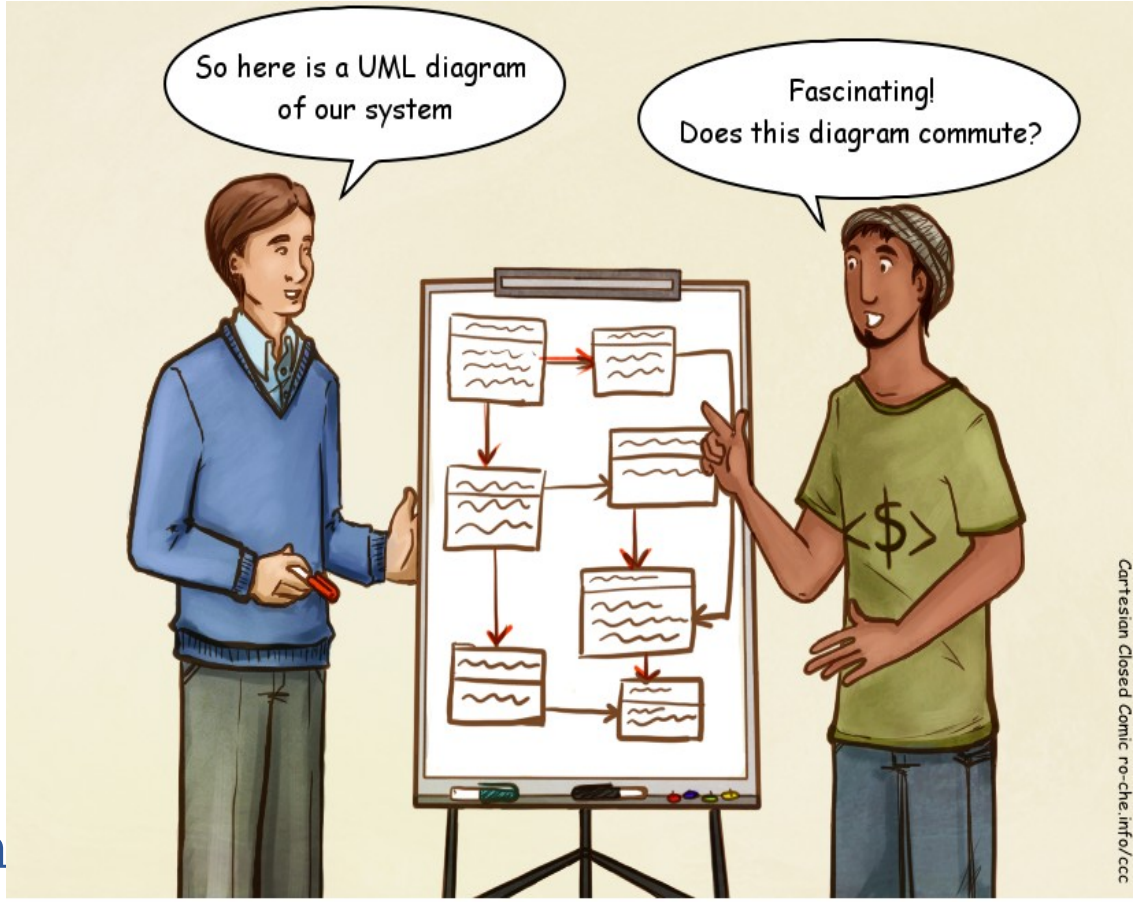
UML

Prof.Francesco Viglietti
www.in4matika.altervista



Transition from Haskell to Java can be awkward

UML OO

- **UML è un linguaggio per la modellazione orientata agli oggetti.**
- **Questo include sia l'analisi che la progettazione orientata agli oggetti:**
  - (OOA) analisi: capire cosa deve fare il sistema, senza occuparsi dei dettagli implementativi
  - (OOD) progettazione: capire come il sistema raggiunge il suo scopo, come viene implementato
- **UML offre strumenti di modellazione OO in entrambi gli ambiti; frammenti differenti di UML sono impiegati in diverse fasi del processo di sviluppo (anche se UML stesso non fornisce indicazioni sul suo utilizzo).**

