

UML: Class diagram

Gianluca Pironato

Class diagram in UML

- ▶ È un diagramma utile a osservare alcuni aspetti strutturali, quindi statici, del sistema;
- ▶ In particolare, ci permette di cogliere al volo **classi e relazioni** essenziali del contesto che stiamo trattando;
- ▶ Il codice del SW in esame riflette ovviamente questa organizzazione a oggetti.

Elementi fondamentali

- ▶ **Classe:** rappresenta un'entità, raffigurata da un rettangolo diviso in tre sezioni.
 - ▶ sezione superiore: nome della classe (solitamente in maiuscolo o in corsivo se astratta)
 - ▶ sezione centrale: attributi (campi, variabili, proprietà)
 - ▶ sezione inferiore: metodi (operazioni, funzioni)
- ▶ Ogni istanza della classe è un oggetto.

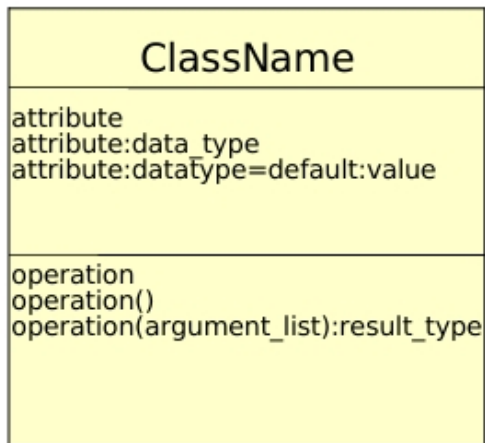
Elementi fondamentali

- ▶ **Attributi:** sono le proprietà significative che descrivono un'istanza di una classe; si indica dopo ":" il tipo di variabile;
 - ▶ es. per una classe `Animal` si possono avere attributi come `name` (stringa), `id` (intero), `age` (intero).
- ▶ **Metodi:** azioni che descrivono il comportamento della classe; si indica dopo ":" il tipo di valore restituito (se presente);
 - ▶ es. per `Animal` si possono definire metodi come `setName()` o `eat()`;
 - ▶ tra le tonde, posso indicare anche un parametro e dopo ":" il tipo di questo.
- ▶ **N.B. sia attributi che metodi vengono preceduti da un simbolo di visibilità.**

Visibilità

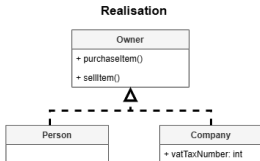
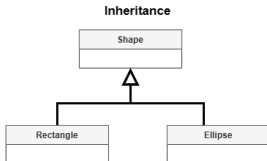
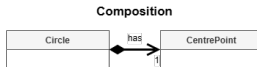
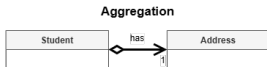
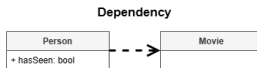
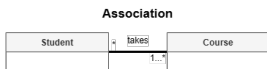
- ▶ La visibilità determina l'accesso ad attributi e metodi:
 - ▶ - indica **private**: accessibile solo all'interno della classe;
 - ▶ + indica **public**: accessibile da qualsiasi altra classe;
 - ▶ # indica **protected**: accessibile dalla classe e dalle sue sottoclassi;
 - ▶ ~ indica **package**: accessibile dalle classi dello stesso package.
- ▶ in genere, gli attributi sono dichiarati `private` o `protected`, mentre i metodi sono `public`;
- ▶ inoltre, per l'accesso agli attributi è bene predisporre un'interfaccia di metodi setter e getter.

Notazione per classe



Elementi fondamentali - prosegue

- **Relazioni tra classi:** associazione, aggregazione, composizione, dipendenza, ereditarietà, implementazione, ecc.
- Ciascuna con notazione grafica diversa.

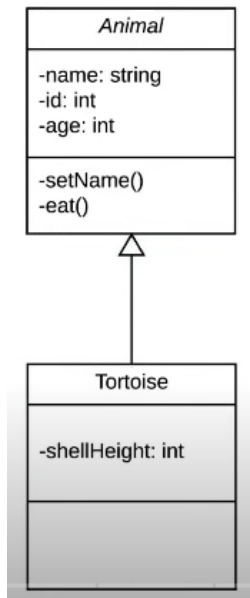


Ereditarietà

- ▶ Permette di definire una relazione tra una classe generica (superclasse, genitore) e classi specializzate (sottoclassi, figli).
- ▶ I figli ereditano attributi e metodi dal genitore.
- ▶ Si rappresenta con una freccia con "testa vuota" che punta dalla sottoclasse alla superclasse.
- ▶ es. le classi Cat e Dog possono ereditare dalla classe Animal.
- ▶ Una classe figlio, essendo più specifica, deve obbligatoriamente avere qualcosa in più rispetto alla classe genitore.

