UNIVERSITÀ DEGLI STUDI DI TERAMO

UML Class Diagram

Prof.ssa Romina Eramo
Università degli Studi di Teramo
Dipartimento di Scienze della Comunicazione
reramo@unite.it

Class Diagram

» Mostra un insieme di classi, interfacce e le loro relazioni (dipendenza, associazione e generalizzazione)

» Può essere visto come un grafico in cui i nodi sono le classi e le interfacce e i bordi sono le relazioni

» Possono anche contenere pacchetti o sottosistemi (utilizzati per raggruppare elementi)

» Modellano la parte statica di un sistema

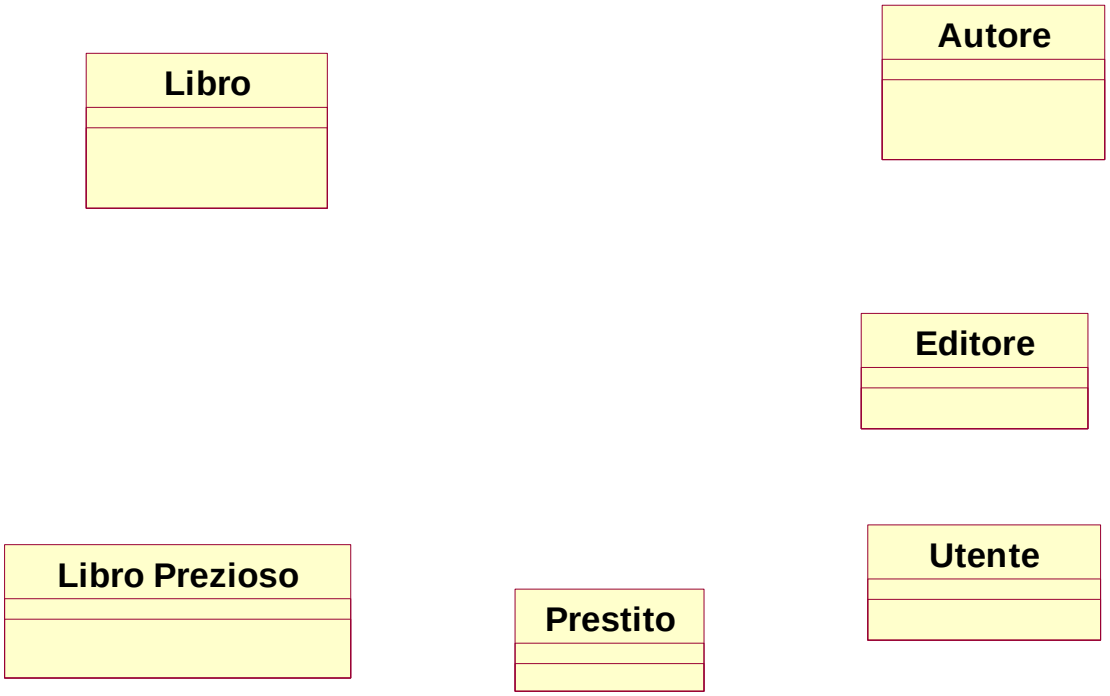
Class Diagram

» Modellare il vocabolario di un sistema
  - Le classi modellano le astrazioni delle cose di un problema
  - Tali astrazioni fanno parte del vocabolario di un sistema
» Modella collaborazioni semplici
  - Le classi non vivono sole
  - Lavorano insieme per fornire un comportamento che è maggiore della somma di tutti gli elementi
» Modellare uno schema logico del database
  - Invece di schemi ER

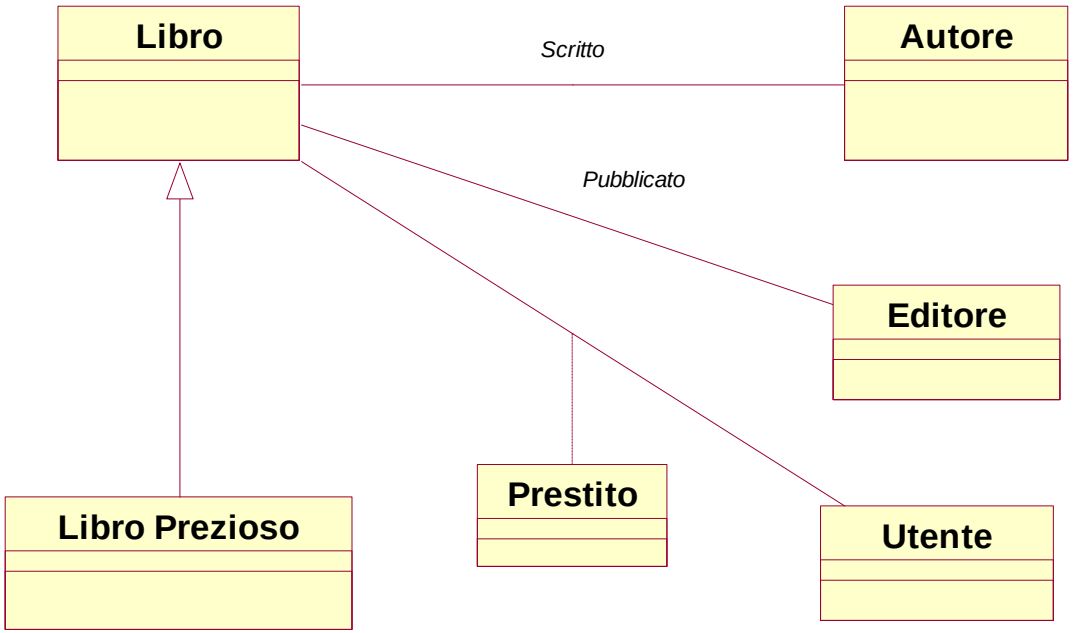
Esempio

» Sistema per l'archiviazione bibliografica dei ***testi , che*** memorizza le informazioni sull'editore e ***sull'autore*** e identifica quelli che sono considerati ***libri prezioso*** . Vuoi anche gestire un'operazione ***di prestito*** da parte di uno o più ***utenti***

