

Unified Modeling Language

- Class Diagram: concetti di base -

Henry Muccini
Università degli Studi dell'Aquila
muccini@di.univaq.it



Engineering | IgTechnology | Imola
Informatica | Maggioli Informatica |
Micron Technology | Neta | Nous
Informatica | ObjectWay SED |
TechnoLabs | Taiprora

Master in Web Technology

IV Edizione 2007/08

Dipartimento di Informatica

Università degli Studi dell'Aquila

Copyright Notice

- » Il materiale riportato in queste slide puo' essere riutilizzato, parzialmente of totalmente, a patto che le fonti e gli autori vengano citati.

Henry Muccini



Use Case Diagram: riassunto

- » Per riassumere i diagrammi mostrati la lezione precedente, far vedere:
 - > Elementi costituenti
 - > Semantica
 - > Sintassi di integrazione

- » Far vedere anche le relazioni tra i diagrammi



Class Diagram

- » Oggetto
- » Classe
 - > Attributo
 - > Operazione
- » Note
- » Visibilita'
- » Package

- » Relazioni:
 - > Dipendenza
 - > Associazione
 - Aggregazione
 - Composizione
 - > Generalizzazione

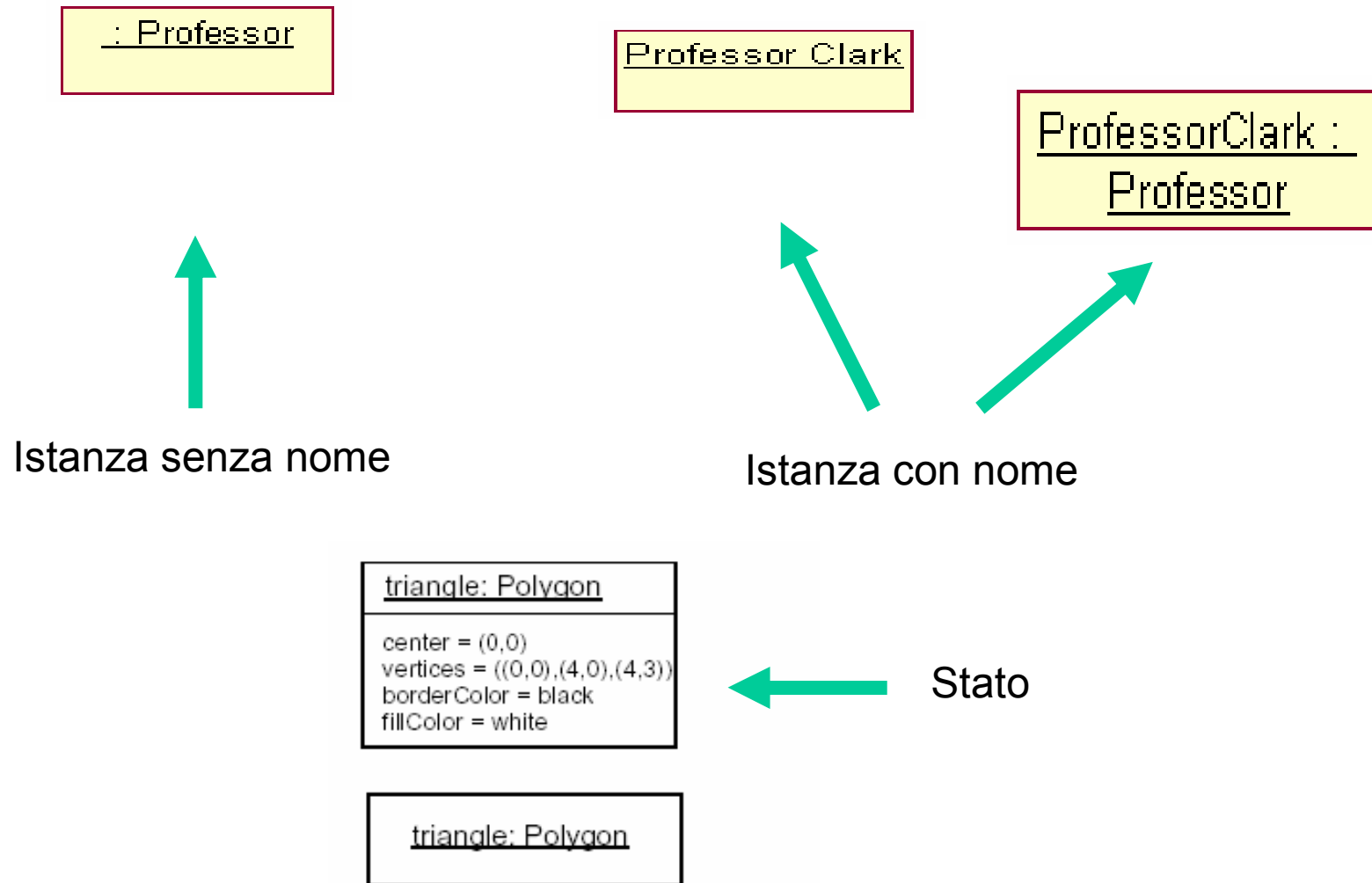


Oggetto

- » Informalmente un oggetto rappresenta un **entità fisica, concettuale o software**
 - > entità fisica: trattore
 - > entità concettuale: processo chimico
 - > entità software: lista, coda...
- » Formalmente
 - > Manifestazione concreta di un'astrazione
 - > Entità con un confine e un'*identità* ben definite che incapsula *stato e comportamento*
 - > **Istanza di una classe**
- » Es.: Ferrari di Schumacher, Ferrari di Barrichello, Mio computer



Rappresentazione in UML

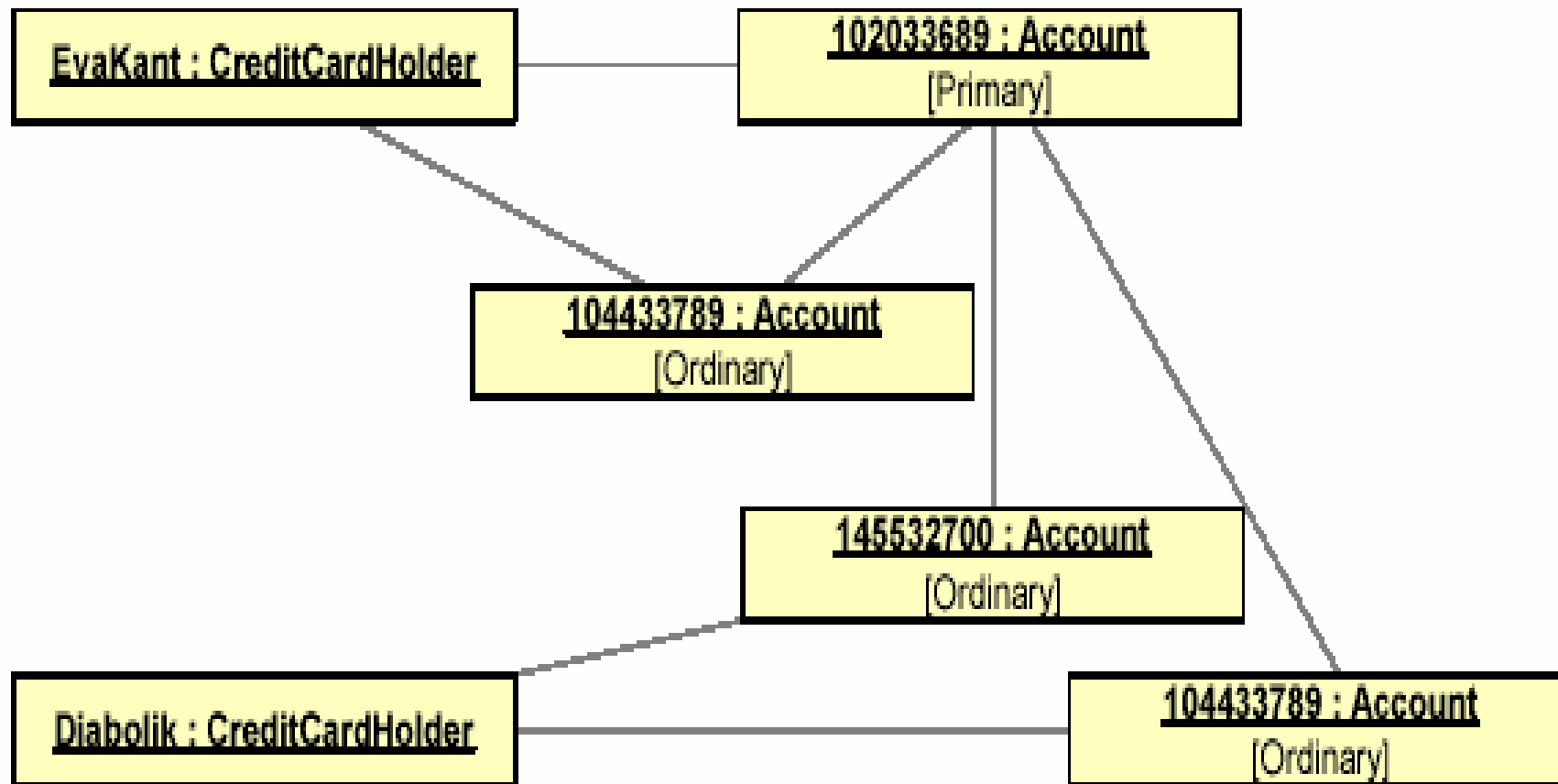


Object Diagram

- » Mostra un insieme di **oggetti** e le loro **relazioni** (links) in un determinato istante di tempo
- » E' una **istanza** di un class diagram
- » Generalmente viene utilizzato per descrivere una fotografia di un sistema complesso



Object Diagram



Class Diagram

- » Mostra un insieme di **classi** e le loro **relazioni** (dipendenza, associazione e generalizzazione)
- » Può essere visto come un **grafo** dove i nodi sono le classi e le interfacce, e gli archi sono le relazioni
- » Possono contenere anche **package** o **sottosistemi** (utilizzati per raggruppare elementi)



