Applicazioni per dispositivi mobili: Class Diagram

Romina Eramo

Università degli Studi di Teramo

Dipartimento di Scienze della Comunicazione

Email: reramo@unite.it

Sommario

» Class Diagram
- Introduzione
- Oggetto e Classe
- Elementi di una classe: Attributi ed operazioni

» Concetti avanzati
- Classi astratte
- Scope
- Visibilità

Sommario

» Relazioni
  - Dipendenza
  - Associazione binaria: Aggregazione, Composizione
  - Associazione n-aria
  - Generalizzazione
  - Realizzazione

Class Diagram

» Mostra un insieme di classi, interfacce 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)

» Modellano la parte statica di un sistema

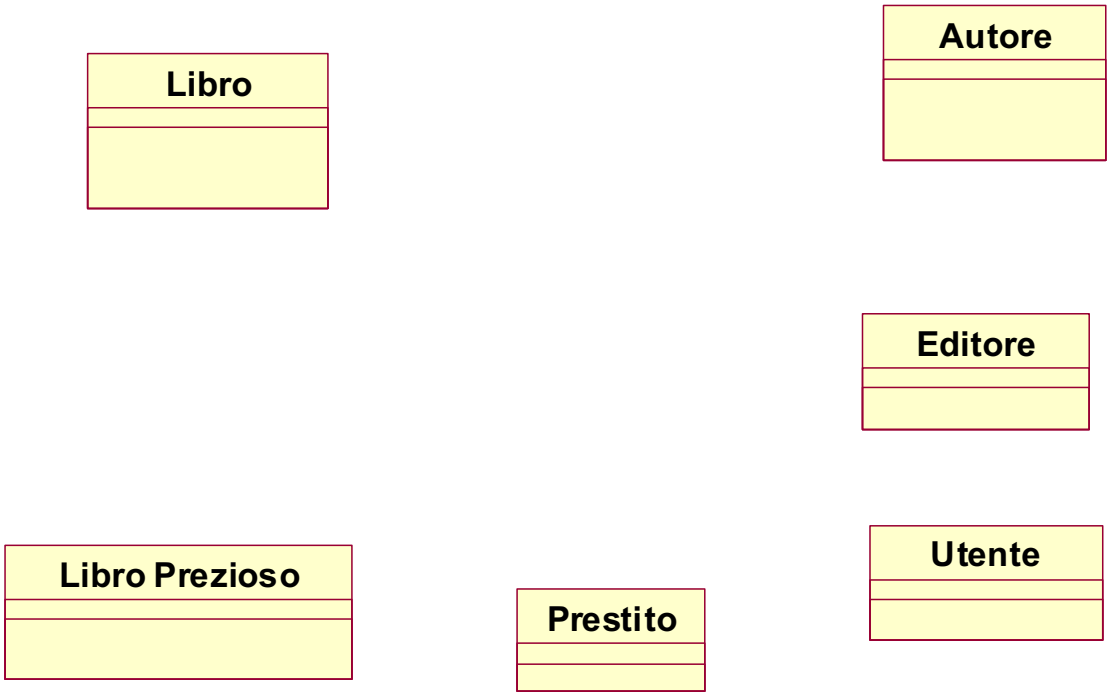
Class Diagram

» 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 somme di tutti gli elementi
» Modellare un schema logico di un database
  - Al posto degli schemi ER

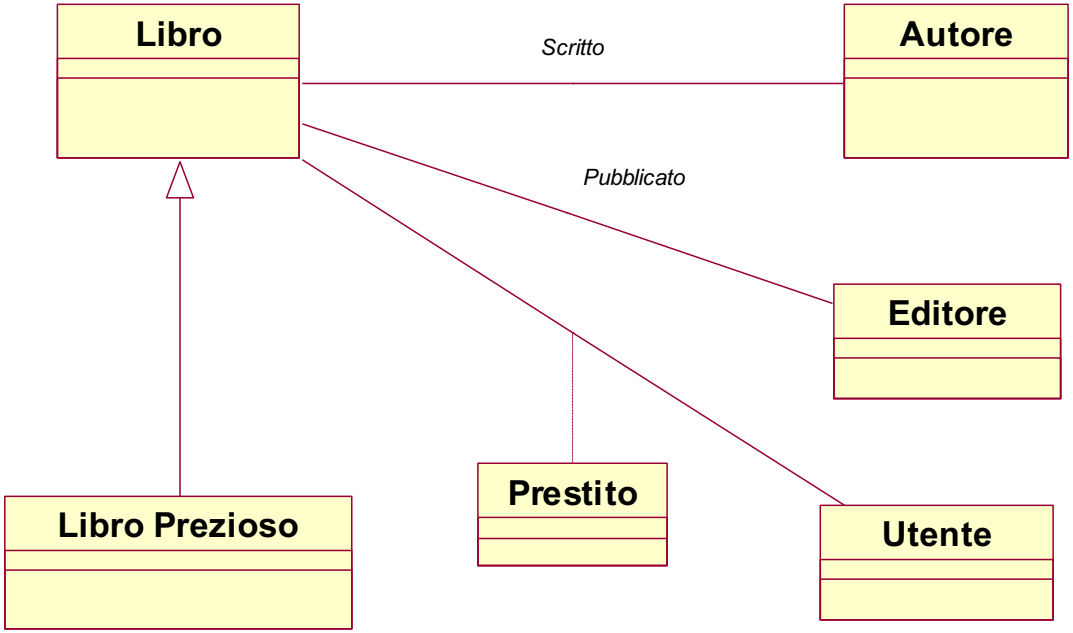
Esempio

» 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 un'operazione di ***prestito*** da parte di uno o piu' ***utenti***