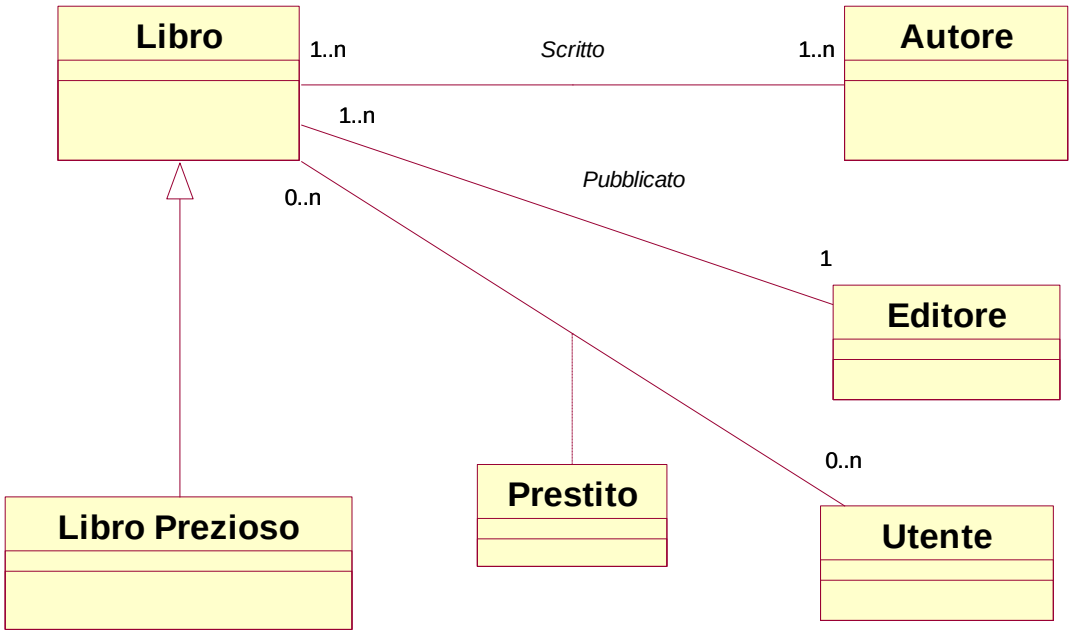
Esempio



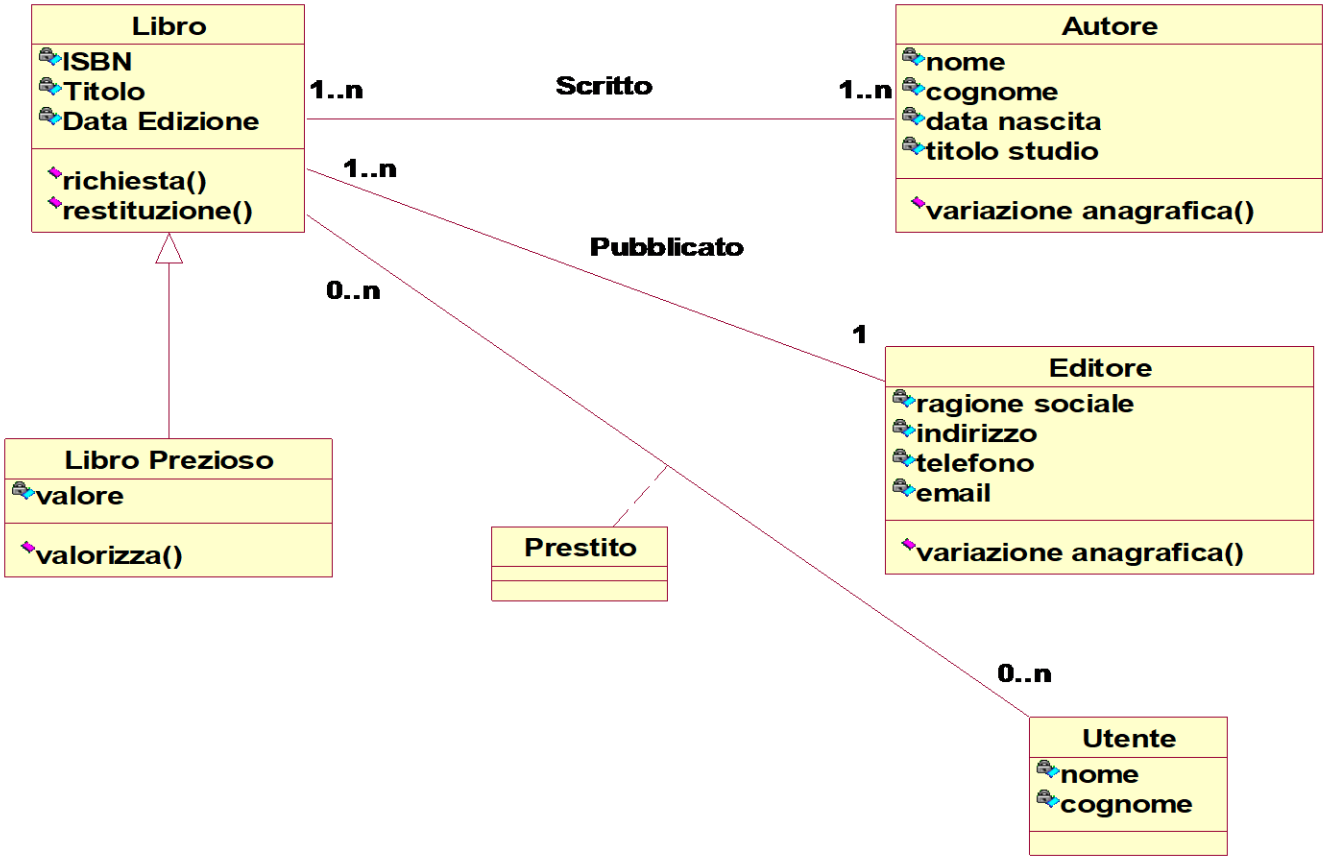
Esempio


Libro
Scritto
Autore
Pubblicato
Editore
Prestito
Utente
Libro Prezioso

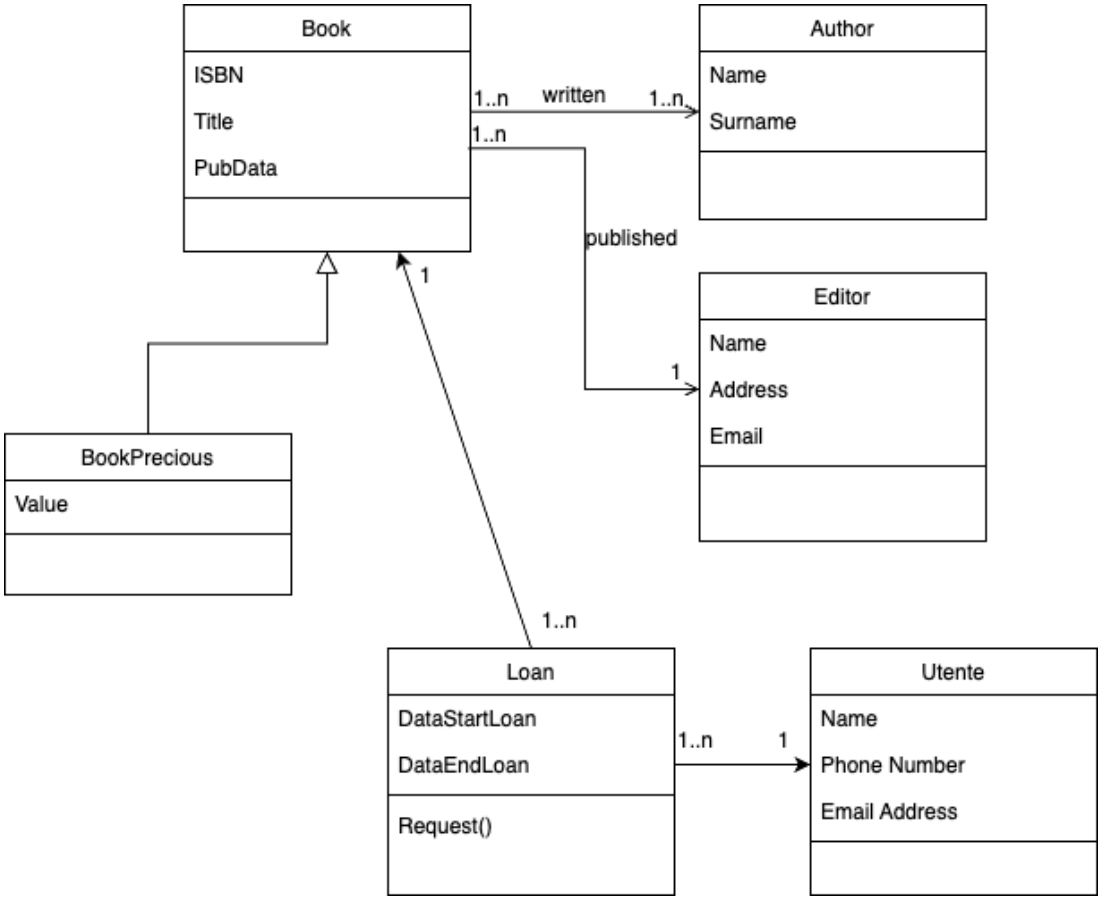
Esempio


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

Esempio


Book
ISBN
Title
PubData
Author
Name
Surname
1..n
written
1..n
1..n
published
1
Editor
Name
Address
Email
1
BookPrecious
Value
1..n
Loan
DataStartLoan
DataEndLoan
Request()
1..n
1
Utente
Name
Phone Number
Email Address