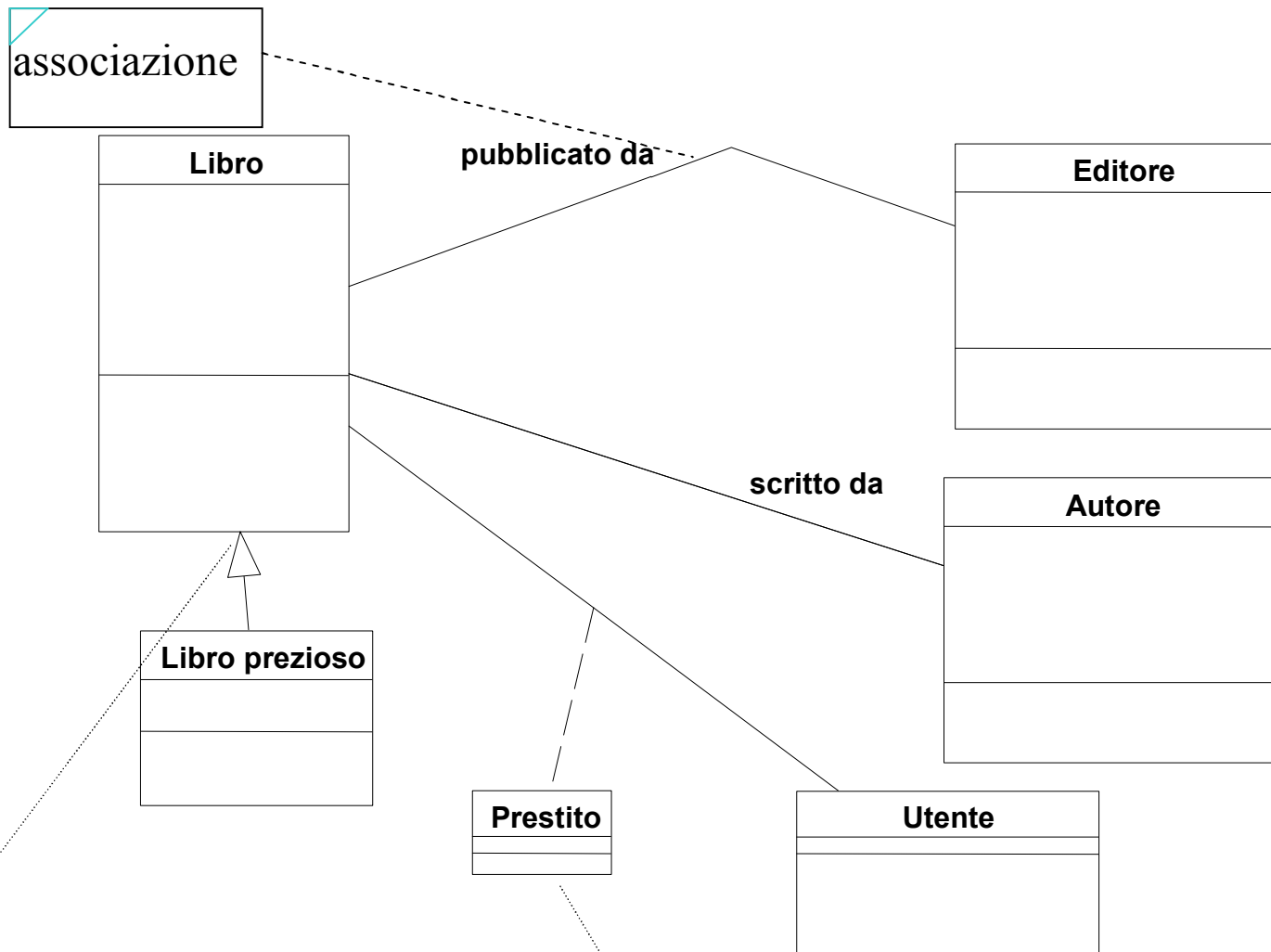
Class Diagram

- » Modellano la parte **statica** di un sistema. In particolare servono per
 - > Modellare il **vocabolario** di un sistema
 - Classi modellano astrazioni di cose di un problema
 - Tali astrazioni fanno parte del vocabolario di un sistema
 - > Modellare **semplici collaborazioni**
 - Classi non vivono da sole
 - Collaborano insieme per fornire un comportamento che è più grande della somma di tutti gli elementi
 - > Modellare un **schema logico di un database**
 - Al posto degli schemi ER



Esempio: Sistema per l'archiviazione bibliografica di *testi*, memorizzando informazioni sull'*editore* e sull'*autore* e identificando quelli che sono considerati i *libri preziosi*. Si vuol inoltre gestire una operazione di *prestito* da parte di uno o piu' *utenti*





associazione

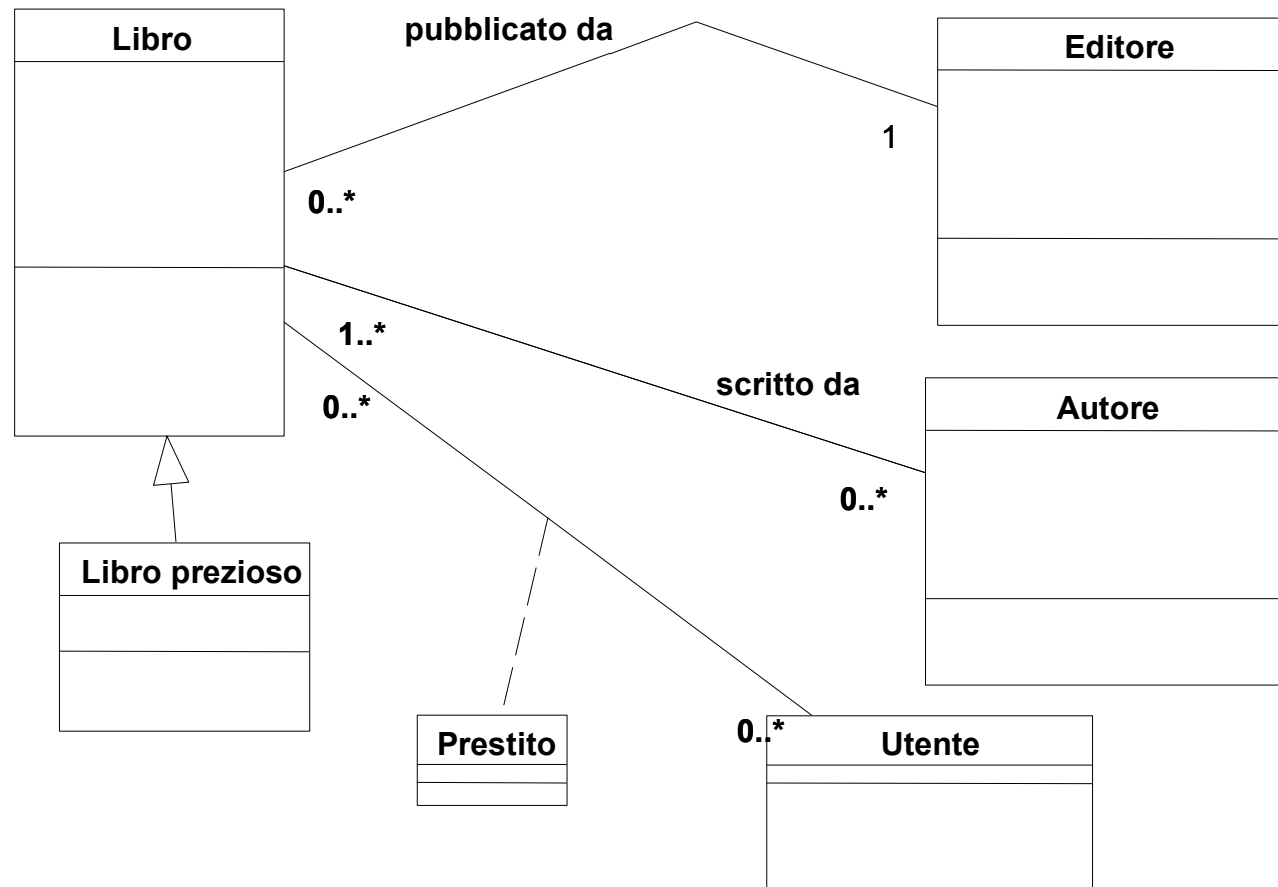
pubblicato da

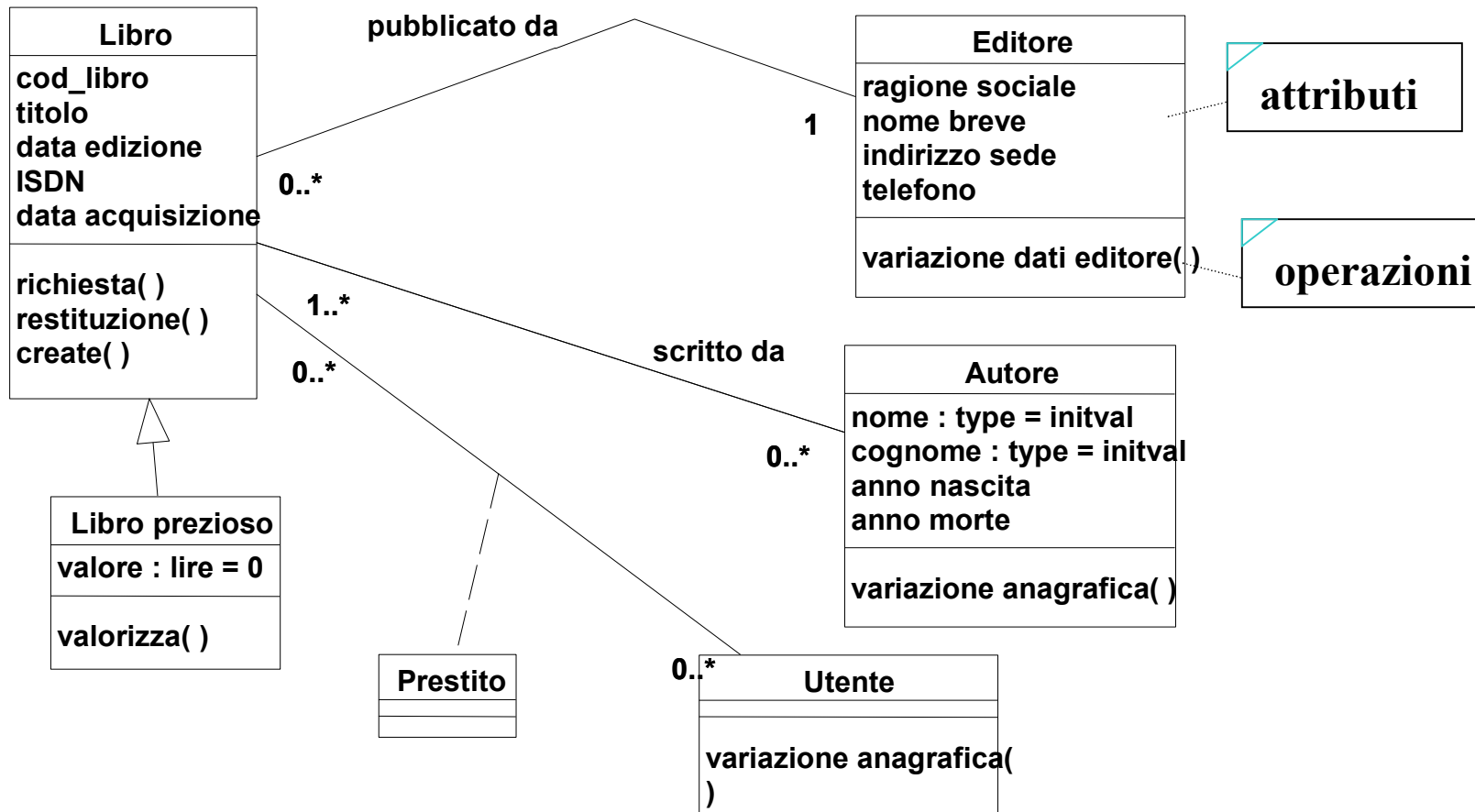
scritto da

gerarchia di specializzazione
(superclasse- sottoclasse)