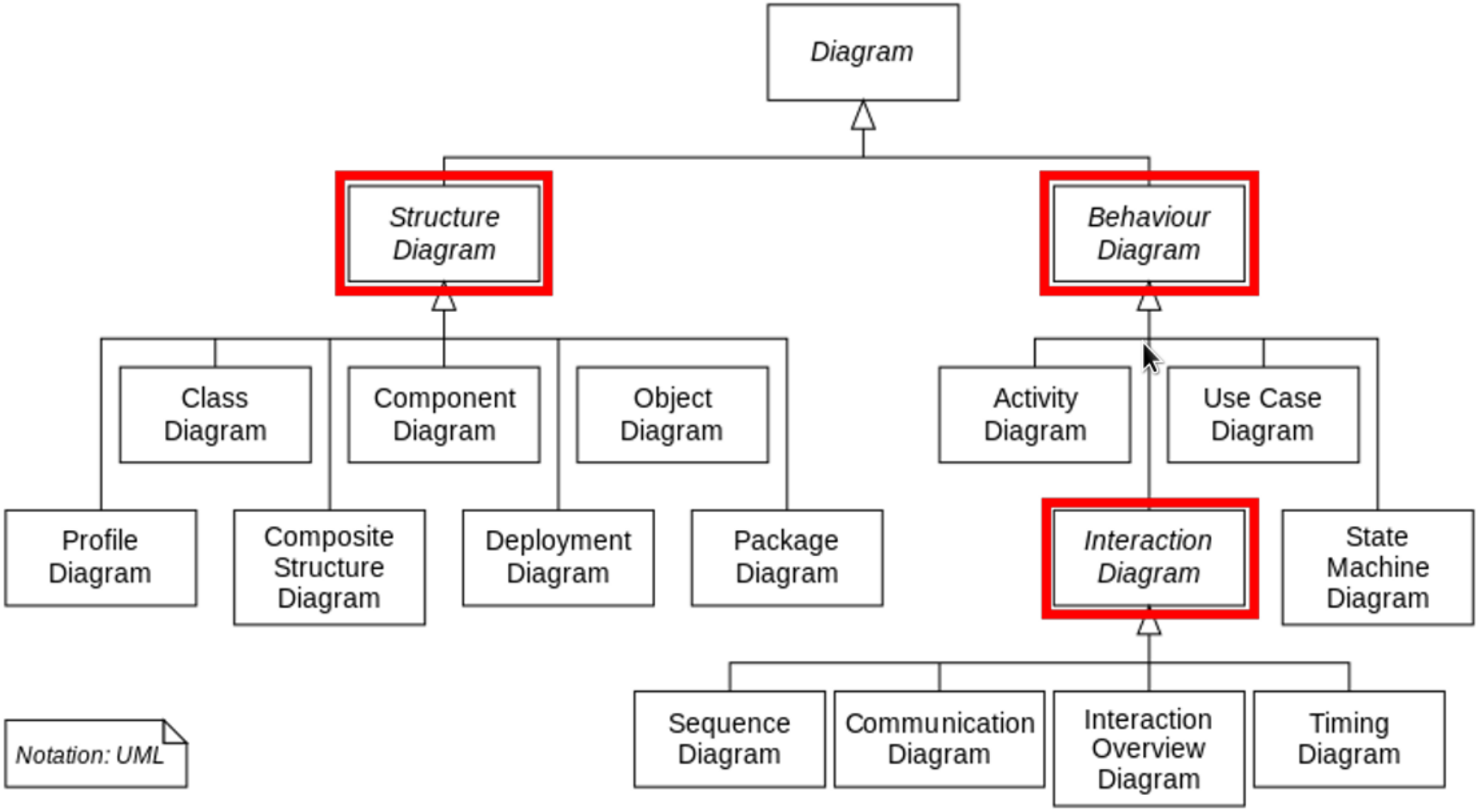
Principi OO

- Abstraction (Astrazione): usare classi per astrarre la natura e le caratteristiche di un oggetto, che è un'istanza della propria classe di appartenenza.
- Encapsulation (Incapsulamento): nascondere al mondo esterno i dettagli del funzionamento di un oggetto; gli oggetti hanno accesso solo ai dati di cui hanno bisogno.
- Inheritance (Ereditarietà): classi possono specializzare altre classi ereditando da esse e implementando solo la porzione di comportamento che differisce.
- Polymorphism (Polimorfismo): invocare comportamento diverso in reazione allo stesso messaggio, a seconda di quale oggetto lo riceve.

I diagrammi e le viste

- **Un diagramma è la rappresentazione grafica di una parte del modello. Fornisce una vista di un sistema o una sua parte, cioè ne mette in risalto diverse proprietà. Viste:**