Oggetto

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

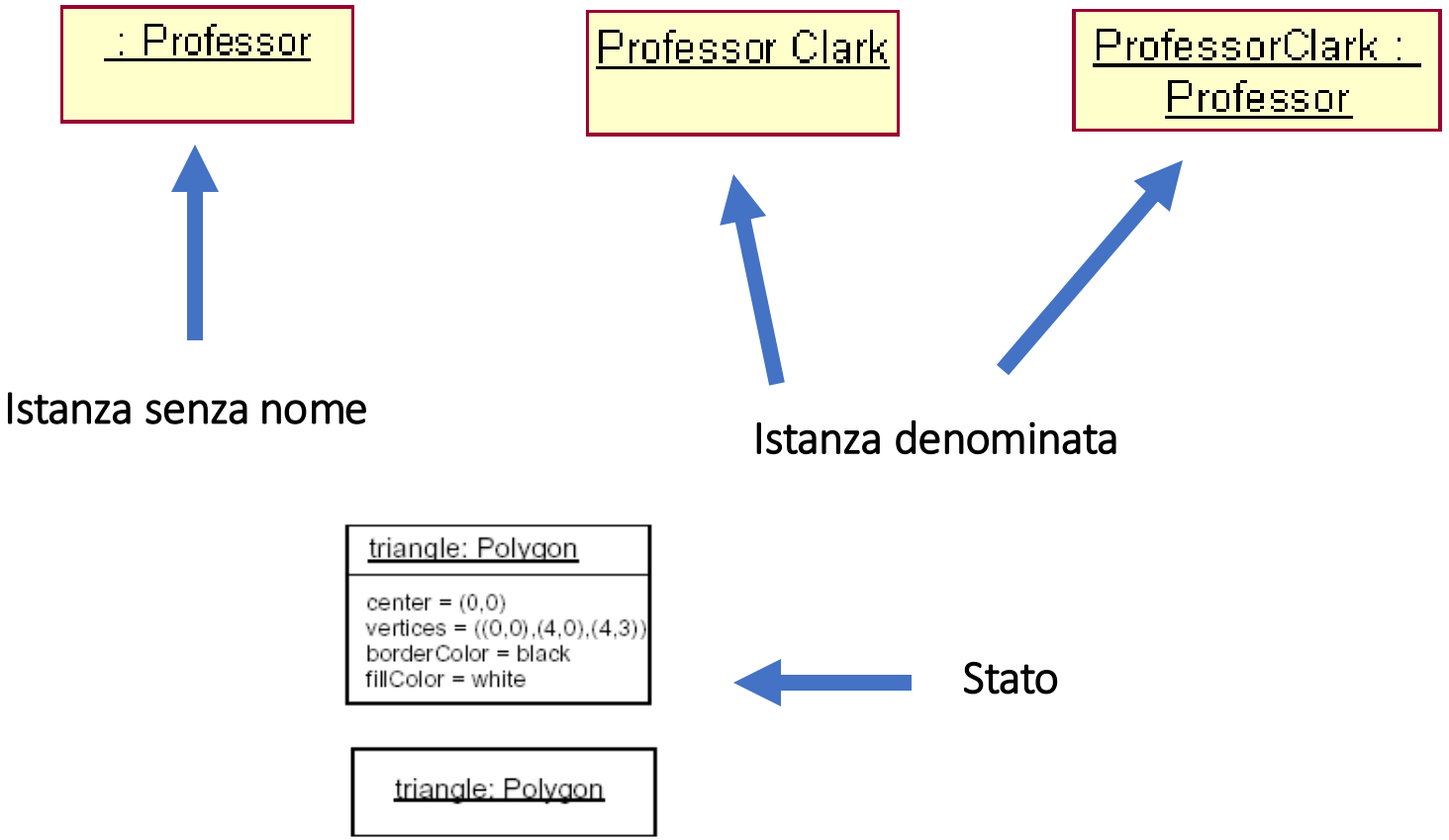
» Formalmente
  - Manifestazione concreta di un'astrazione
  - Entità con un confine e un'identità ben definiti *che* incapsulano *stato* e *comportamento*
  - Istanza di una classe

» Ad esempio: la Ferrari di Schumacher, la Ferrari di Barrichello, il mio computer

Oggetto

- Stato
  - Possibile condizione in cui l'oggetto potrebbe esistere e generalmente cambia nel tempo
  - Implementato utilizzando 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 dei messaggi a cui può rispondere (operazioni)
- Identità
  - Permette di distinguere due oggetti anche se hanno lo stesso stato e lo stesso valore nei loro attributi

Rappresentazione in UML



Classe

» Descrizione di un gruppo di oggetti con proprietà (attributi), comportamento (operazioni), relazioni e semantica comuni
  - Un oggetto è un'istanza di una classe

» Astrazione che
  - Sottolineare le caratteristiche rilevanti
  - Sopprime altre funzionalità

Esempio di classe

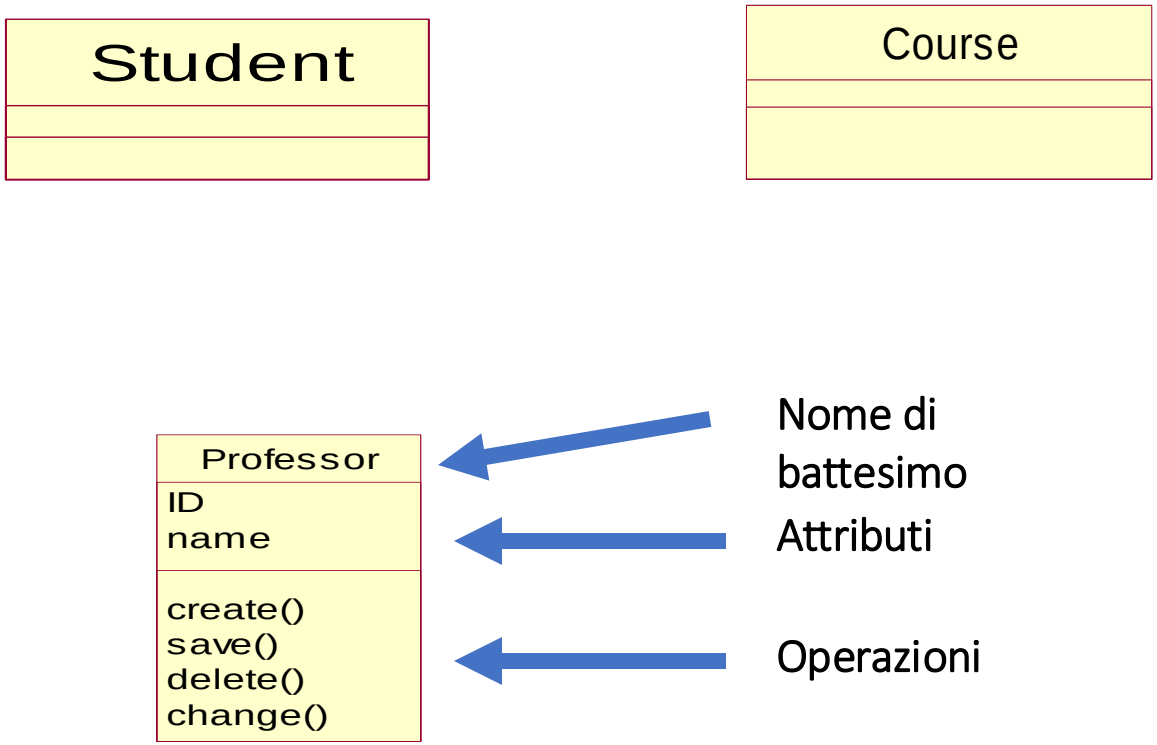
» Nome di battesimo
  - Corso
» Proprietà
  - Nome, Luogo, Durata, Titoli di coda, Inizio, Fine
» Comportamento
  - Aggiungere uno studente
  - Cancellazione studente
  - Controlla se è pieno