Classe associata







Classe

- » Descrizione di un **gruppo di oggetti** con **proprietà** (attributi), **comportamento** (operazioni), **relazioni e semantica comuni**
 - > Un oggetto è una istanza di una classe
- » Astrazione che
 - > Enfatizza caratteristiche rilevanti
 - > **Sopprime le altre caratteristiche**

Principio O.O. Astrazione



Esempio di classe

- » Nome
 - > Corso
- » Proprietà
 - > Nome, Luogo, Durata, Crediti, Inizio, Fine
- » Comportamento
 - > Aggiunta studente
 - > Cancellazione studente
 - > Verifica se è pieno

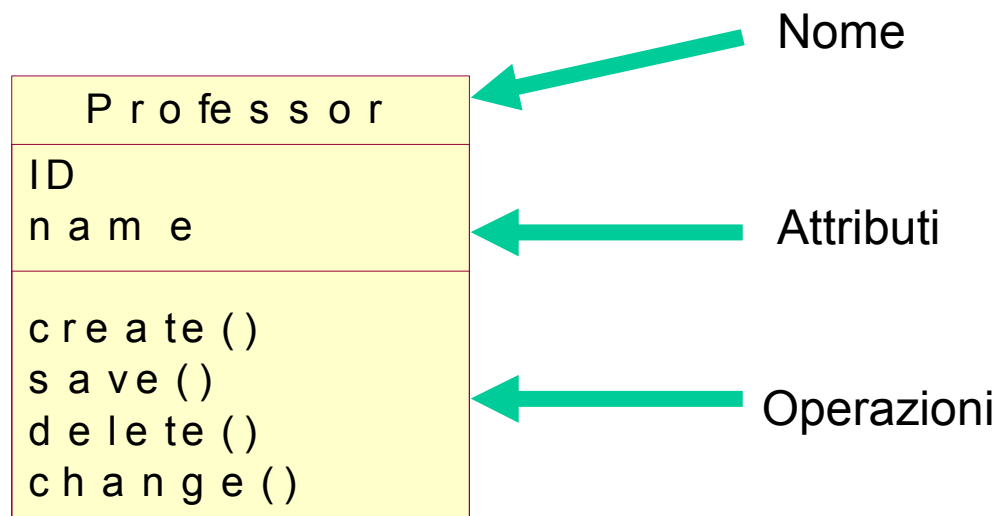
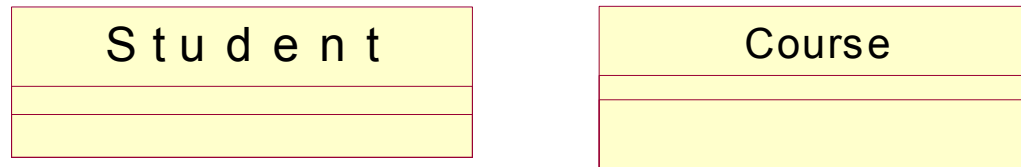


Relazione tra classe ed oggetto

- » Classe è una definizione astratta di un oggetto
 - > Definisce la struttura e il comportamento di ogni oggetto nella classe
 - > Serve come *template* per creare oggetti
- » Oggetti sono raggruppati in classi



Rappresentazione in UML



Classe: Nome

- » Rappresenta un nome, cioè un'entità
- » Stringa di testo
 - > lettere
 - > numeri
 - > alcuni caratteri speciali
- » Nome
 - > Semplice
 - > Path (prefisso + “:” + nome classe)
- » Convenzione
 - > Lettera iniziale maiuscola



Classe: Nome

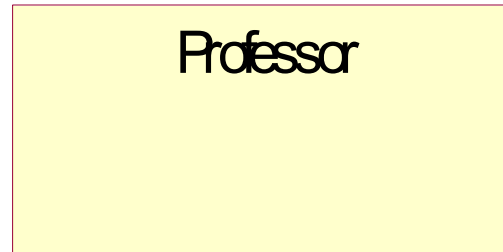
Professor

DBManager::Professor

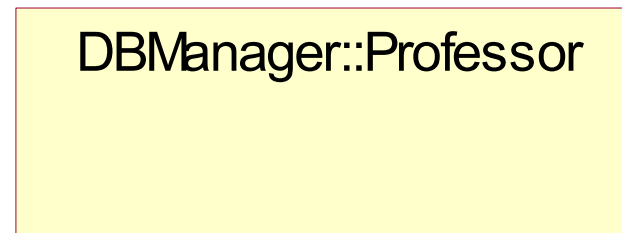


Java

```
public class Professor {  
    /*  
        Corpo della classe  
    */  
}
```



```
package DBManager;  
public class Professor {  
    /*  
        Corpo della classe  
    */  
}
```



Classe: Attributi

- » Proprietà di una classe che descrive un **insieme di valori** che le istanze degli attributi possono assumere
- » Rappresenta **proprietà** delle cose che si stanno modellando che è **condivisa da tutti gli oggetti di quella classe**

- » Esempio
 - > Muro ha un'altezza, ampiezza e profondità
 - > Cliente ha un nome, indirizzo, numero di telefono



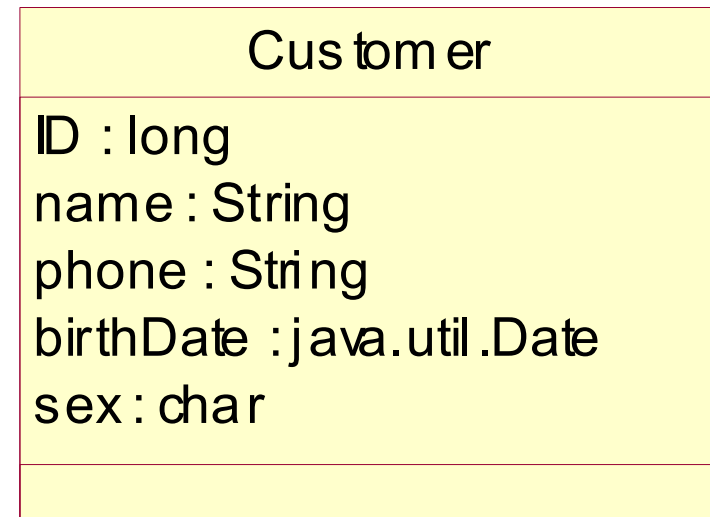
Classe: Attributi

- » Nome
 - > Stringa di testo
- » Tipo
 - > Es. int, float, double,
- » Valore di iniziale
- » Convenzione
 - > Lettera iniziale minuscola



Java

```
public class Customer {  
  
    long ID;  
    String name;  
    String phone;  
    java.util.Date birthDate;  
    char sex = 'F';  
}
```



Classe: Operazioni

- » Implementazione di un **servizio** che può essere richiesto da qualsiasi oggetto
- » E' un'astrazione di qualcosa che è **condivisa** tra tutti gli **oggetti di quella classe**
- » Rappresenta un **verbo** o una frase
- » Generalmente l'invocazione di un'operazione cambia lo stato dell'oggetto
- » Esempi
 - > Rettangolo ha operazioni di move, resize



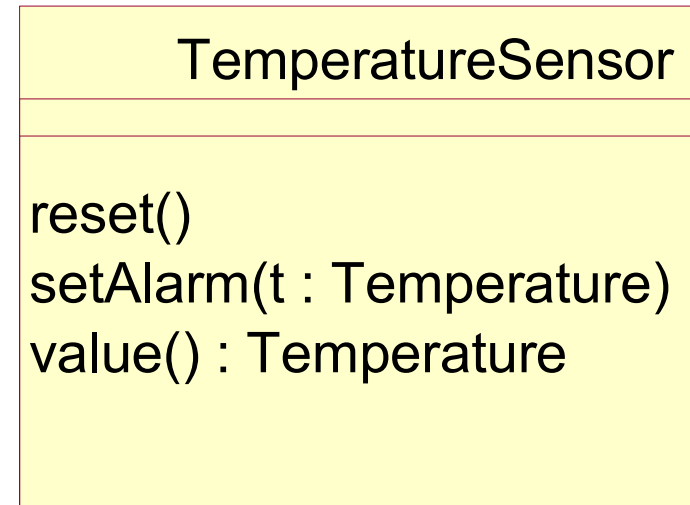
Classe: Operazioni

- » Nome
 - > Stringa di testo
- » Segnatura
 - > lista separata da virgola di
 - Nome, tipo, valore di default
- » Tipo ritorno
 - > Per le “funzioni”
- » Convenzione
 - > Lettera iniziale minuscola

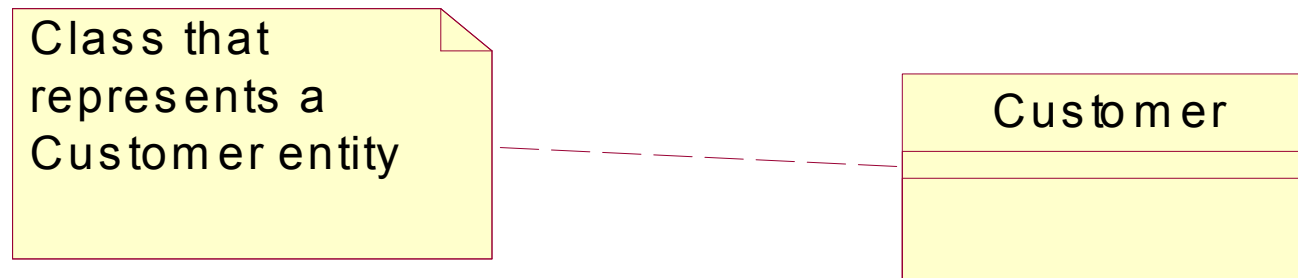


Java

```
public class TemperatureSensor {  
    void reset() {  
        .....  
    }  
    void setAlarm(Temperature t) {  
        .....  
    }  
    Temperature value() {  
        .....  
    }  
}
```



Note



Publish this component in the project repository after the next design review
ads 01/05/00