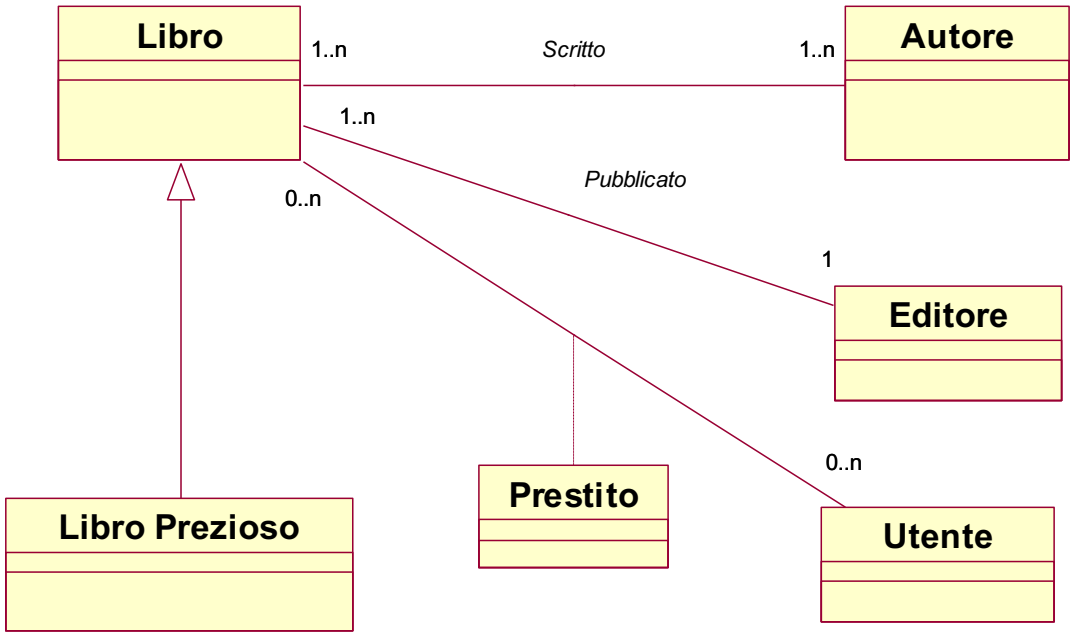
Esempio



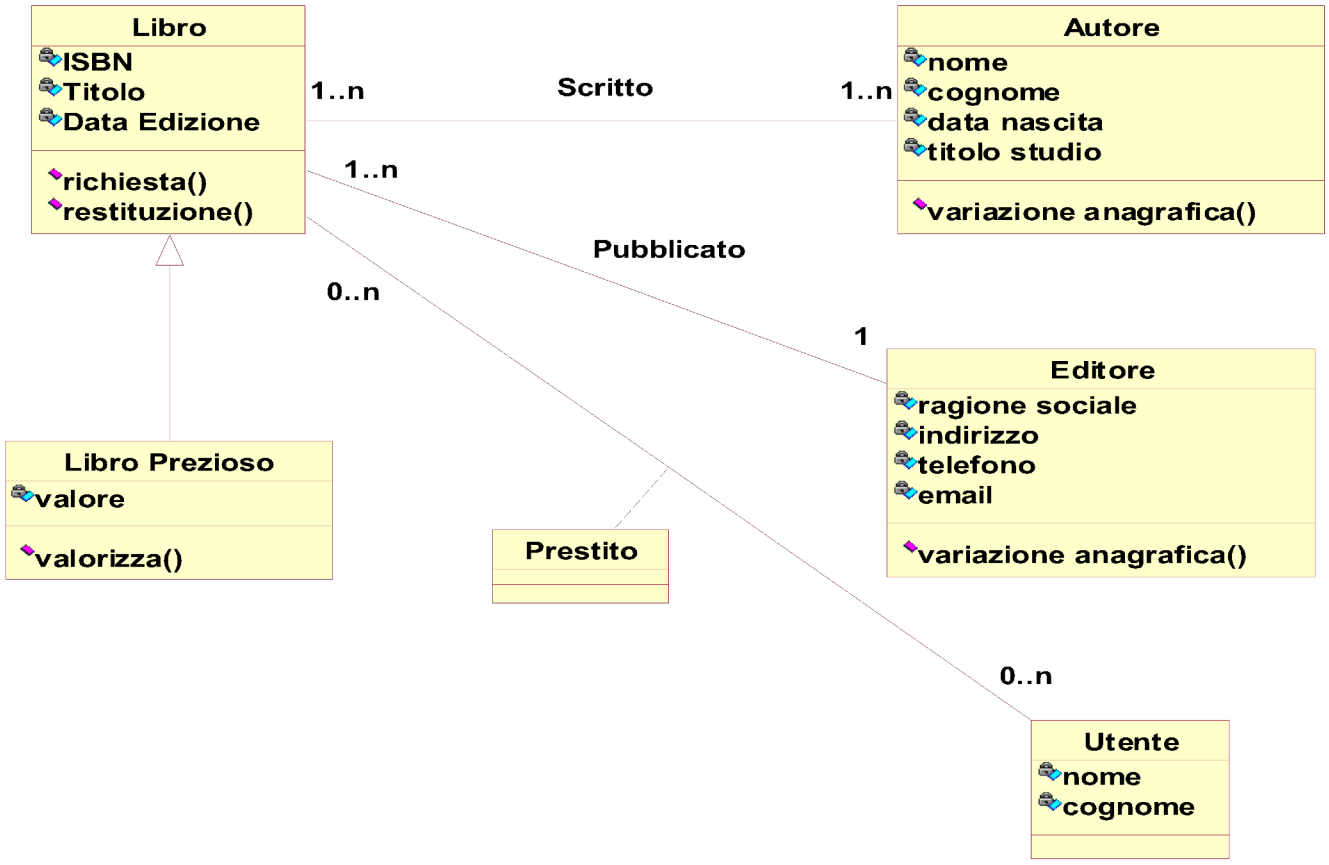
Esempio


Libro
Scritto
Autore
Pubblicato
Editore
Prestito
Utente
Libro Prezioso

Esempio


Libro
1..n
Scritto
1..n
Autore
1..n
Pubblicato
0..n
1
Editore
0..n
Prestito
Utente
Libro Prezioso

Esempio


Libro
ISBN
Titolo
Data Edizione
richiesta()
restituzione()
1..n
Scritto
1..n
Autore
nome
cognome
data nascita
titolo studio
variazione anagrafica()
1..n
Pubblicato
0..n
1
Editore
ragione sociale
indirizzo
telefono
email
variazione anagrafica()
Libro Prezioso
valore
valorizza()
Prestito
0..n
Utente
nome
cognome

Oggetto

» Informalmente un oggetto rappresenta un entità fisica, concettuale o software
  - entità fisica: trattore
  - entità concettuale: processo chimico
  - entità software: lista, coda...