Relazione tra classe e oggetto

» La classe è una definizione astratta di un oggetto
  - Definisce la struttura e il comportamento di ogni oggetto nella classe
  - Serve come *modello* per la creazione di oggetti

» Gli 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 di battesimo
  - Semplice
  - Percorso (prefisso + ":" + nome classe)
- Convenzione
  - Lettera iniziale maiuscola

Classe: Attributi

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

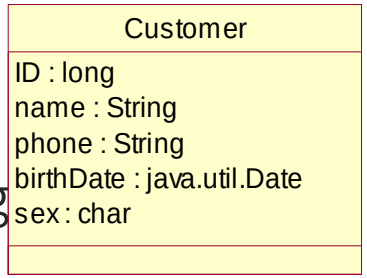
» Rappresenta le proprietà delle cose che stai modellando che sono condivise da tutti gli oggetti di quella classe

» Esempio

- Il muro ha un'altezza, una larghezza e una profondità
- Il cliente ha un nome, un indirizzo, un numero di telefono

Classe: Attributi

» Nome di battesimo
  – Stringa di testo
» Tipo
  – Ad esempio int, float, double, String
» Valore iniziale
» Convenzione
  – Lettera iniziale minuscola



Classe: Operazioni

» Implementazione di un servizio che può essere richiesto da qualsiasi oggetto

» È un'astrazione di qualcosa che è condiviso tra tutti gli oggetti di quella classe

» Rappresenta un verbo o una frase

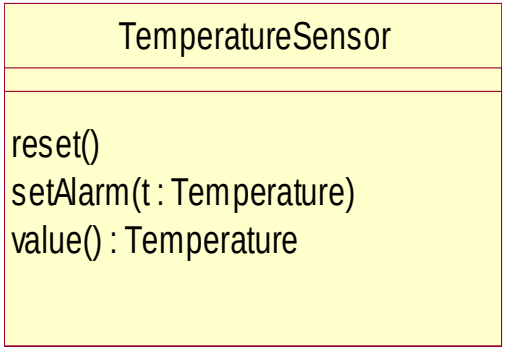
» Generalmente l'invocazione di un'operazione modifica lo stato dell'oggetto

» Esempi

- Il rettangolo ha operazioni di spostamento e ridimensionamento

Classe: Operazioni

» Nome di battesimo
  – Stringa di testo
» Firma
  – elenco separato da virgole di
    » Nome del tipo di valore predefinito
» Tipo di ritorno
  – Per le "funzioni"
» Convenzione
  – Lettera iniziale minuscola



Classi astratte

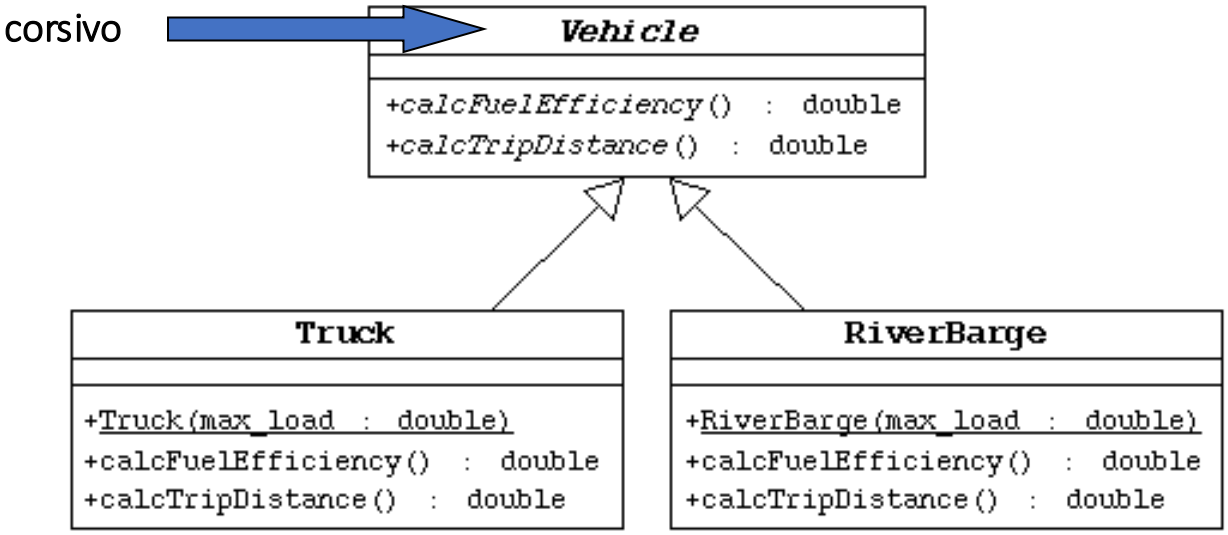
» Classe con operazioni il cui corpo non è definito

» È possibile dichiarare una classe astratta anche se non ha operazioni astratte

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

» Può contenere operazioni che hanno implementazione

Rappresentazione in UML



Relazioni

» Una relazione è una connessione tra cose (ad esempio classi, interfacce, componenti, pacchetti)

» Una relazione fornisce un percorso per la comunicazione tra oggetti

Associazione

» Rappresenta una relazione strutturale tra oggetti di classi diverse

» Collegamento bidirezionale tra classi, è possibile navigare da un oggetto di una classe all'altro e viceversa

» È possibile avere associazioni circolari, cioè tra oggetti della stessa classe

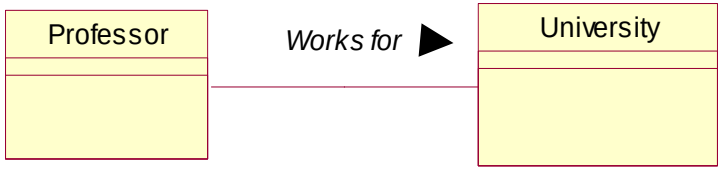
» L'associazione che collega due classi è detta binaria; n-air che collega n classi (poco utilizzata)

» Rappresentato da una linea continua che collega le due classi

Associazione: Nome

» Descrive la natura dell'associazione

» È possibile dare una direzione al nome tramite un triangolo che punta nella direzione in cui si desidera che la direzione venga letta

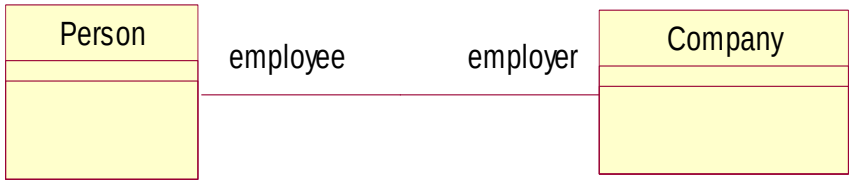


Associazione: Ruolo

» Specificare l'aspetto che una classe svolge nell'associazione

» Ha un nome e viene posizionato accanto alla classe che svolge quel ruolo nell'associazione rispetto alle altre