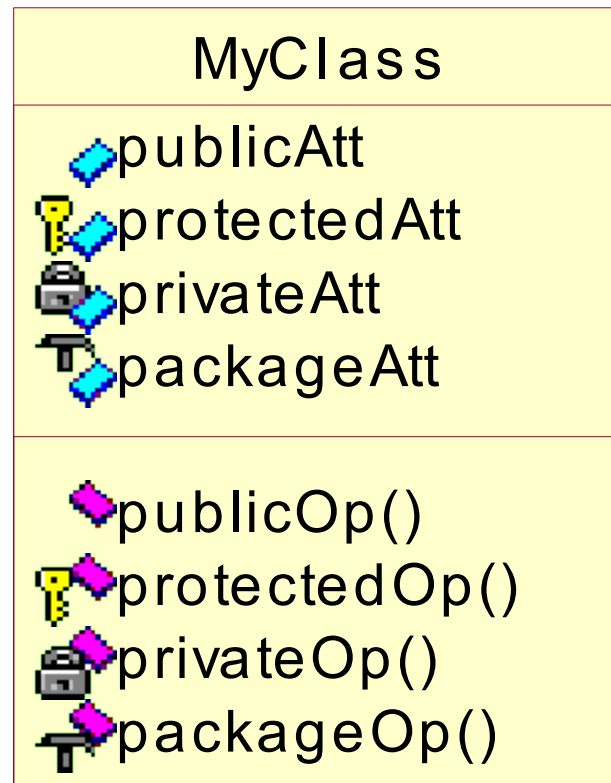


Visibilità

- » Possono essere applicati alle **classi, attributi e operazioni**
- » Tre livelli di visibilità
 - > **public**: qualsiasi classe può vedere tale feature (simbolo utilizzato +)
 - > **private**: soltanto la classe può vedere tale feature (simbolo utilizzato -)
 - > **protected**: solo le classi derivate possono vedere tale feature (simbolo utilizzato #)
 - > **package**: solo le classi che si trovano all'interno del package possono vedere tale feature (simbolo utilizzato ~)



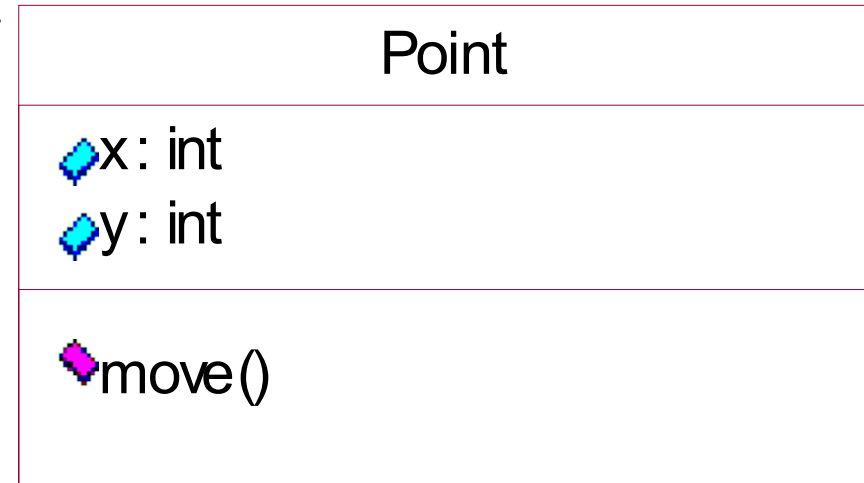
Rappresentazione in Rose



Visibilità pubblica

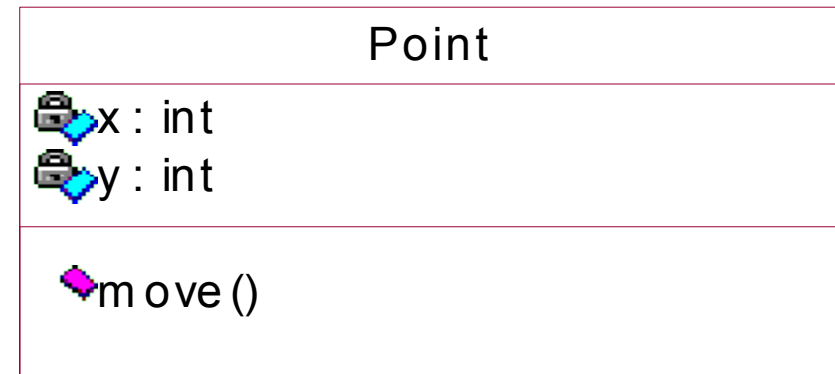
```
public class Point {  
    public int x,y;  
    public void move(int dx, int  
dy) {  
        x+= dx;  
        y+= dy;  
    }  
}
```

```
public class AnotherClass{  
  
    public void testMethod(){  
        Point p = new Point();  
        p.x = 10;           //OK  
        p.y = 11;           //OK  
        p.move(20, 15);    //OK  
    }  
}
```



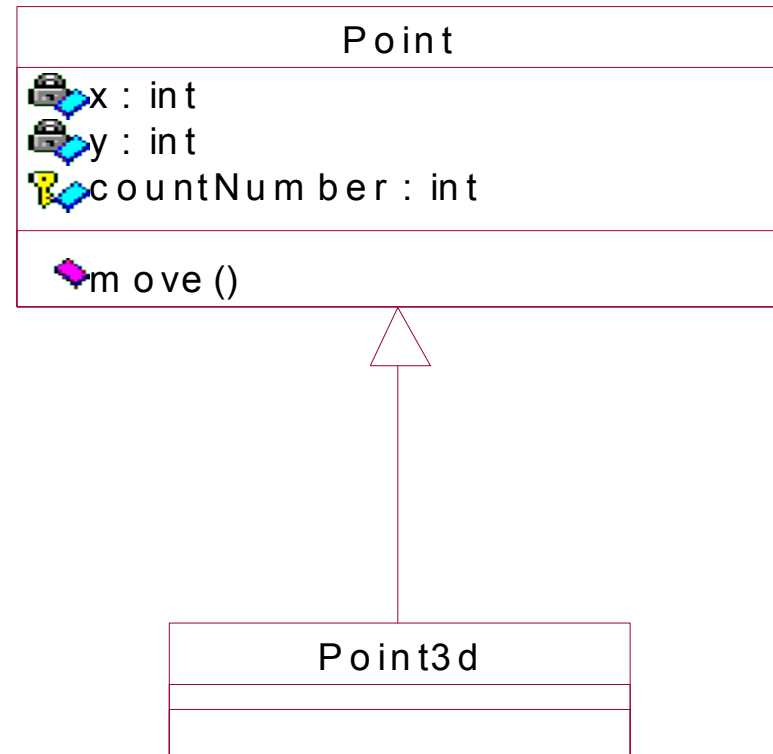
Visibilità privata

```
public class Point {  
  
    private int x,y;  
    public void move(int dx, int dy)  
    {  
        x+= dx;  
        y+= dy;  
    }  
}  
  
public class AnotherClass{  
  
    public void testMethod(){  
        Point p = new Point();  
        p.x = 10;    //ERRORE IN COMPILAZIONE  
        p.y = 11;    //ERRORE IN COMPILAZIONE  
        p.move(20, 15); //OK  
    }  
}
```



Visibilità protetta

```
public class Point {  
    private int x,y;  
    protected int CountNumber;  
  
    public void move(int dx,  
int dy) {  
        x+= dx;  
        y+= dy;  
    }  
}  
  
public class Point3d extends  
    Point {  
  
    public void testMethod(){  
  
        move(10, 15); //OK  
        CountNumber++; //OK  
    }  
}
```



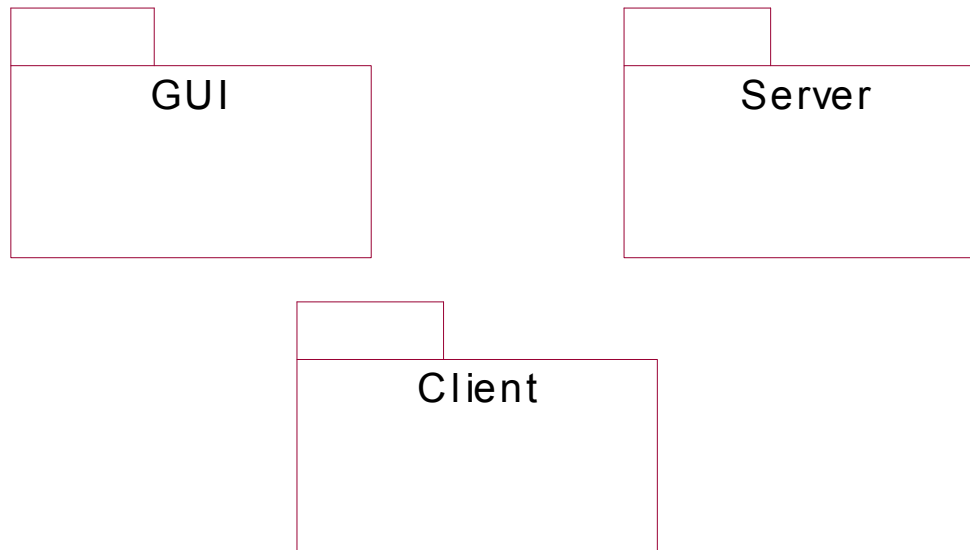
Package

- » Meccanismo general-purpose per organizzare gli **elementi in gruppi**
- » Nome può essere semplice oppure composto (se appartiene ad un altro package)
- » Può essere **annidato all'interno di un altro package**
- » Possibile controllare la visibilità degli elementi del package utilizzando gli attributi di visibilità (+, #, -, ~)

Principio O.O. Modularità



Rappresentazione UML



Relazioni



Relazioni

- » Un relazione è una **connessione** tra cose (cioè classi, interfacce, componenti, package)
- » Una relazione fornisce un **pathway** (strada) per la comunicazione fra oggetti



Relazioni

- » Dipendenza
- » Associazione binaria
 - > Aggregazione
 - > Composizione
- » Associazione n-aria
- » Generalizzazione
- » Realizzazione

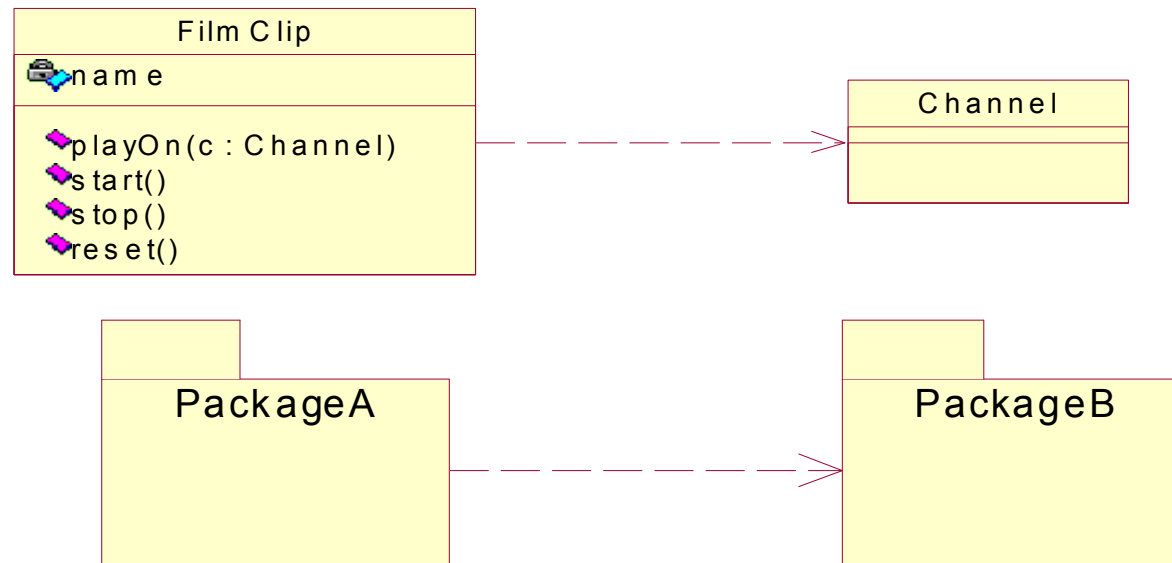


Dipendenza

- » E' una relazione tra due elementi dove **il cambiamento di un elemento può causare il cambiamento nell'altro**, ma **non necessariamente il viceversa**
- » Generalmente è rappresentata nel contesto delle classi **per mostrare che una classe utilizza un'altra classe** come argomento nella segnatura di un'operazione