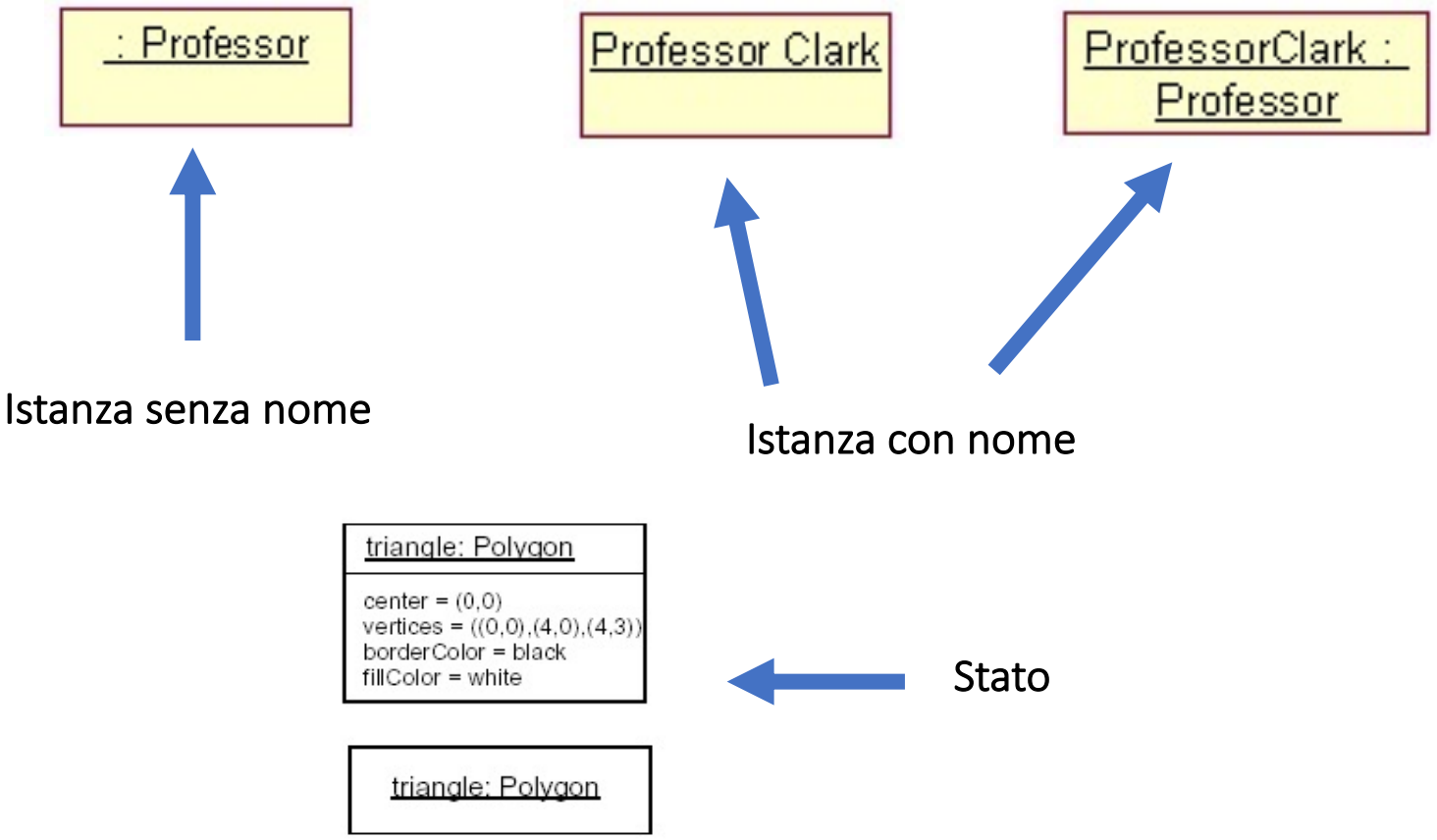
» Formalmente
  - Manifestazione concreta di un'astrazione
  - Entità con un confine e un'*identità* ben definite che incapsula *stato* e *comportamento*
  - Istanza di una classe

» Es.: Ferrari di Schumacher, Ferrari di Barrichello, Mio computer

Oggetto

- Stato
  - Possibile condizione nel quale l'oggetto potrebbe esistere e generalmente cambia nel tempo
  - Implementato mediante proprietà (attributi) con valori, e collegamenti ad altri oggetti
- Comportamento
  - Determina come un oggetto agisce e reagisce alle richieste di un altro oggetto
  - Rappresentato dall'insieme di messaggi a cui può rispondere (operazioni)
- Identità
  - Rende possibile la distinzione tra due oggetti anche se hanno lo stesso stato e lo stesso valore nei suoi attributi