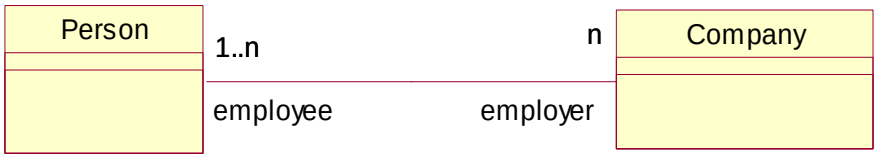
» L'uso del ruolo o del nome sono reciprocamente esclusivi



Associazione: Molteplicità

» Definisce quanti oggetti partecipano a una relazione
  - Specifica il numero di istanze di una classe che è correlata a UN'istanza dell'altra classe

» Applicato alla fine di ogni associazione



Associazione: Molteplicità

Valore	Descrizione
NO	Numero illimitato di istanze
1 (predefinito)	Solo un'istanza
0..n	Zero o più istanze
1..n	Una o più istanze
0..1	Zero o un'istanza
<letterale>*	Numero esatto di istanze
<letterale>..n	Numero esatto o più istanze
<letterale>..<letterale>	Intervallo specificato di istanze
<letterale>..<letterale>, <letterale>	Intervallo più numero specificato di istanze
<letterale>..<letterale>, <letterale>..<letterale>	Il numero di istanze sarà compreso in uno degli intervalli specificati
* Dove <letterale> è un numero intero maggiore o uguale a 1	

Associazione: Molteplicità

» Se la molteplicità è maggiore di 1, l'insieme degli elementi associati può essere ordinato o meno

» Specificato tramite vincolo alla fine dell'associazione

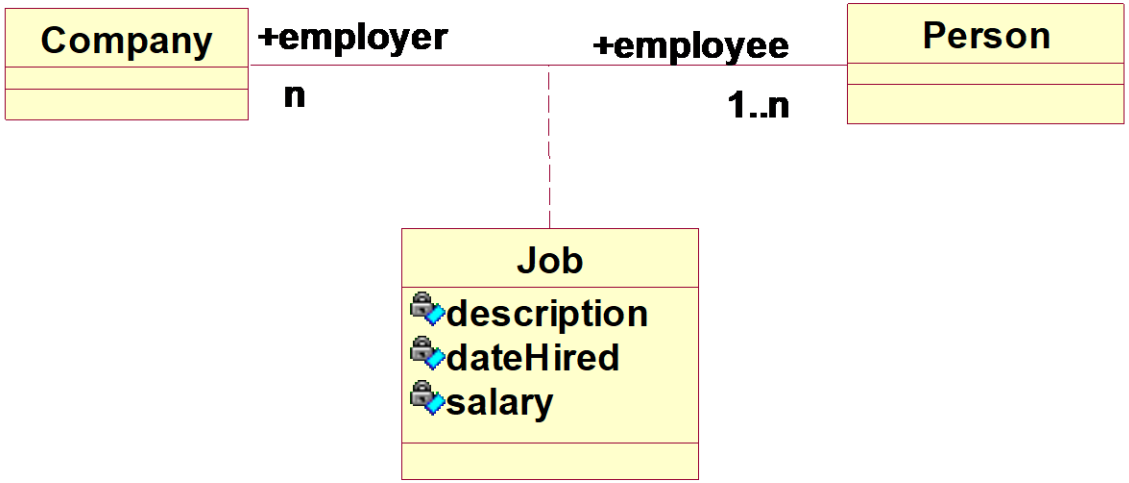
- **non ordinato** : elementi forma un insieme non ordinato (predefinito)
- **ordinato** : elementi hanno un ordinamento e io non sono ammessi duplicati ({ordinato})

» La relazione ordinata è specificata generando codice dipendente dal linguaggio di implementazione

Associazione: Classe di associazione

» Un'associazione può possedere _ al di là al molteplicità , ruoli e visibilità Anche dal proprietà strutturale e comportamentale

» Esempio



Associazione: Aggregazione

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

» C'è bisogno di modellare situazioni in cui una classe esprime una nozione che è concettualmente più grande delle altre che la costituiscono

» Rappresenta una relazione *ha -a , consiste -di , contiene , è -parte-di*

Associazione: Aggregazione

» È necessario utilizzare relazioni di tipo "tutto-parte" (*tutto - parte*), 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 chiamate **Aggregazione**

» Rappresentato con una linea che collega gli oggetti correlati utilizzando un diamante posto accanto alla classe completa

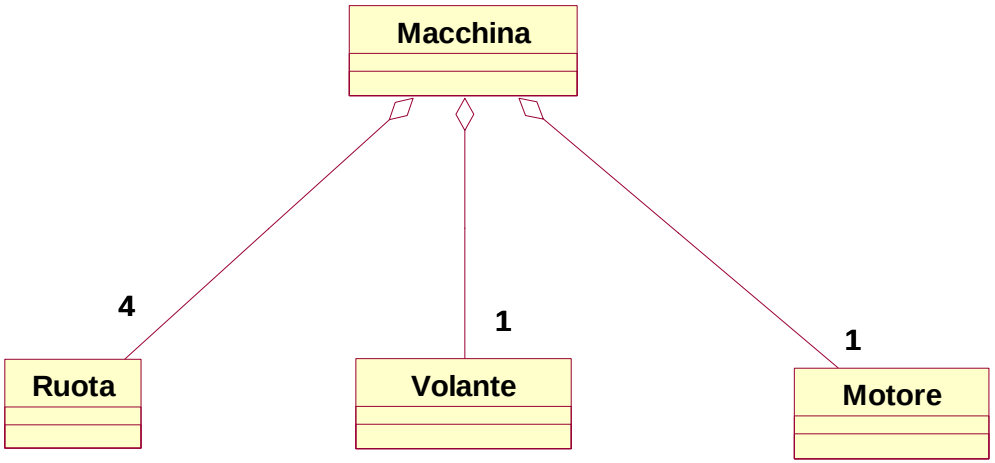
Associazione: Aggregazione

» Esempio
  - Automobile composta da ruote, motore, volante, sedili, ...

» Differenza rispetto all'associazione puramente concettuale

» Le aggregazioni circolari non hanno senso
  - Classe A composta da una Classe B; B composta da una Classe C che a sua volta è composta da Classe A

Associazione: Aggregazione



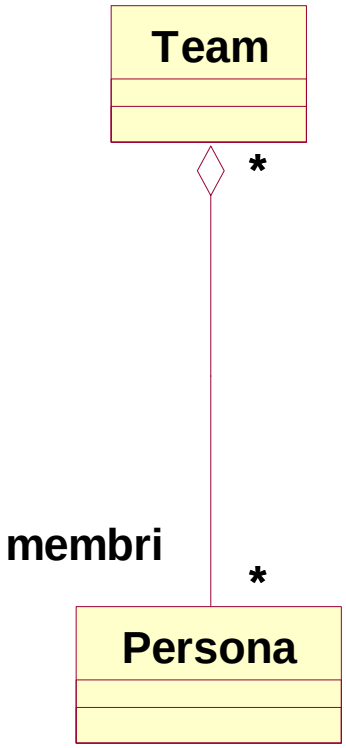
Associazione: Aggregazione condivisa

» Caso speciale di aggregazione normale

» La classe di parte può essere parte di qualsiasi intero

» E' condiviso se la molteplicità nella parte intera è maggiore di uno

Associazione: Aggregazione condivisa



Associazione: Composizione

» Forma di aggregazione con una forte connotazione di possesso e una (quasi) coincidenza del ciclo vitale tra istanze dal classi “ parte ” e l'istanza dal classe “ tutto ” (classe composto)