Dipendenza



Associazione

- » Rappresenta una relazione strutturale tra oggetti di classi differenti
 - > Esempio: “sono connessi”, “ \forall oggetto X, esiste un Y”, “conoscono info”
- » Possibile avere associazioni circolare, cioè tra oggetti della stessa classe
- » Associazione che collega due classi è detta **binaria**; **n-aria** che collega n classi (poco utilizzata)
- » **Rappresentata mediante una linea continua che collega le due classi**



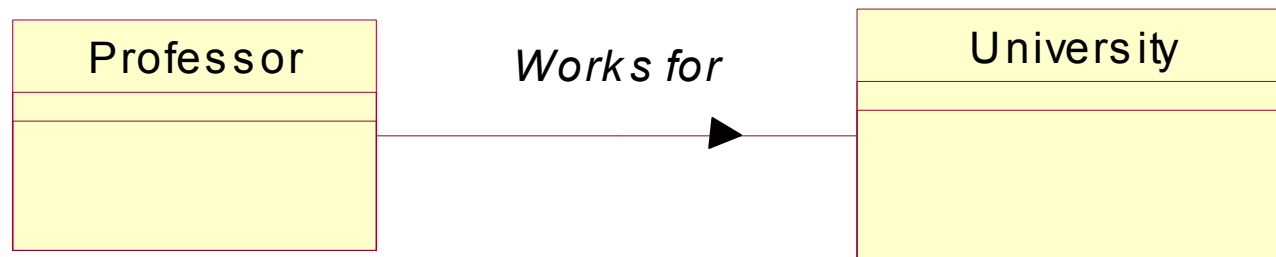
Associazione

- » Nome
- » Ruolo
- » Molteplicità
- » Navigabilità
- » Visibilità
- » Associazione xor
- » Qualifier
- » Interface Specifier
- » Association class



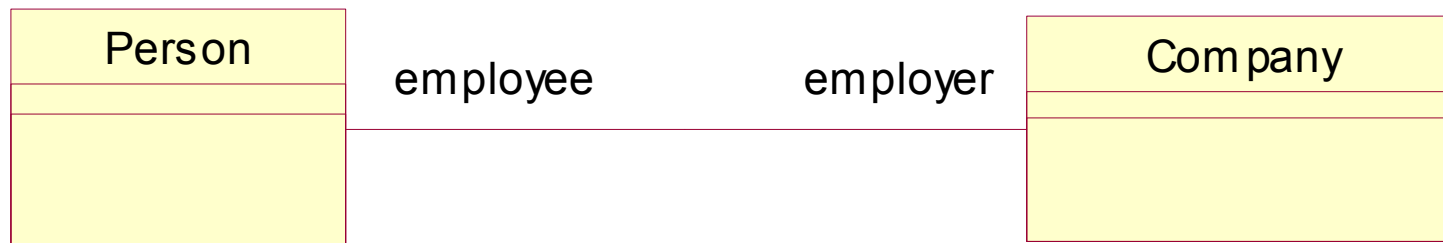
Associazione: Nome

- » Descrive la natura dell'associazione
- » E' possibile dare una direzione al nome mediante un triangolo che punta alla direzione che si intende far leggere la direzione



Associazione: Ruolo

- » Specifica **l'aspetto** che una classe gioca nell'associazione
- » Ha un **nome** ed è **posto** vicino alla classe che gioca quel ruolo nell'associazione rispetto all'altra
- » L'utilizzo del ruolo o del nome è mutuamente esclusivo



Associazione: Molteplicità

- » Definisce quanti oggetti partecipano in una relazione
 - > Specifica UNA istanza di una classe, a quante istanze dell'altra classe e' collegata
- » Applicata alla fine di ogni associazione



Associazione: Molteplicità

Valore	Descrizione
n (default)	Numero illimitato di istanze
1	Una sola istanza
0..n	Zero o più istanze
1..n	Una o più istanze
0..1	Zero o una istanza
<literal>*	Esatto numero di istanze
<literal>..n	Esatto numero o più istanze
<literal>..<literal>	Intervallo specificato di istanze
<literal>..<literal>, <literal>	Intervallo più numero specificato di istanze
<literal>..<literal>, <literal>..<literal>	Il numero di istanze sarà in uno degli intervalli specificati
* Dove <literal> è un intero maggiore o uguale a 1	



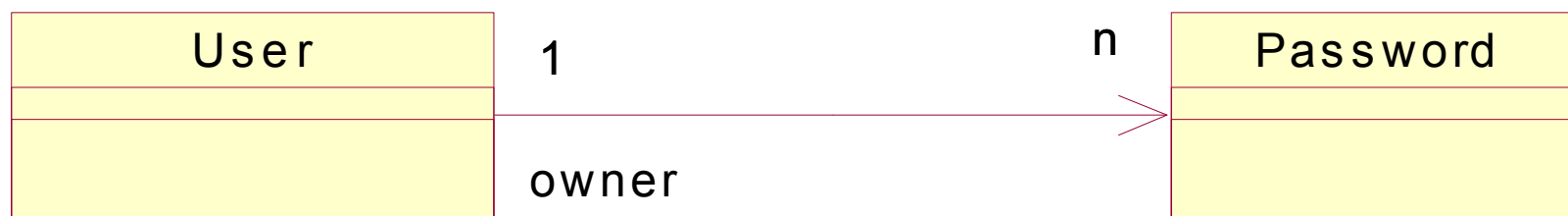
Associazione: Navigabilità

- » Nell'associazione la navigazione tra le classi è **bi-direzionale**
- » In alcuni casi è possibile limitare tale navigazione ad una direzione
- » Graficamente c'è una freccia che indica la navigazione



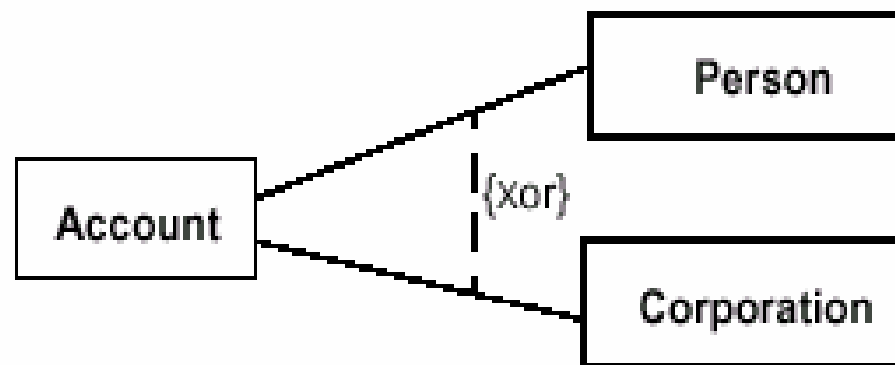
Associazione: Navigabilità

- » Dato un utente siamo in grado di sapere tutte le sue password
- » Dato una password **non** siamo in grado di risalire all'utente (è ragionevole)



Associazione Xor

- » Lega insieme più associazioni
- » Indica una situazione dove in un determinato istante di tempo per una specifica istanza della classe, può verificarsi **solo una delle potenziali associazioni** che una stessa classe può instaurare con altre



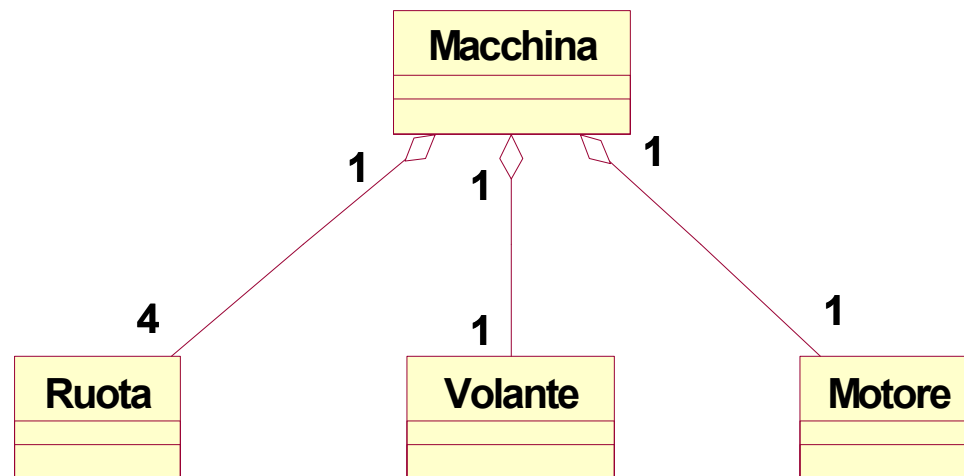
Associazione: **Aggregazione**

- » E' necessario utilizzare relazioni del tipo “**tutto-parte**” (*whole-part*), in cui esiste una classe che rappresenta il concetto “più grande” (il tutto) costituito dalle restanti che rappresentano i concetti più piccoli (le parti)
- » Tali relazioni sono dette **Aggregazione**
- » Rappresentata con una linea che connette gli oggetti in relazione utilizzando un **diamante** posto vicino alla classe completa



Associazione: Aggregazione

- » Esempio
 - > Automobile composta di Ruote, Motore, Volante, Sedili,
- » Differenza rispetto a Associazione puramente concettuale
 - > Si afferma che la classe aggregata è composta di altre classi
- » Non hanno senso aggregazioni circolari
 - > Classe A composta da una Classe B; B composta da una Classe C a sua volta composta dalla Classe A



Associazione: **Composizione**

- » Forma di aggregazione con una **forte connotazione di possesso**
- » Parti possono essere generate anche in un tempo successivo alla creazione dell'istanza della classe composta, **ma una volta generate queste vivono e sono distrutte con l'istanza della classe composta di appartenenza**
 - > se elimino la classe contenitore, elimino anche la classe aggregata