N.B. se il nome della classe genitore è scritto in corsivo, allora questa è da considerarsi una classe astratta (non viene mai direttamente istanziata nel codice).

Ereditarietà



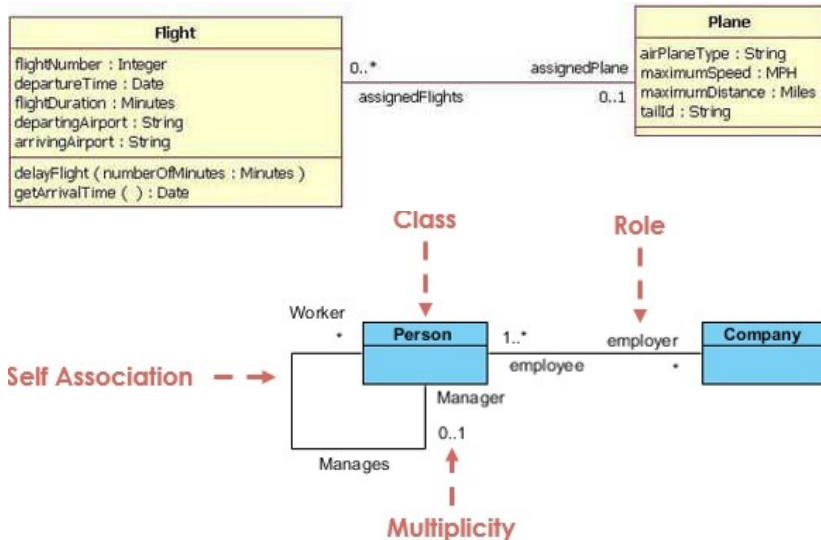
Associazione

- ▶ Permette di definire una relazione semplice tra due classi.
- ▶ es. Cat mangia Mouse
- ▶ Si rappresenta con una linea accompagnata da un verbo e/o dal ruolo e dalla cardinalità minima e massima (o molteplicità) su ogni lato.
- ▶ In genere, sono associazioni tutte le relazioni che non rendono una classe dipendente dall'altra.

Molteplicità

- ▶ Specifica quanti oggetti di una classe possono essere associati a un oggetto di un'altra classe.
- ▶ esempi:
 - ▶ 1: esiste esattamente un oggetto (1 a 1);
 - ▶ 0..1: opzionale, al massimo uno;
 - ▶ 1..*: almeno uno, ma possono essercene molti;
 - ▶ 0..*: opzionale e possono essercene molti.
- ▶ La molteplicità viene letta e interpretata in base al ruolo scritto vicino.

Associazione



Aggregazione

- ▶ Trattasi di un tipo di associazione utile a definire un'entità e le sue parti.
- ▶ **L'aggregazione delle parti non è però essenziale per la loro esistenza**; ovvero, le parti possono esistere indipendentemente.
- ▶ Si rappresenta con una linea che culmina in un rombo vuoto sul lato della classe "aggregante".

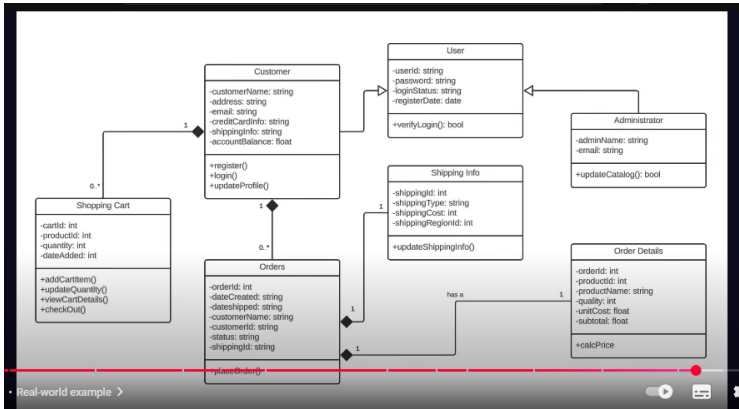


Composizione

- ▶ Trattasi di una forma più forte di aggregazione.
- ▶ **La composizione delle parti è essenziale per la loro esistenza**; ovvero, le parti non possono esistere indipendentemente.
- ▶ Si rappresenta con una linea che culmina in un rombo pieno sul lato della classe "composta".



Esempio riassuntivo



Continua...