» Partire Potere essere generato anche in un secondo momento al creazione dell'istanza dal classe composto , ma uno una volta che questi vengono generati vivono e sono distrutto con l'istanza dal classe composto da membri

Associazione: Composizione

» La classe composita si occupa di eliminare le istanze delle sue parti in un momento precedente alla sua distruzione

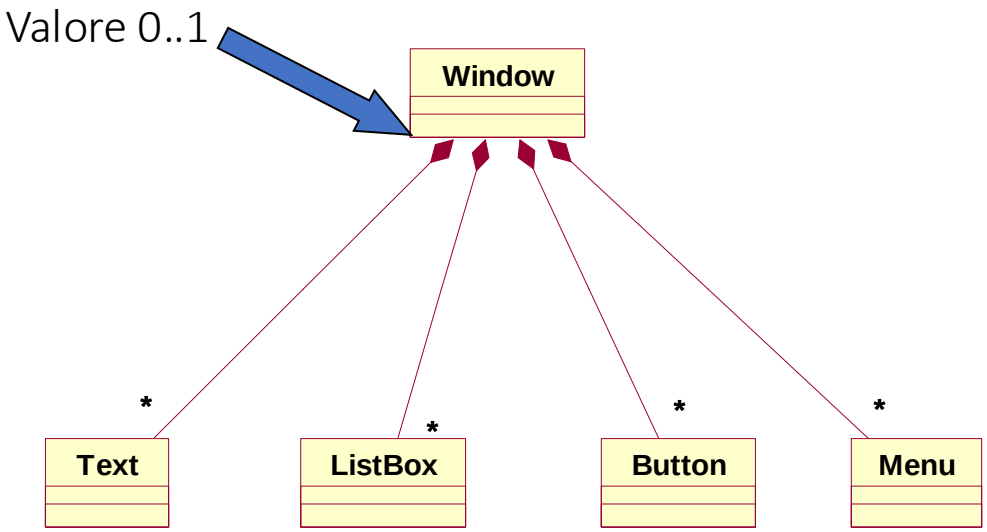
» Un oggetto può essere parte di un solo oggetto composito alla volta

- Esempio: un *frame* appartiene a una sola *finestra*

» Tuttavia, in un'aggregazione una parte può essere condivisa tra più composti

- Esempio: un *muro* può appartenere a più *stanze*

Associazione: Composizione



Associazione vs. dipendenza

» L'associazione è una relazione strutturale (quindi "persistente"), che evidenzia classi semanticamente correlate

» La dipendenza ha un carattere transitorio, un legame che si instaura (o almeno così dovrebbe essere) temporaneamente, per il periodo di tempo necessario a usufruire di un servizio, a creare un oggetto, ecc., e poi perde di significato.

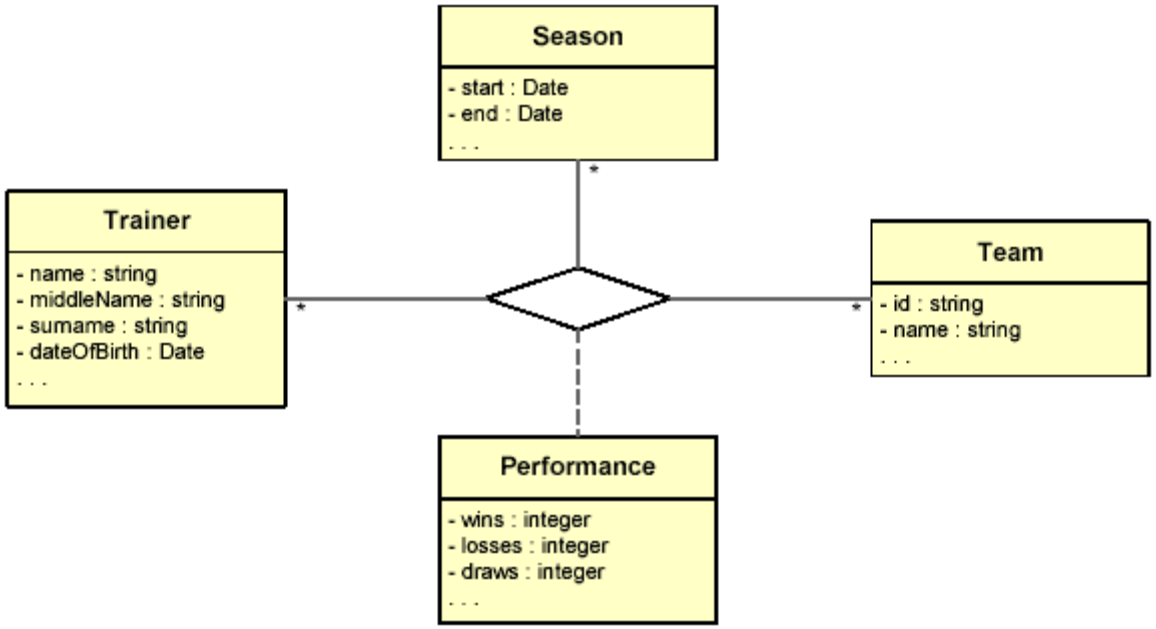
associazione n-air

» Relazione che coinvolge più di due classi

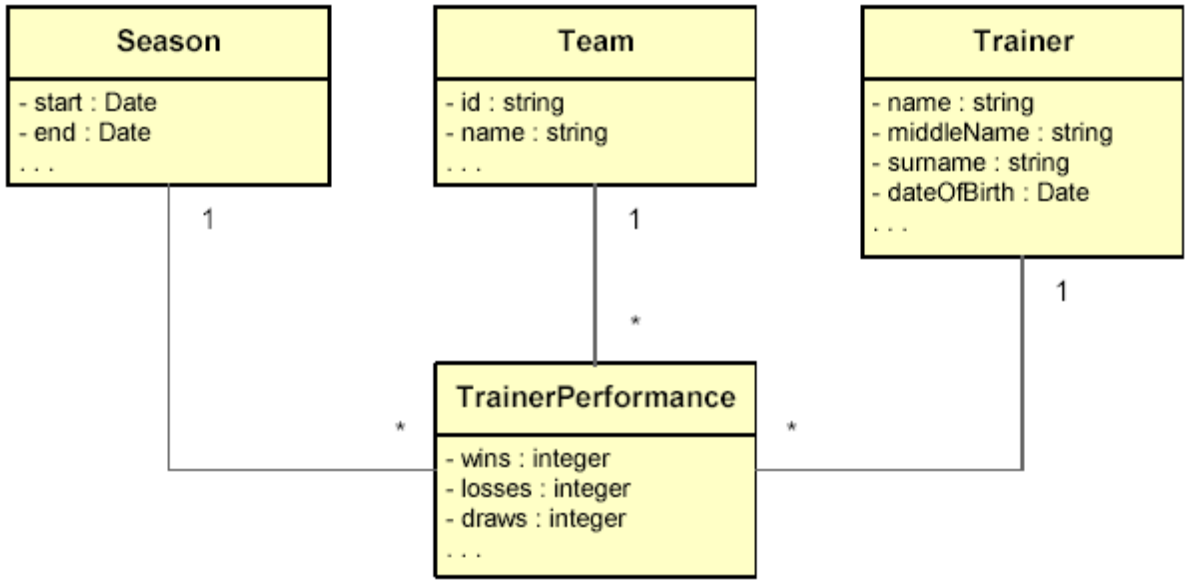
» Rappresentato da un rombo in cui sono collegate le classi appartenenti alla relazione