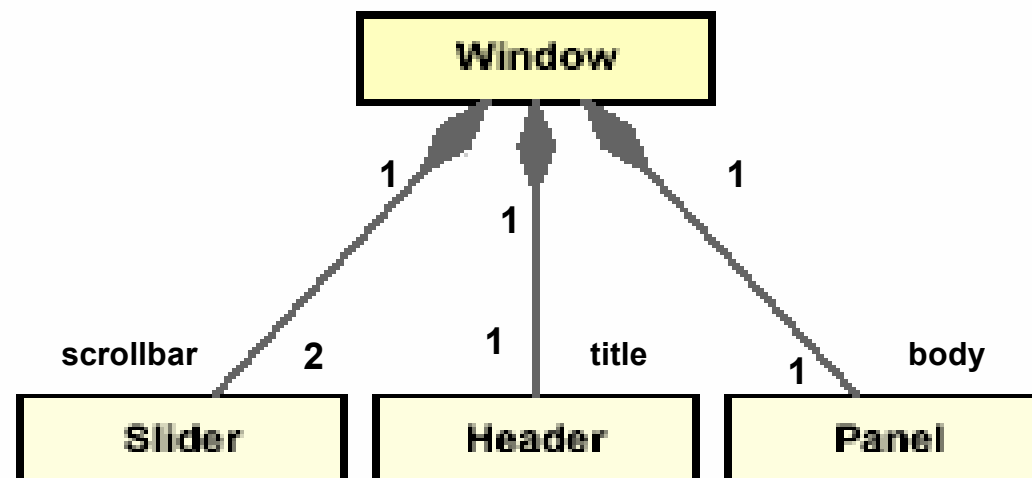


Associazione: Composizione

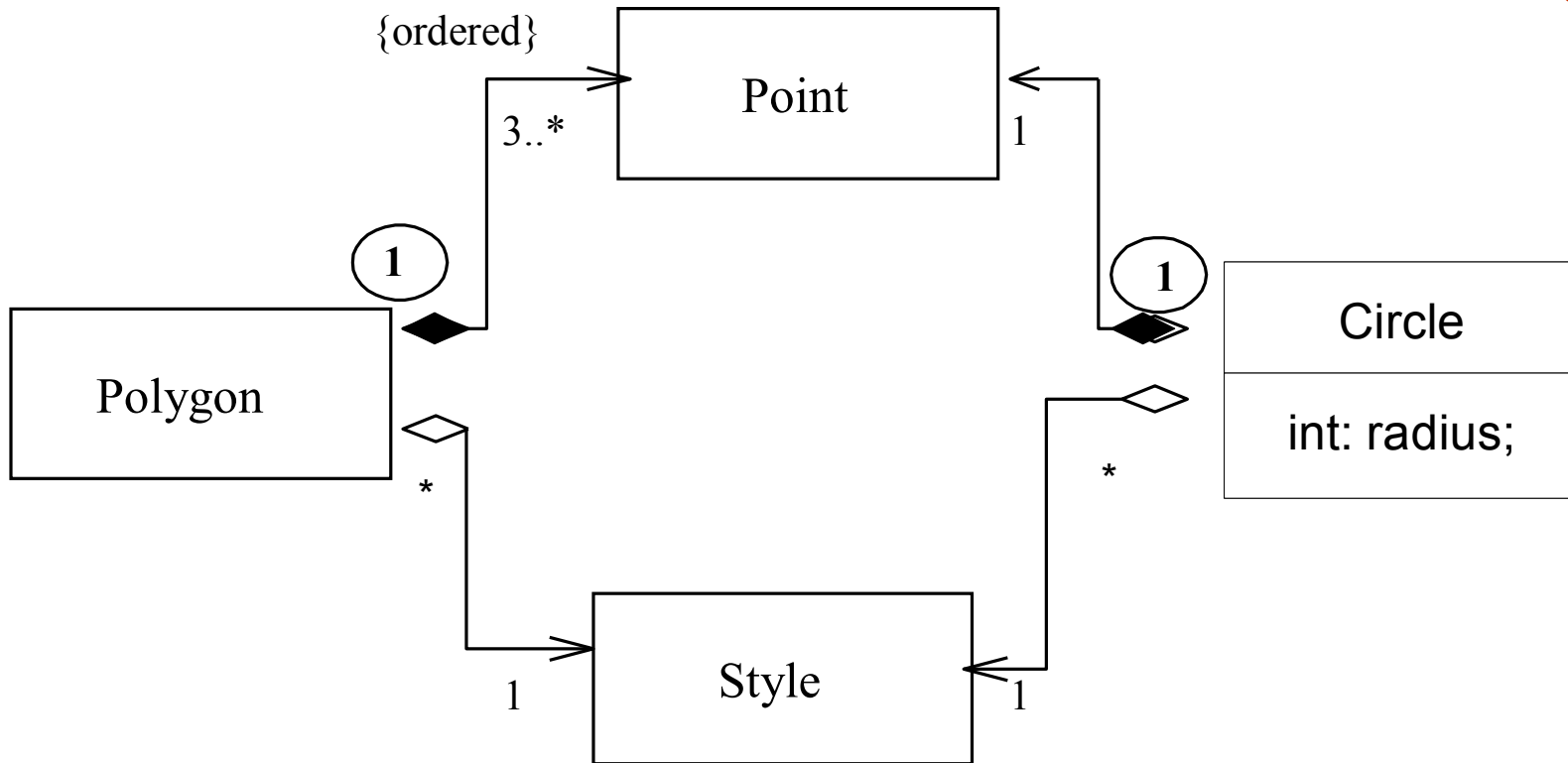
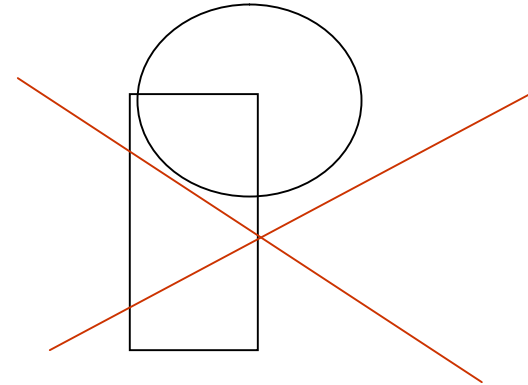
- » Classe composta si occupa di eliminare le proprie parti in un momento antecedente alla propria distruzione
- » Un oggetto può essere parte soltanto di un oggetto composto alla volta
 - > Esempio: Un *Panel* appartiene ad una sola *Window*

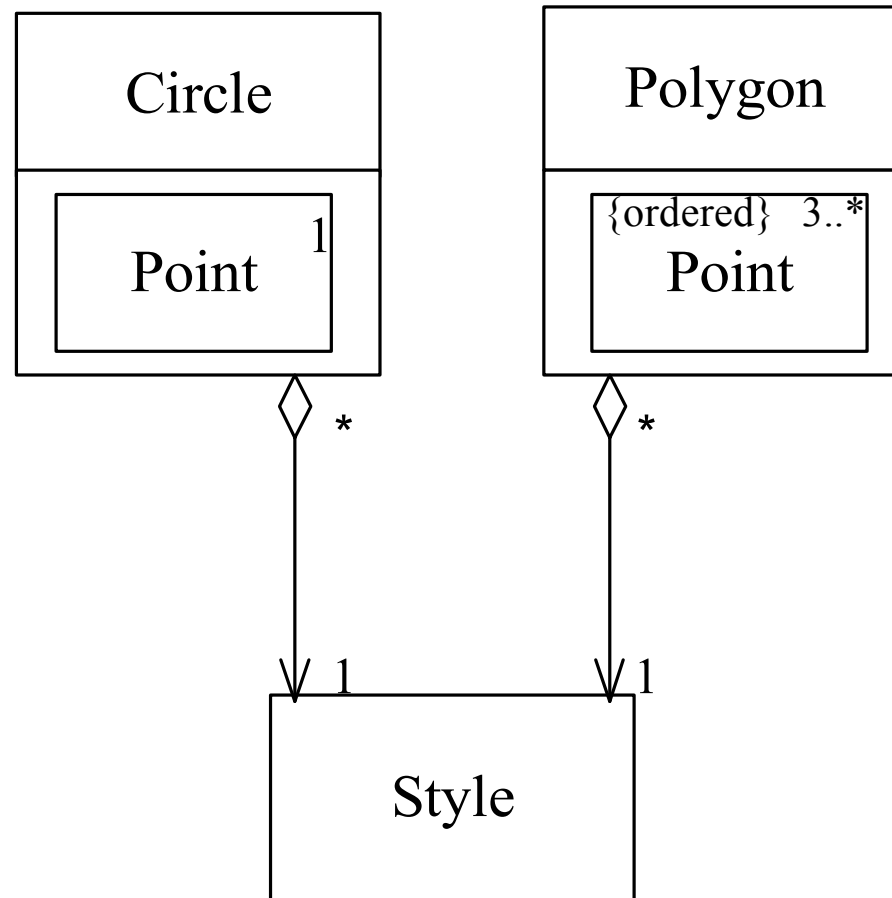
Invece in un'aggregazione una parte può essere condivisa tra più composte

- > Esempio: Un *Wall* può appartenere a più *Room*



Esempio:





Associazione vs. Dipendenza

- » **Associazione** è una relazione **strutturale** (quindi "persistente"), che evidenzia classi **semanticamente correlate**
- » **Dipendenza** ha un carattere **transitorio**, un legame che si instaura (o almeno così dovrebbe essere) **temporaneamente**, per il lasso di tempo necessario per fruire di un servizio, per creare un oggetto, ecc., per poi perdere di significato