Rappresentazione in UML



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

Esempio di classe

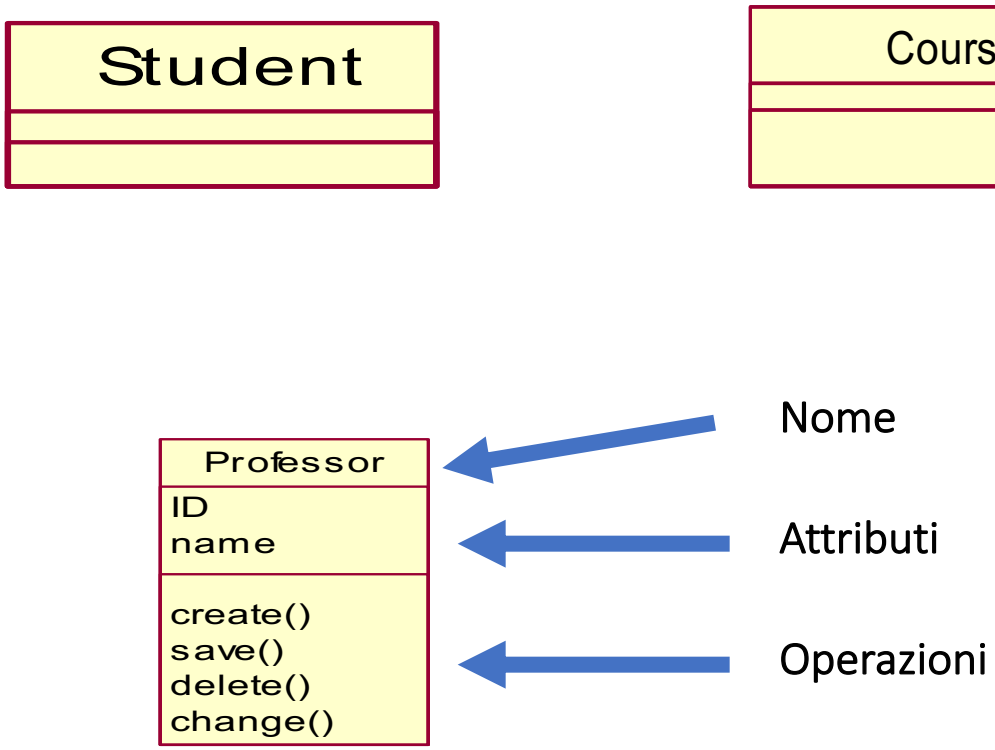
- Nome
  - Corso
- Proprietà
  - Nome, Luogo, Durata, Crediti, Inizio, Fine
- Comportamento
  - Aggiunta studente
  - Cancellazione studente
  - Verifica se è pieno

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: Attributi

» Proprietà di una classe che descrive un insieme di valori che le istanze degli attributi possono assumere

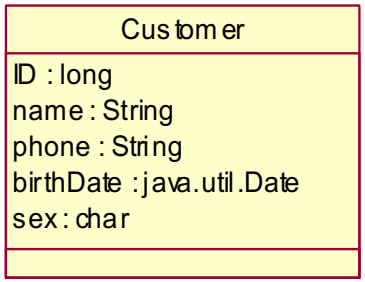
» Rappresenta proprietà delle cose che si stanno modellando che è condivisa da tutti gli oggetti di quella classe

» 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, String
- Valore iniziale
- Convenzione
  - Lettera iniziale minuscola



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

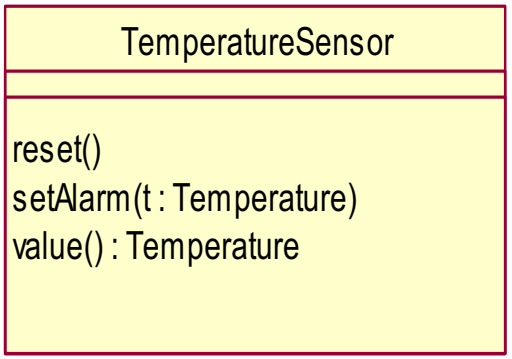
» Generalmente l'invocazione di un'operazione cambia lo stato dell'oggetto

» 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



Classi astratte

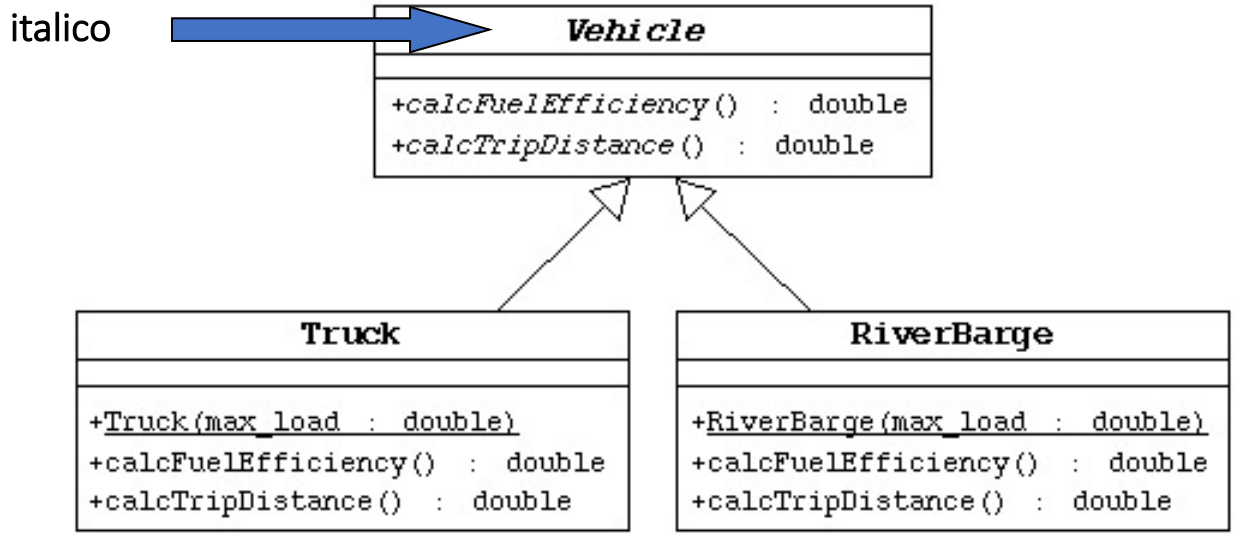
» Classe con operazioni il cui corpo non è definito