  - Logical: mette in risalto la scomposizione logica del sistema tramite classi, oggetti e loro relazioni
  - Development: mostra l'organizzazione del sistema in blocchi strutturali (package, sottosistemi, librerie, ...)
  - Process: mostra i processi (o thread) del sistema in funzione, e le loro interazioni
  - Physical: mostra come il sistema viene installato ed eseguito fisicamente
  - Use case: la vista che agisce da 'collante' per le altre; spiega il funzionamento desiderato del sistema

Diagrammi di UML 2.5



Entità

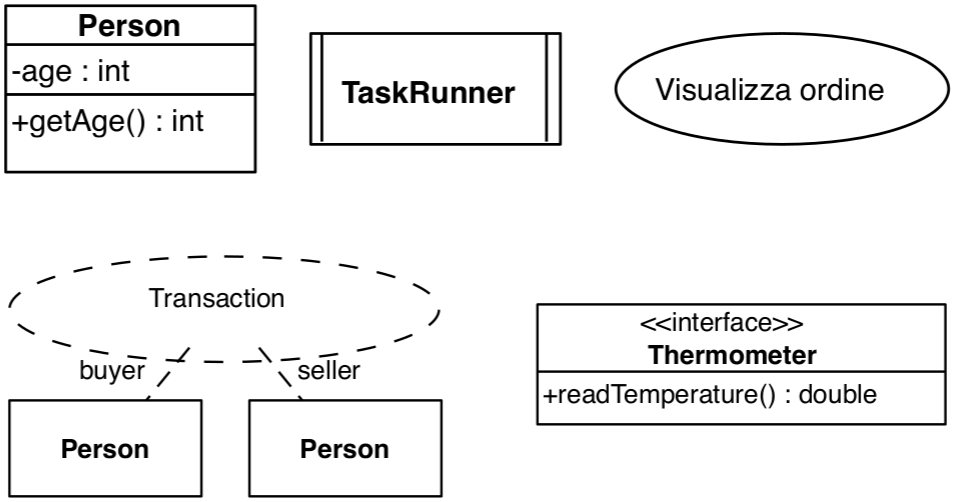
Dalla precedente slide in rosso vengono evidenziate le entità. UML prevede diversi tipi di entità che possono essere organizzati in quattro categorie:

- Strutturali
- Comportamentali
- Informative
- Raggruppamento e contenimento

Una lista delle entità usate da ogni singolo diagramma è disponibile all'url: www.uml-diagrams.org/uml-25-diagrams.html