» Difficoltà nell'attribuzione dei valori delle molteplicità. Esse specificano, per ogni classe, il numero potenziale di istanze della classe che possono partecipare alla relazione, fissando i valori delle altre *n–1* classi

associazione n-air



associazione n-air



Trasformazione di un'associazione n-aria in associazioni binarie

Generalizzazione

» E' un relazione Fra UN Che cosa Di più generale (detto superclasse o genitore) ed UN Di più specifico (detto sottoclasse o figlia)

» Lui viene disse Anche la relazione " *è-un-tipo-di* "

» Oggetti figlio Potere essere utilizzato al posto degli oggetti padre (principio di sostituibilità di Liskov) ma non viceversa , cioè il padre non è un sostituto del figlio

Generalizzazione

» Il figlio eredita tutte le proprietà dei suoi padri, cioè attributi e operazioni

» A volte il bambino può ignorare le operazioni del genitore

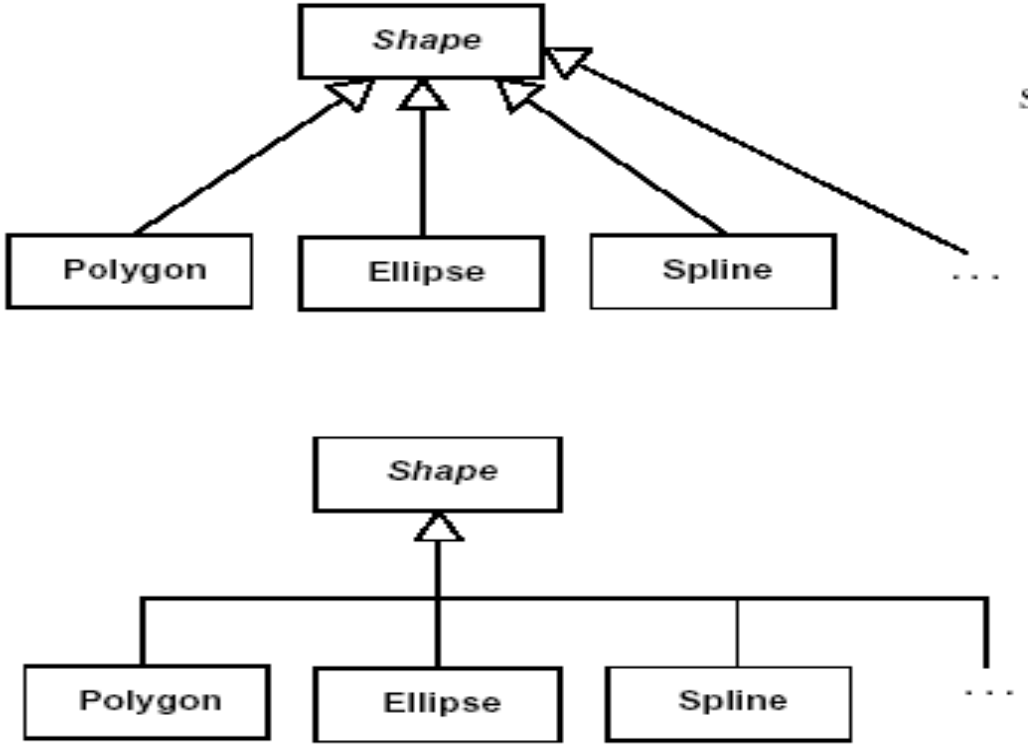
» Relazione

- Transitivo: se C generalizza (cioè eredita) B e B eredita da A, ne consegue che anche C generalizza A
- Antisimmetrico: se A eredita da B, il contrario non è assolutamente vero. Se questo è vero allora A e B sono uguali

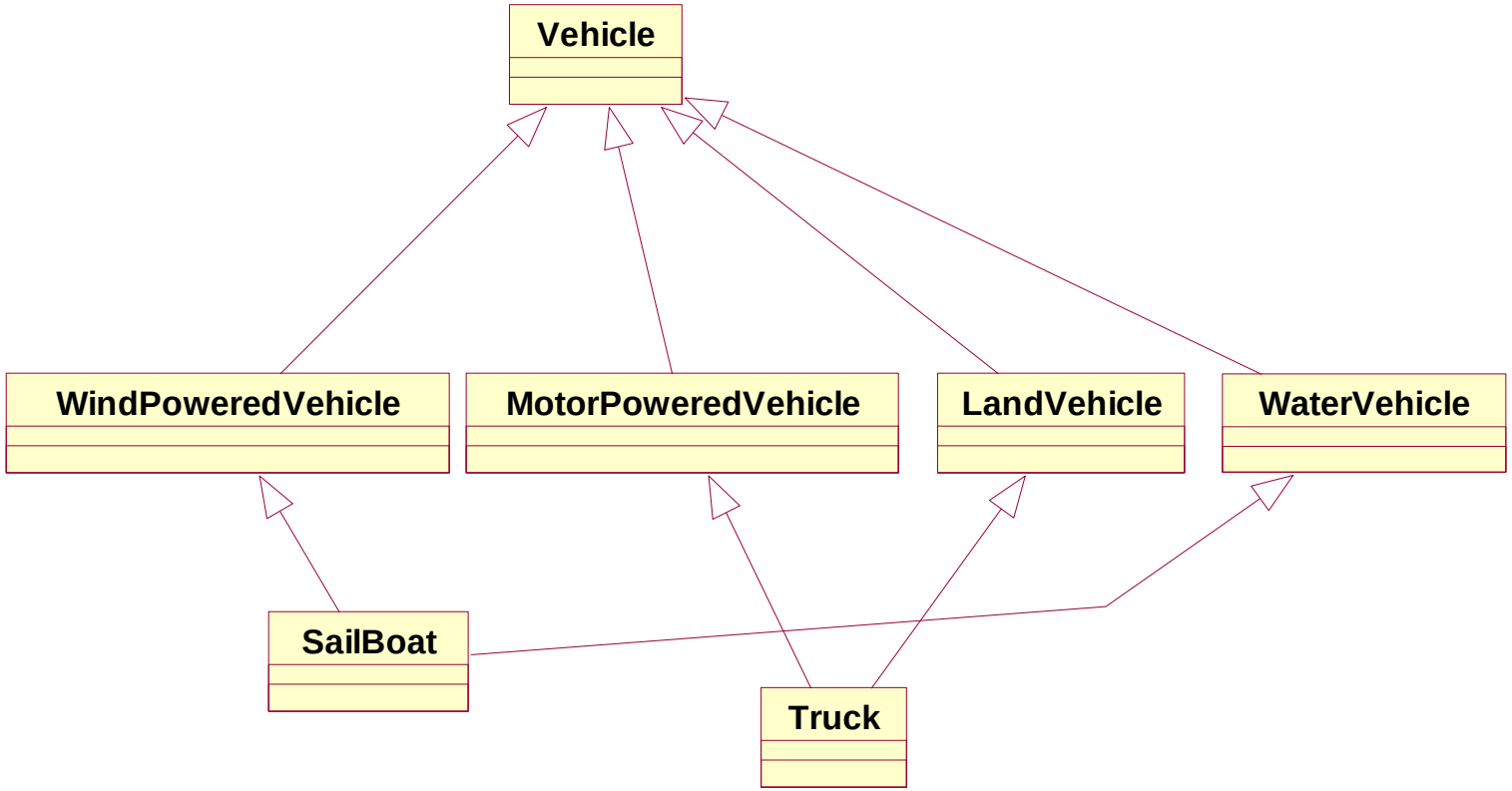
Generalizzazione

- Disegnato con una freccia chiusa che punta verso il padre
- Tipi di generalizzazione
  - Singolo (Java)
    - Hai un solo padre
  - Multiplo (C++ e NO Java)
    - È possibile avere più padri