» Si può dichiarare una classe astratta anche se non ha operazioni astratte

» Non può essere istanziata, cioè non possono esistere istanze di quella classe

» Può contenere operazioni che hanno implementazione

Rappresentazione in UML



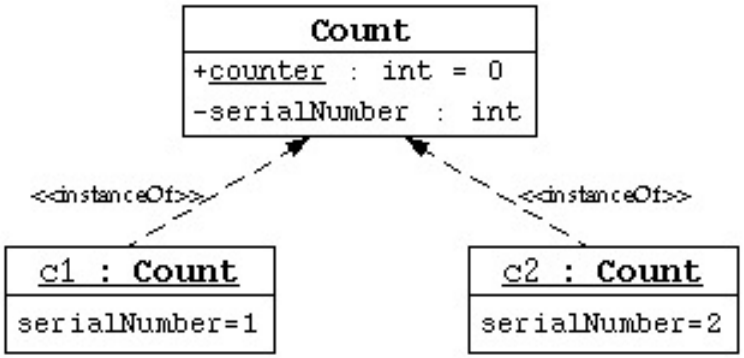
Scope

» Può essere associato sia agli attributi che alle operazioni

» Due tipi

- **instance**: ogni oggetto della classe ha il proprio valore (default)
- **classifier**: tutti gli oggetti della classe hanno lo stesso valore (UML è sottolineato)

Rappresentazione in UML



Java: Scope Attributi

```
public class Count {
  int serialNumber;
  static int counter = 0;
}
public class OtherClass {
   public void incrementNumber() {
     Count.counter++;
   }

   public void setNumber() {
       ….
       c1.serialNumber = 1;
       c2.serialNumber = 2;
   }
}
```

E’ come se fossero variabili globali

Java: Scope Operazioni

```
public class Count {
    int serialNumber;
    static int counter = 0;

    public static int getTotalCount() {
       return counter++;
    }
}

public class TestCounter {
    public static void main(String[] args) {
      System.out.println("Number of counter is " + Count.getTotalCount());
      System.out.println("Number of counter is " + Count.getTotalCount());
    }
}
```

E' come se fossero "funzioni" globali

Visibilità

- Possono essere applicati alle classi, attributi e operazioni
- Quattro 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 ~)

Relazioni

» Una relazione è una connessione tra cose (cioè classi, interfacce, componenti, package)

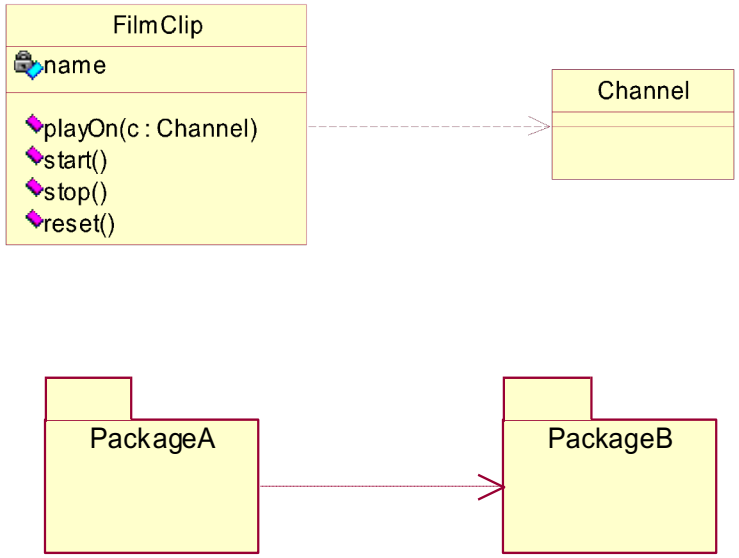
» Una relazione fornisce un cammino (pathway) per la comunicazione fra oggetti

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


FilmClip
name
playOn(c : Channel)
start()
stop()
reset()
Channel
PackageA
PackageB

Associazione

» Rappresenta una relazione strutturale tra oggetti di classi differenti

» Connessione bi-direzionale tra classi, si può navigare da un oggetto di una classe all'altro e viceversa

» Possibile avere associazioni circolare, cioè tra oggetti della stessa classe

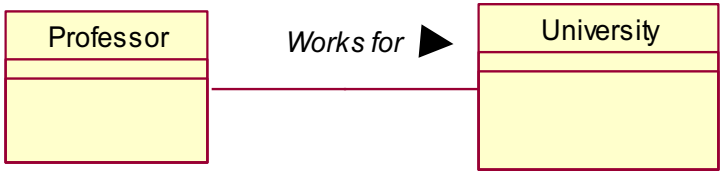
» Associazione che collega due classi è detta binaria; n-aria che collega n classi (poco utilizzata)

» Rappresentata mediante una linea continua che collega le due classi

Associazione: Nome

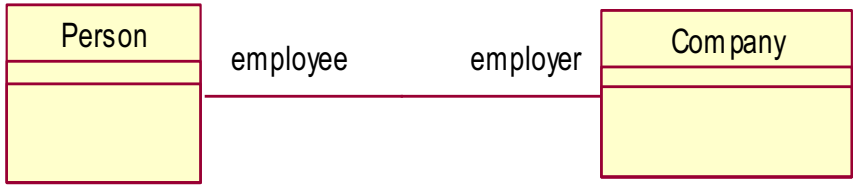
» Descrive la natura dell'associazione

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



Associazione: Ruolo

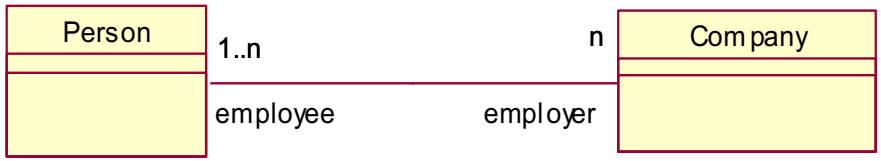
- Specifica l'aspetto che una classe gioca nell'associazione
- Ha un nome ed è posto vicino alla classe che gioca quel ruolo nell'associazione rispetto all'altra
- L'utilizzo del ruolo o del nome è mutuamente esclusivo



Associazione: Molteplicità

» Definisce quanti oggetti partecipano in una relazione
  - Specifica il numero di istanze di una classe che è in relazione con UNA istanza dell'altra classe