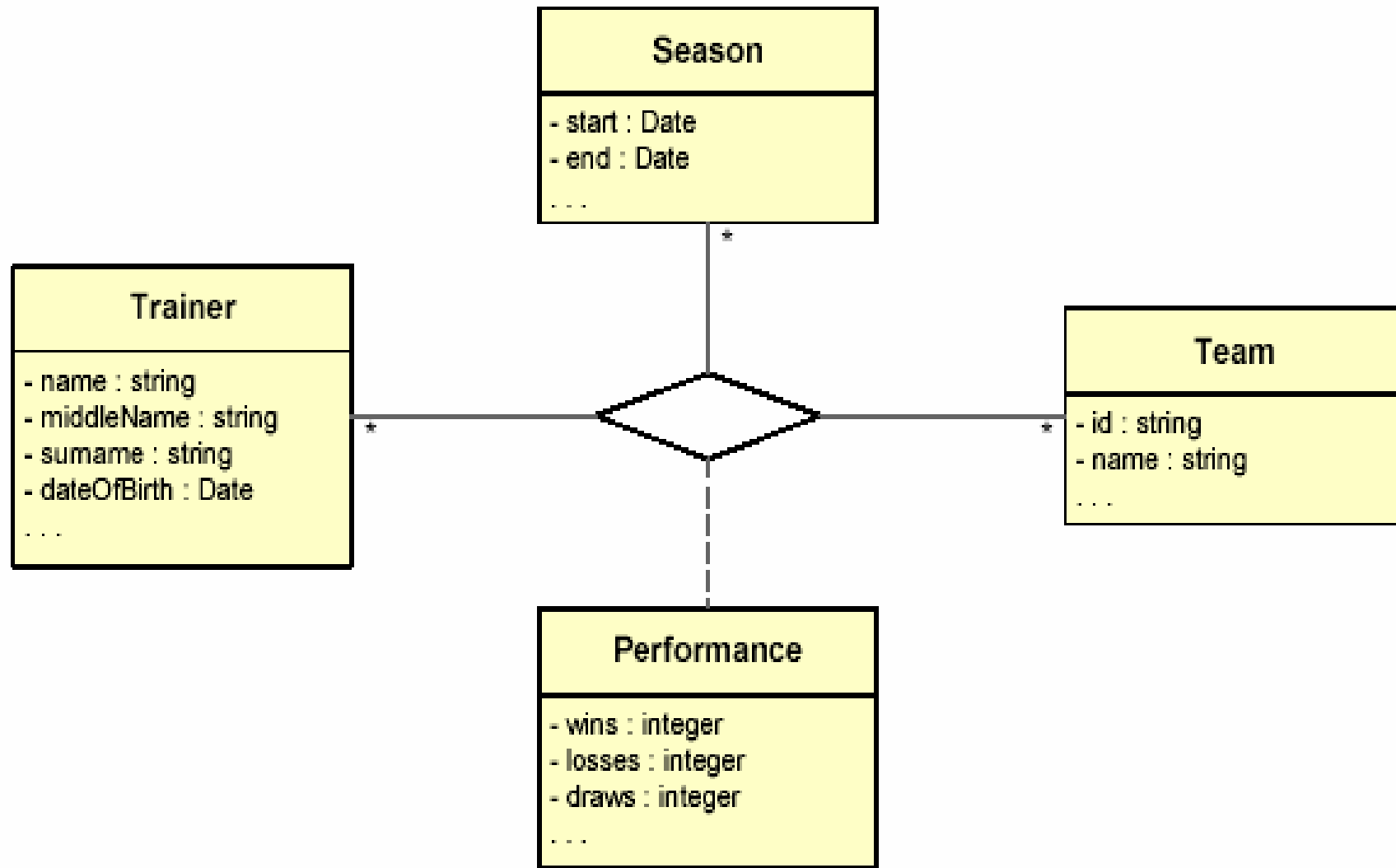


Associazione n-aria

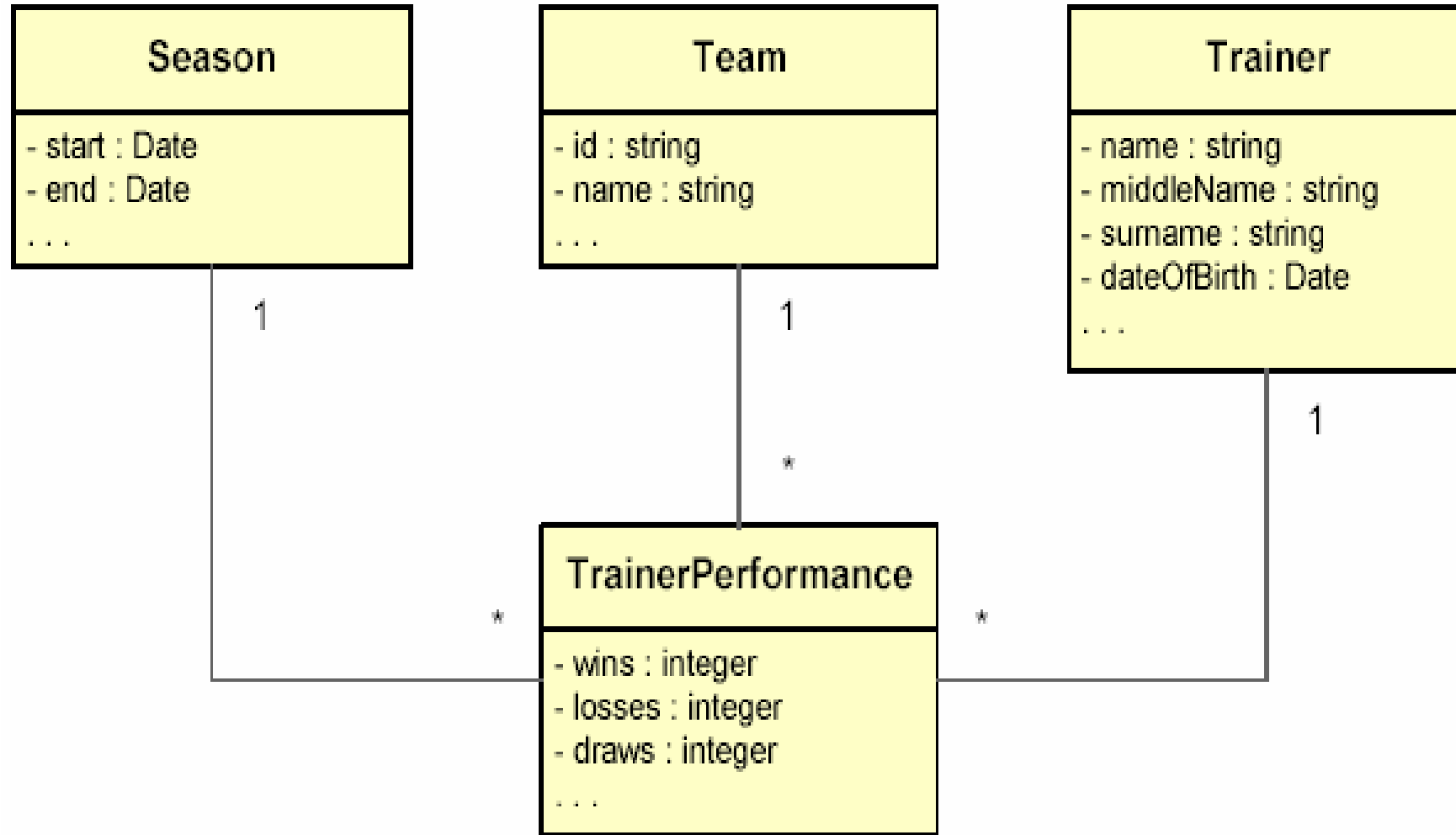
- » Relazione che coinvolge più di due classi
- » Rappresentata mediante un **rombo** dove sono collegate le classi appartenenti alla relazione
- » Difficoltà nell'attribuire i valori delle molteplicità. Specificano, per ogni classe, il potenziale numero di istanze della classe che possono partecipare nella relazione, fissando i valori delle altre $n-1$ classi



Associazione n-aria



Associazione n-aria



Trasformazione di un'associazione n-aria in associazioni binarie



Generalizzazione

- » E' una relazione tra una cosa più **generale** (detta superclasse o padre) ed una più **specifica** (detta sottoclasse o figlia)
- » Viene detta anche relazione “*is-a-kind-of*”
- » Oggetti **figlio** possono essere utilizzati al posto di oggetti **padre** (principio di sostituibilità di Liskov) ma non il viceversa, cioè il padre non è un sostituto per il figlio

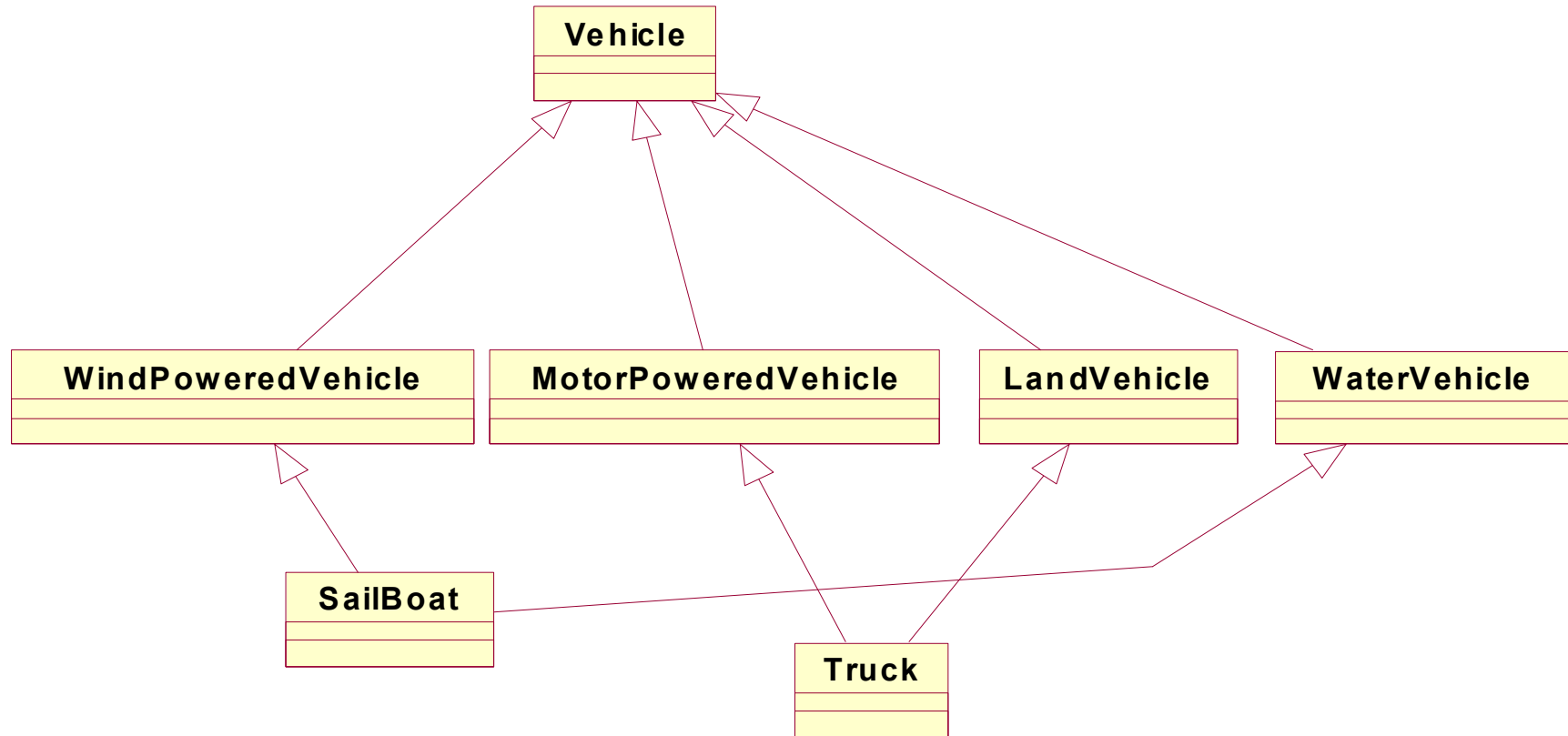


Generalizzazione

- » Il figlio eredita tutte le proprietà dei suoi padri, cioè **attributi ed operazioni**
- » Alcune volte il figlio può sovrascrivere (override) operazioni del padre
- » Relazione
 - > **Transitiva**: se C generalizza (ossia eredita) B e B eredita da A, ne segue che C generalizza anche A
 - > **Antisimmetrica**: se A eredita da B, non è assolutamente vero il contrario. Se è vero allora A e B sono uguali