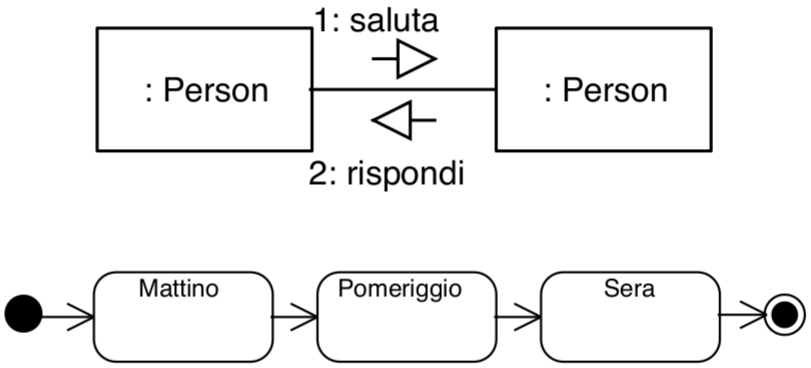
Entità strutturali

- Definiscono le "cose" (classificatori, things) del modello
- Alcuni esempi: classi, classi attive, use-case, collaborazioni, interfacce



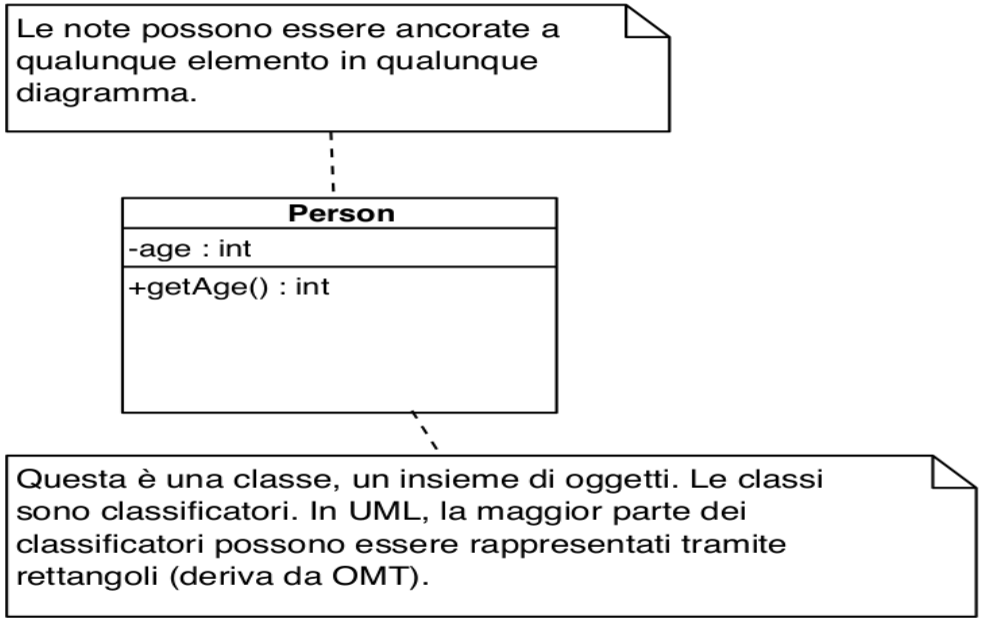
Entità comportamentali

- Descrivono il "behavior": interazioni, collaborazioni (communication), scambi di messaggi, transizioni di stato, etc.