» Applicata alla fine di ogni associazione



Associazione: Molteplicità

```
public class Employee  {
   public Company employer[];
}

public class Company {
   public Employee employee[];
}
```

Associazione: Molteplicità

Valore	Descrizione
N	Numero illimitato di istanze
1 (default)	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: Molteplicità

» Se Molteplicità maggiore di 1 allora l'insieme degli elementi associati puo' essere o meno ordinato

» Specificato mediante vincolo alla fine dell'associazione

- **unordered**: elementi formano un insieme non ordinato (default)
- **ordered**: elementi hanno un ordinamento e i duplicati non sono ammessi ({ordered})

» Relazione ordinata viene specificata generando codice dipendente dal linguaggio di implementazione

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)


```
public class User {
   public Password thePasswords[];

}


public class Password {

}
```

Associazione: Navigabilità

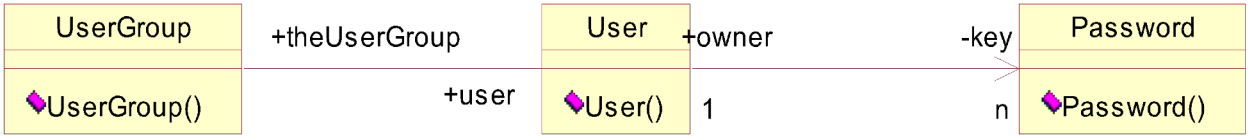
```
public class Password {
   public User owner;

}

public class User {
   public Password thePasswords[];

}
```

Associazione: Visibilità

- In un'associazione è possibile vedere e navigare da un oggetto all'altro a meno che non ci sono restrizioni (navigabilità)
- E' possibile limitare la visibilità di tale navigazione
- Tre livelli di visibilità (+, -, #) collegati al ruolo dell'associazione
- Esempio: Password è privata di un Utente

Associazione: Visibilità

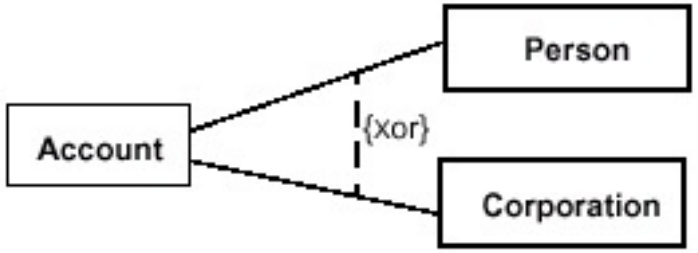
```
public class UserGroup {
   public User user;

}
public class Password {
}
public class User {
   private Password key[];
   public UserGroup theUserGroup;
}
```



Associazione Xor

» Lega insieme più associazioni

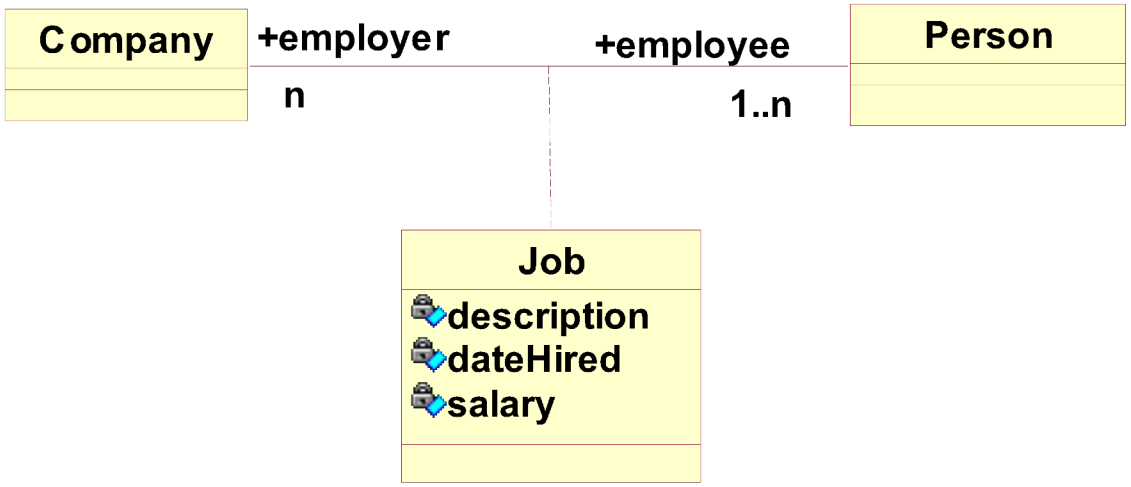
» Indica una situazione dove in un determinato istante di tempo per una specifica istanza della classe, può verificarsi solo una delle potenziali associazioni che una stessa classe può instaurare con altre



Associazione: Association Class

» Una associazione puo' possedere oltre alla molteplicità, ruoli e visibilità anche delle proprietà strutturali e comportamentali

» Esempio



Associazione: Aggregazione

» Associazione tra classi mostra una relazione strutturale paritetica (*peer-to-peer*)
  - Non è possibile distinguere una classe concettualmente più importante delle altre (sono tutte allo stesso livello)

» Esiste necessità di modellare situazioni in cui una classe esprime una nozione concettualmente più grande delle altre che la costituiscono

» Rappresenta una relazione *has-a*, *consists -of*, *contains*, *is-part-of*

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

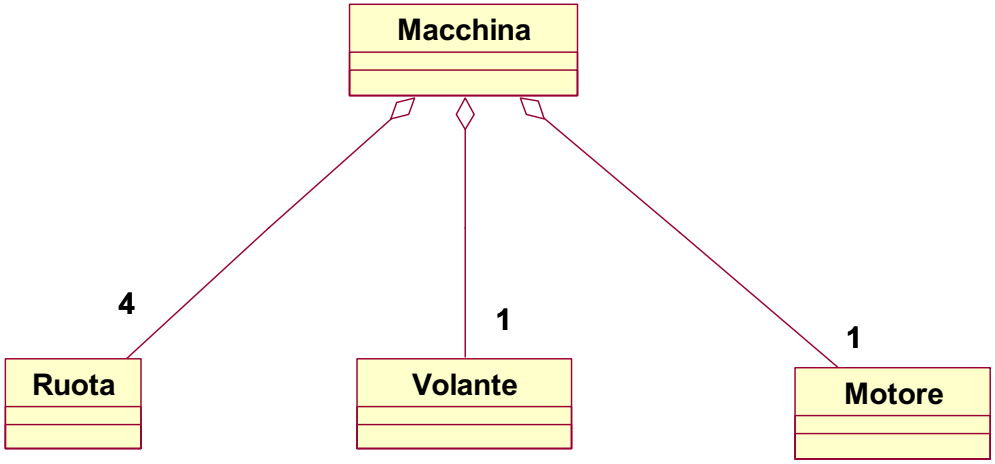
Associazione: Aggregazione

» Esempio
  - Automobile composta di Ruote, Motore, Volante, Sedili, ...