Generalizzazione



Da Use Case Diagram a Class Diagram



Forward e Reverse Engineering

- » Da Use Case diagram a Class Diagram
- » Da Codice a Class Diagram



Da Use Case a Classi (Forward Engineering)

- » **Per ogni Use Case (o attore)**, bisognerebbe individuare le classi che lo implementano, e quindi realizzare un class diagram dello use case (o attore)
 - > Poi, bisognerebbe accorpare i vari class diagram realizzati per singolo Use Case (o attore)
 - > Un glossario dei termini (classi, metodi) puo' aiutare in questa fase
- » In alternativa, si puo' creare un **class diagram direttamente per tutto il sistema**
 - > Richiede forse maggior esperienza



Da Class a Use Cases (Reverse Engineering)

- » Per ogni package, si crea un package diagram
- » Per ogni classe, si crea un modello di classe
 - > Con attributi e metodi
- » Se due classi sono collegate, una relazione viene introdotta nel class diagram

- » Si vedano le classi nel folder **ReverseEngineering.zip**



Esempio di Forward Engineering



Ospite

- » *“Il sistema deve contenere una parte pubblica, visibile agli ospiti, in cui sia possibile visualizzare informazioni inerenti all’offerta formativa, nello specifico a quelli che sono gli obiettivi che il master si prefigge, descrivendo quindi i contenuti didattici dello stesso, nonché il tipo di frequenza, i destinatari dell’offerta formativa, le aree didattiche ricoperte dal master e tutti gli aspetti legati alla tassa di iscrizione.*
- » *Sempre nella parte pubblica, dovrà esserci una sezione inerente alla modulistica, contenente il bando del master, la domanda per l’ammissione allo stesso e i vari moduli di pagamento delle tasse.*



Ospite

- » *Dovrà esserci poi la possibilità di visualizzare il calendario delle lezioni ed eventualmente permettere di scaricarlo, l'elenco dei docenti che impartiranno le lezioni in un dato anno accademico, l'elenco dei corsi proposti in uno specifico anno accademico, nonché l'elenco delle aziende che sponsorizzano il master e i seminari da loro tenuti.*
- » *Infine un'ospite dovrà poter visualizzare le news e gli avvisi di carattere generale, i link ad altri master affini, la rassegna stampa degli articoli e degli spot pubblicitari, tutte le informazioni per contattare gli organizzatori del master ed eventuali faq di carattere generale e specifico.*



Ospite (con selezione)

- » “Il sistema deve contenere una parte pubblica, visibile agli ospiti, in cui sia possibile visualizzare informazioni inerenti all’offerta formativa, nello specifico a quelli che sono gli obiettivi che il master si prefigge, descrivendo quindi i contenuti didattici dello stesso, nonché il tipo di frequenza, i destinatari dell’offerta formativa, le aree didattiche ricoperte dal master e tutti gli aspetti legati alla tassa di iscrizione.
- » Sempre nella parte pubblica, dovrà esserci una sezione inerente alla modulistica, contenente il bando del master, la domanda per l’ammissione allo stesso e i vari moduli di pagamento delle tasse.



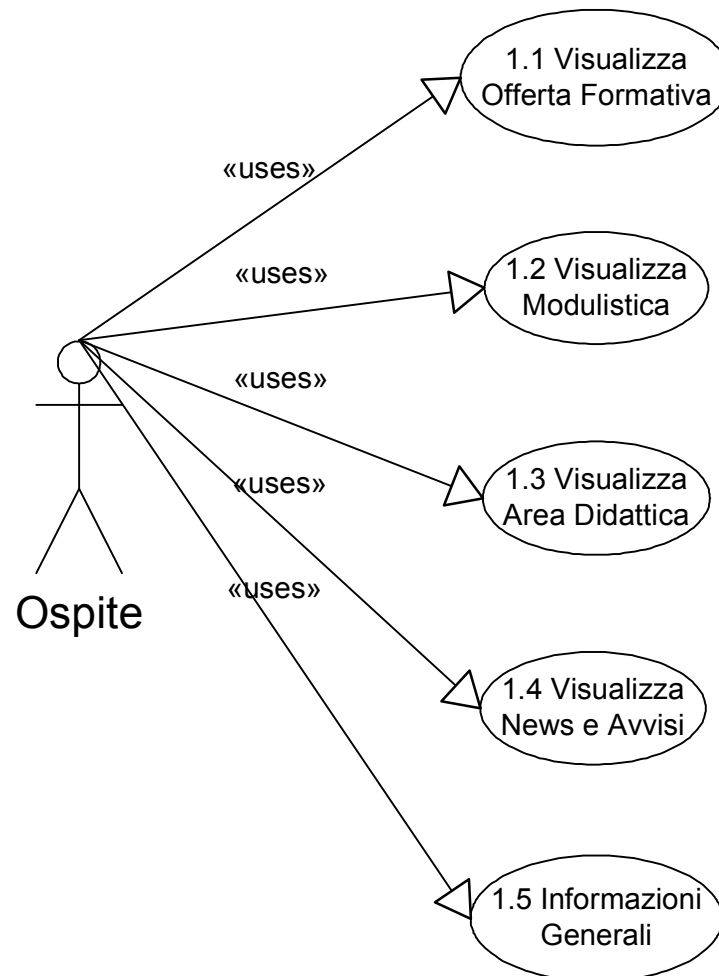
Ospite (con selezione)

- » Dovrà esserci poi la possibilità di visualizzare il calendario delle lezioni ed eventualmente permettere di scaricarlo, l'elenco dei docenti che impartiranno le lezioni in un dato anno accademico, l'elenco dei corsi proposti in uno specifico anno accademico, nonché l'elenco delle aziende che sponsorizzano il master e i seminari da loro tenuti.
- » Infine un'ospite dovrà poter visualizzare le news e gli avvisi di carattere generale, i link ad altri master affini, la rassegna stampa degli articoli e degli spot pubblicitari, tutte le informazioni per contattare gli organizzatori del master ed eventuali faq di carattere generale e specifico.



Template per Use Case

Use Case 1 – Funzionalità Ospite



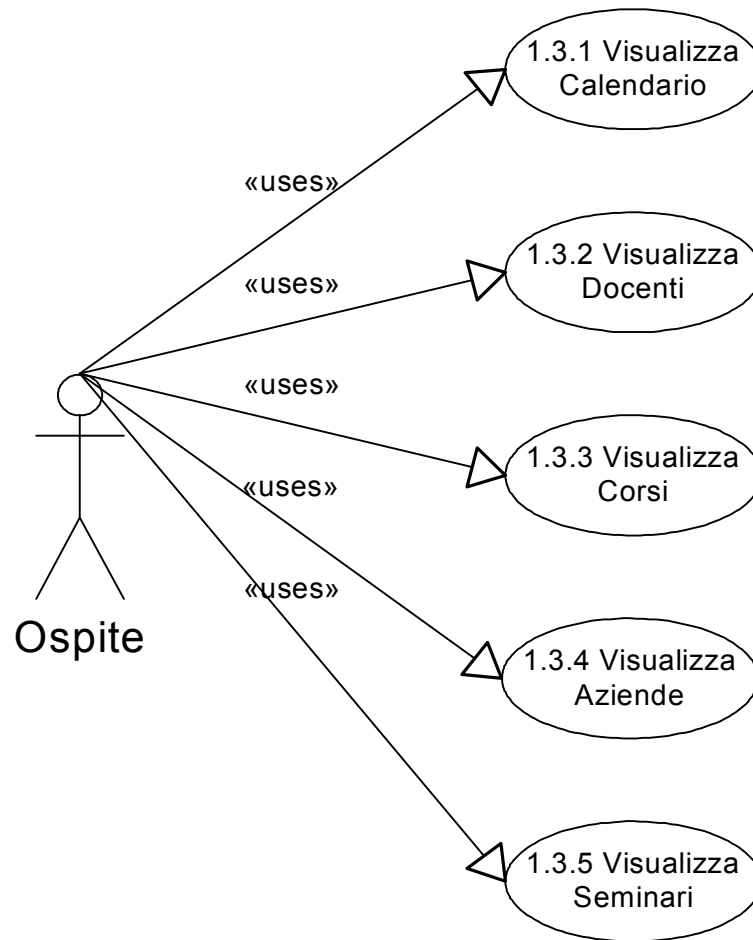
Template per Use Case

1.3	•Visualizza Area Didattica
Descrizione	Visualizza il <u>calendario</u> delle lezioni, <u>l'elenco dei docenti</u> che impartiranno le lezioni in un dato anno accademico, <u>l'elenco dei corsi</u> proposti in uno specifico anno accademico, nonché <u>l'elenco delle aziende</u> che sponsorizzano il master e i seminari da loro tenuti.
Importanza	Media.
Evento scatenante	Click dell'ospite nel menù della parte pubblica.
Uses	Include la visualizzazione di altre parti.



Template per Use Case (raffinamento)

Use Case 1.3 – Visualizza Area Didattica



Template per Use Case (raffinamento)

1.3.1	•Visualizza Calendario
Descrizione	Visualizza il <u>calendario</u> del master.
Importanza	Media.
Evento scatenante	Click dell'ospite nel menù della parte pubblica.
Uses	Include la visualizzazione di altre parti.
Requisiti non funzionali	

1.3.2	•Visualizza Docenti
Descrizione	Visualizza l' <u>elenco dei docenti</u> del master, dell'anno accademico in corso.
Importanza	Media.
Evento scatenante	Click dell'ospite nel menù della parte pubblica.
Uses	Include la visualizzazione di altre parti.
Requisiti non funzionali	



Da Use Case a Classi

- » Analizzando singolarmente gli Use Case Diagram, si individuano le classi di seguito elencate, alcune delle quali sono classi astratte, nel senso che al loro interno non contengono alcuna implementazione, ma sono semplicemente dei contenitori logici.

Nome Classe
-Sito Mwt;
-Anno Accademico;
-Offerta Formativa;
-Modulistica;
-Area Didattica;
-News e Avvisi;
-Informazioni;
-Obiettivi,
-Contenuti didattici;
-Aree Didattiche;
-Frequenza;
-Destinatari;
-Tasse;
-Bando;
-Moduli Pagamento;
-Domande;
-Calendario,
....

