobili per la **capacità di carico** (quintali o persone). Il sistema cataloga anche i **clienti**, con le solite caratteristiche: **nome**, **cognome**, **indirizzo** e **codice fiscale**. Un cliente può stipulare uno o più **contratti** per acquistare uno o più automezzi. Ogni contratto deve avere una **data** di stipula, un **ammontare**, una **data di inizio** validità ed eventuali **dilazioni di pagamento** pattuite tra le parti. Il sistema deve anche gestire lo storico dei diversi automezzi, cioè la storia del mezzo che contiene tutti i passaggi di proprietà noti al rivenditore.





# Rivendita auto usate



# Esercizio: Compagnia aerea MyAir



UNIVERSITÀ  
DEGLI STUDI  
DI BERGAMO

Dipartimento  
di Ingegneria Gestionale,  
dell'informazione e della Produzione

# Compagnia aerea MyAir

Progettiamo un'applicazione per la gestione del programma fedeltà della compagnia aerea *MyAir*. Chi si iscrive al programma, ogni volta che vola con MyAir, accumula punti (miglia) che danno diritto a premi. Ad esempio, bisogna volare per almeno 25.000 miglia per avere diritto a un volo gratuito in Europa; ci vogliono 65.000 miglia per un volo negli Stati Uniti; bastano 5.000 miglia per un buono acquisto in un negozio convenzionato. Il sistema deve gestire i clienti della compagnia che partecipano al programma. I partecipanti sono organizzati in tre fasce di merito in funzione delle miglia volate durante un anno solare: tutti appartengono al primo livello. Se si volano 35.000 miglia si passa al secondo livello; si accede al terzo livello con 100.000 miglia volate in un anno. I tre livelli danno diritto a facilitazioni e premi differenziati. Oltre ai clienti, il sistema deve gestire i tipi di premi (volo gratuito, soggiorno gratuito, buono sconto), il numero di miglia necessarie per ogni premio particolare (un volo gratuito a New York richiede più miglia di un volo per Roma) e lo storico dei clienti: quanti voli ha effettuato ogni cliente, quante miglia ha guadagnato, quali premi ha già riscosso e quante miglia gli restano da "spendere". Teniamo presente che le miglia scadono dopo 5 anni dal momento in cui sono state acquisite, cioè dalla data del volo. Il sistema deve essere in grado di aggiornare la posizione di ogni cliente in funzione di ogni volo effettuato e di ogni premio richiesto. Deve anche gestire l'effettiva disponibilità dei premi. Ad esempio, un volo gratuito potrebbe non essere soddisfacibile se il volo richiesto fosse già pieno.



# Compagnia aerea MyAir