» Differenza rispetto ad Associazione puramente concettuale

» 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: Aggregazione



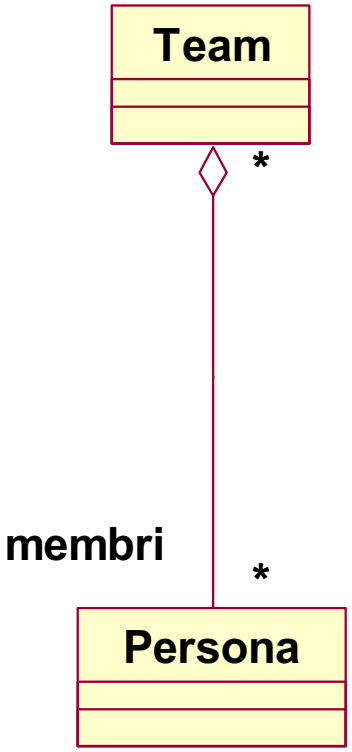
Associazione: Aggregazione Shared

» Caso speciale della normale aggregazione

» La classe parte può essere parte di qualsiasi intero

» E’ shared se la molteplicità nella parte intera è maggiore di uno

Associazione: Aggregazione Shared



Associazione: Composizione

» Forma di aggregazione con una forte connotazione di possesso e una (quasi) coincidenza del ciclo di vita tra istanze delle classi “parte” e l’istanza della classe “tutto” (classe composta)

» 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

Associazione: Composizione

» Classe composta si occupa di eliminare le istanze proprie parti in un momento antecedente alla propria distruzione

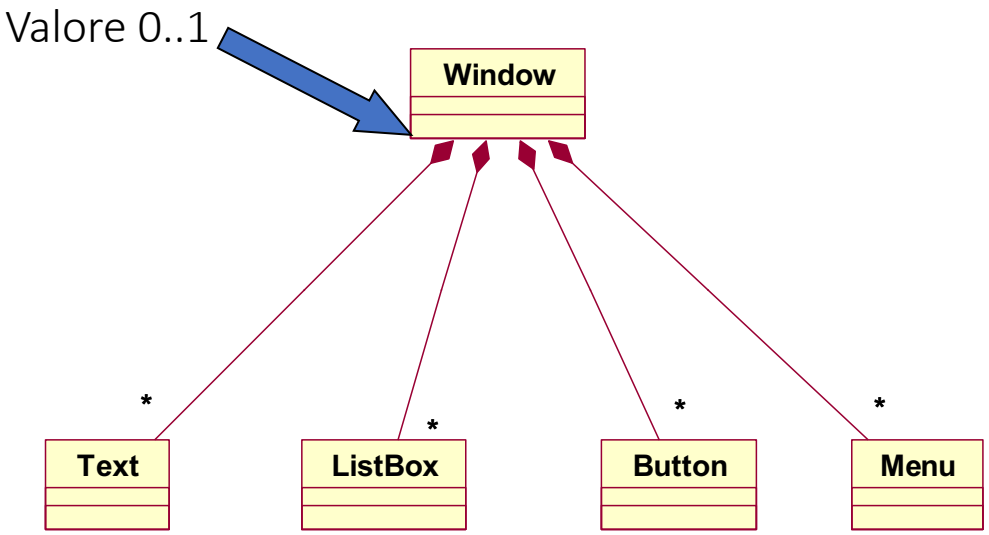
» Un oggetto può essere parte soltanto di un oggetto composto alla volta