Generalizzazione



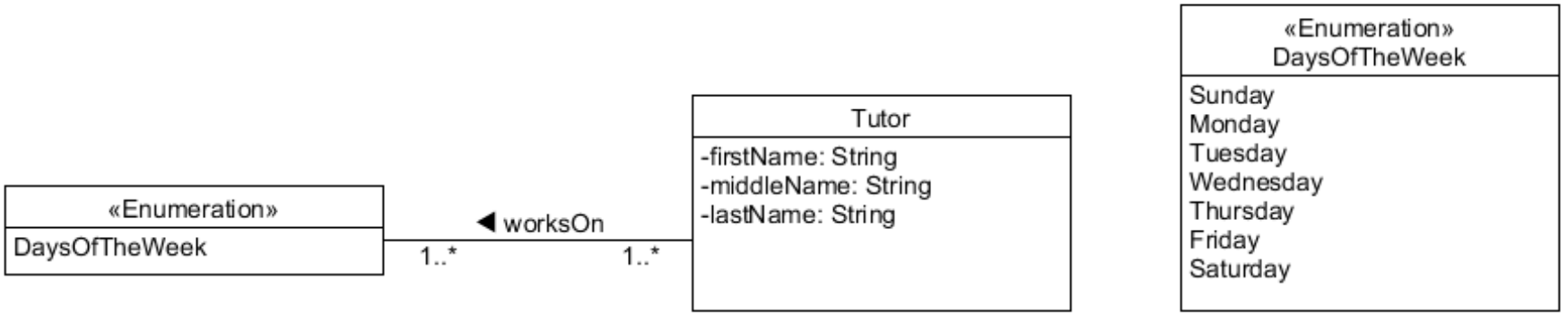
Generalizzazione



E numerazione

» Le enumerazioni sono elementi modello nei diagrammi di classe che rappresentano tipi di dati definiti dall'utente

» Le enumerazioni contengono set di identificatori denominati che rappresentano i valori



Identificatore

» Una sequenza di *lettere* e *cifre di lunghezza illimitata*. Sono utilizzate per nomi *di classi* , nomi *di attributi , nomi di operazioni* e sono *sensibili alle maiuscole e alle minuscole* . Le parole chiave e i letterali true, false e null non identificano

» Esempi

- Prodotto
- prodotto
- $nomeutente

Letterale (1)

» Rappresenta il valore costante che ogni tipo primitivo (String o null) può assumere

» Tipi
  - Int
  - Galleggiante
  - Booleano: `vero` o `falso`
  - Cartello
  - Corda
  - `null`

Letterale: Int (2)

» Il tipo è `int`
» Quando usato come suffisso `L` o `l` diventa `lungo`
» Espresso come
  – decimale (base 10): numero
  – ottale (base 8): `0` numero
  – esadecimale (base 16): numero `0x`
» Esempi
  – Decimale `int` : `10`
  – Decimale `lungo` : `10L`
  – `Int` ottale : `010`
  – Esadecimale `int` : `0x10`

Letterale: Int (3)

» Tipo letterale decimale più grande `int` è `2147483648` (2^{31})

» Da 0 a `2147483647` `un int` letterale può apparire ed essere utilizzato ovunque sia possibile utilizzare un `int`

» `2147483648` può apparire solo come numero negativo

» Il più grande letterale decimale di tipo `long` è `9223372036854775808L` (2^{63})

Letterale: Int (4)

» Il tipo letterale **positivo** più grande `int` esadecimale e ottale sono rispettivamente `0x7fffffff` e `017777777777` (`2147483647`)

» Il più grande letterale di tipo **negativo** `interno` esadecimale e ottale sono `0x80000000` e `020000000000` che rappresentano `-2147483648`

» `0xffffffff` e `037777777777` rappresentano -1 rispettivamente in esadecimale e ottale

Letterale: galleggiante (5)

» Composto da
  - Parte intera Punto decimale (.) Parte frazionaria
  - Esponente: `E` o `e` seguito da un numero intero con segno
  - Suffisso
    - » `Per of galleggiante`
    - » `D od doppio` (predefinito)

» Esempi
  - Doppio : `1e1, 2., .3, 0.0, 3.14`
  - Flottante : `1e1f, 2.f, .3f, 0f, 3.14f`

Letterale: galleggiante (6)

» Maggiore è il valore letterale float positivo, è `3.40282347e+38f`

» Numero finito positivo letterale diverso da zero, più piccolo è `1,40239846e-45f`

» Più grande è il doppio letterale positivo, è `1.79769313486231570e+308`

» Il numero letterale positivo doppio finito non zero più piccolo è `4,94065645841246544e-324`

Letterale : Carattere (7)

» Espresso come un carattere o una sequenza *di escape* racchiusa tra virgolette singoleTipo è sempre `char`

» Esempi

- `'C'`
- `'\\'`
- `' \n ', ' \t ', ' \b ', ' \r ', '\'', '\”'`

Letterale: Stringa (8)

» Espresso come una sequenza di zero o più caratteri racchiusi tra virgolette doppie

» Ogni carattere può essere sfuggito

» Esempi

- `"Benvenuto"`
- `"\""`
- `" \N "`
- `"ciao" + " ciao "`

Tipi di dati

» Permette di esprimere la natura del dato Indica come il dato può essere rappresentato e interpretato La stessa sequenza può rappresentare un intero o un carattere Determina l'intervallo di valori che un dato può assumere Specifica le possibili operazioni sui dati

Tipi

- Ogni tipo di dati ha
  - Un nome
    - `int, doppio, carattere`
  - Un insieme di possibili letterali
    - `3, 3.1, 'c'`
  - Un insieme di operazioni lecite
    - `+, *, /, %, ……`
- UML considera
  - **Tipi primitivi Tipi di riferimento**

Tipi primitivi

- Logico
  - `booleano`
- Numerico
  - Integrante
    - `byte, corto, int, lungo e char`
  - Galleggiante
    - `doppia e galleggiante`

Tipi primitivi: Booleano

» Il valore `booleano` rappresenta una condizione di verità o falsità

» Un attributo tipizzato `booleano` può rappresentare un valore a due stati

- come un interruttore che è acceso o spento

» `VERO` e `false` sono gli unici valori consentiti

» Esempio

- `booleano b = falso;`

Tipi primitivi: Char

» Caratteri

» È rappresentato dallo schema di codifica Unicode

Tipi primitivi: Char

Caratteri	Nome	Valore Unicode
\B	Indietro	\u0008
\T	Tabulazione	\u0009
\N	Avanzamento riga	\u000a
\R	Invio a capo	\u000d
" "	Doppi apici	\u0022
\'	Apici singoli	\u0027
\\	Barra rovesciata	\u005c

Tipo primitivi: byte, short, int e long

» La rappresentazione avviene mediante la notazione a due complementi

Tipo	Dimensione	Allineare
byte	1 byte	-128 a 127
corto	2 byte	-32768 a 32767
intero	4 byte	-2147483648 a 2147483647
lungo	8 byte	-9223372036854775808 a 9223372036854775807

Ragazzi primitivi: float e double

» Definisci i numeri con parti frazionarie I numeri doppi sono chiamati numeri a doppia precisione

Tipo	Dimensione	Allineare
`galleggiante`	`4 byte`	`±3,40282347E+38F` (6 o 7 cifre decimali significative)
`raddoppiare`	`8 byte`	`±1,79769313486231570E+308` (15 cifre decimali significative)

Tipo di riferimento

» Sono puntatori (riferimenti) ad oggetti di una classe