- Esempio: Un *Frame* 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*

Associazione: Composizione



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

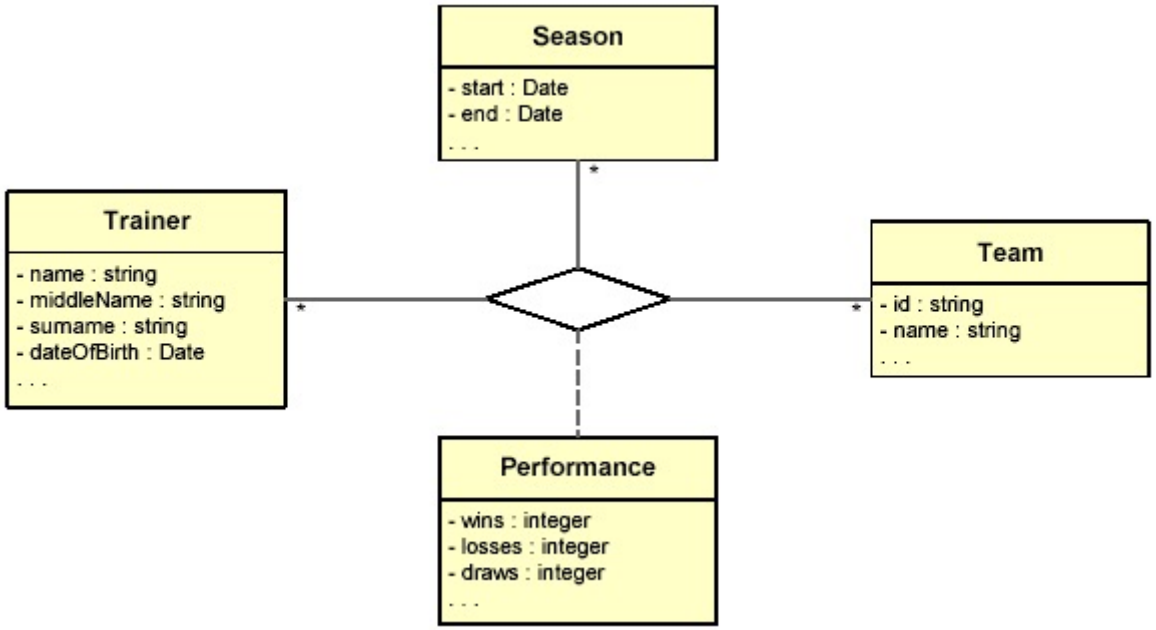
Associazione n-aria

» Relazione che coinvolge più di due classi

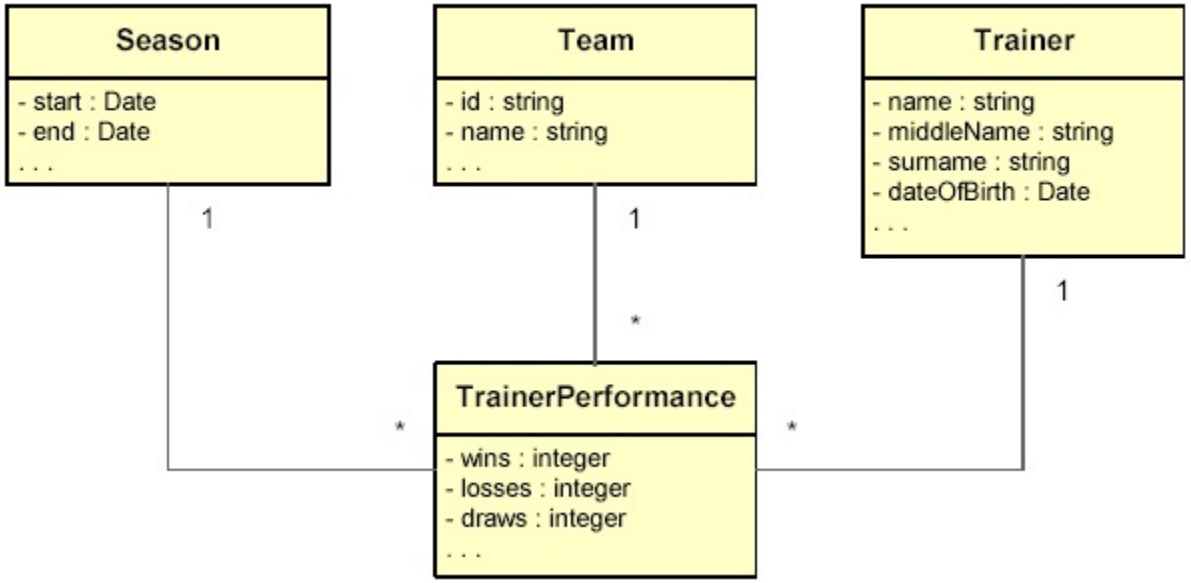
» Rappresentata mediante un rombo dove sono collegate le classi appartenenti alla relazione

» 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

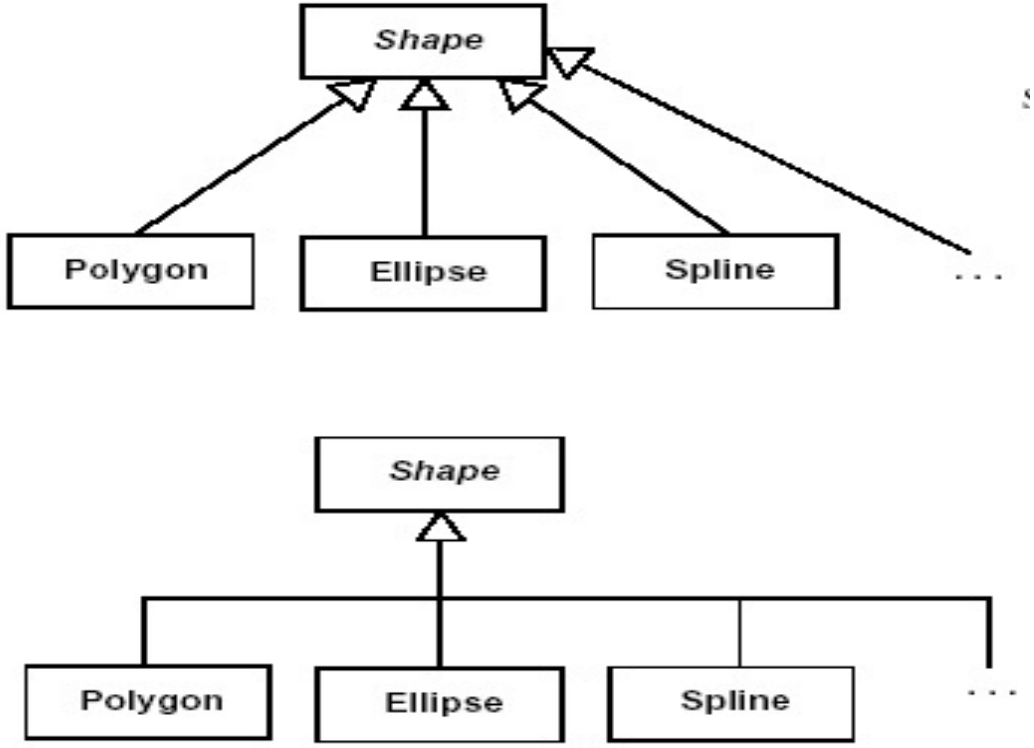
» Relazione

- 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

- Disegnata con una freccia chiusa diretta verso il padre
- Tipi di generalizzazione
  - Singola (Java)
    - Si ha un solo padre
  - Multipla (C++ e NO Java)
    - E' possibile avere più padri

Generalizzazione



Generalizzazione