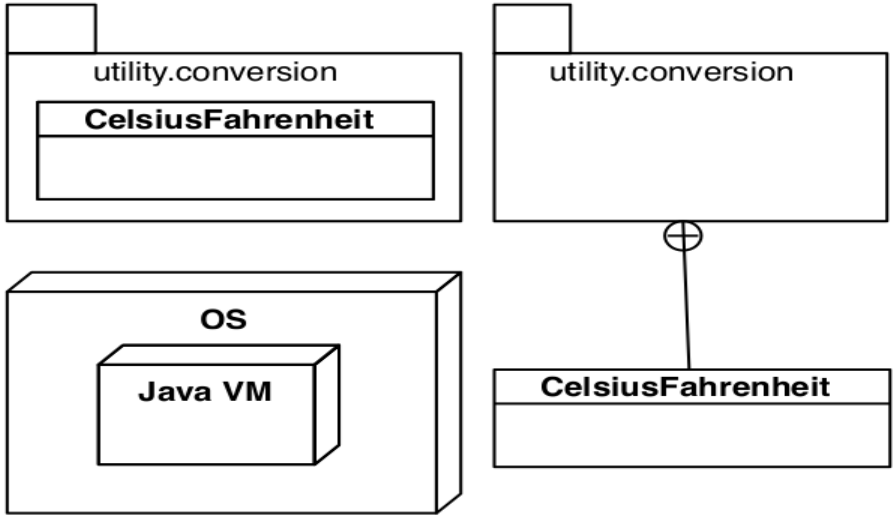
Entità informative

- Uno degli scopi principali della modellazione è la leggibilità: un diagramma non leggibile e informativo serve a poco...
- Le note UML non hanno effetti sul modello ma migliorano la leggibilità.



Entità di raggruppamento e contenimento

- I package raggruppano altri elementi e forniscono loro un namespace (che fa identificare ogni elemento con il suo nome).
- In UML, moltissimi elementi possono contenere altri elementi al loro interno, formando una struttura gerarchica, rappresentabile graficamente in vari modi.

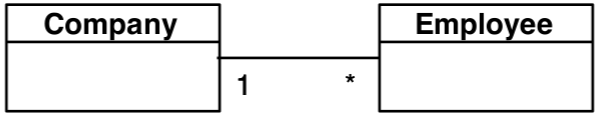


Relazioni

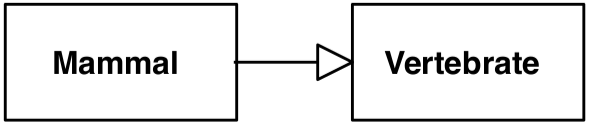
- Gli elementi del modello possono essere collegati da relazioni.
- Rappresentate graficamente tramite linee.
- Possono avere un nome.
- Quattro sottotipi fondamentali:
  - Association
  - Generalization
  - Dependency
  - Realization

Associazioni e generalizzazioni

- Un'associazione descrive l'esistenza di un nesso tra le istanze di classificatori (things) e ha varie caratteristiche, alcune opzionali.

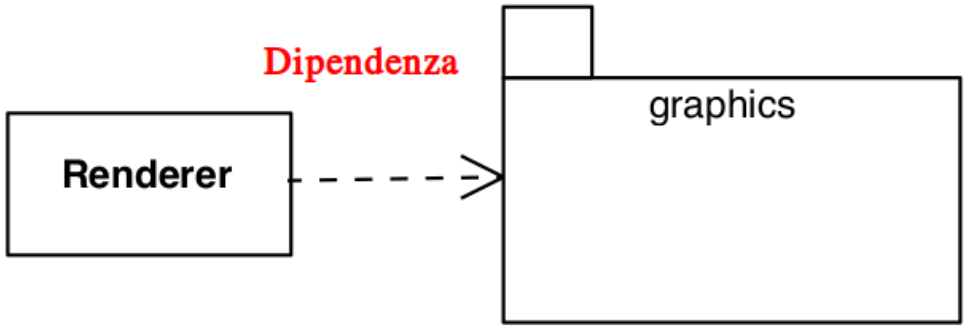


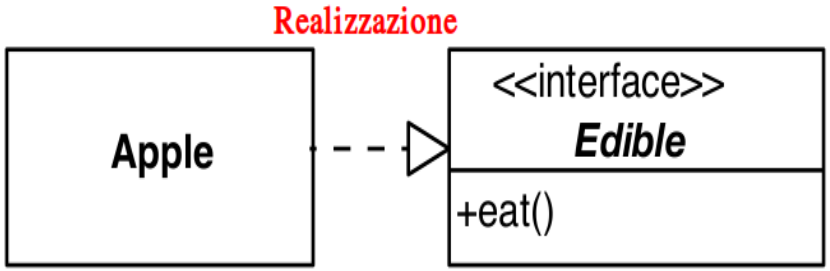
- La generalizzazione è una relazione tassonomica da un elemento specializzato verso un altro, più generale, dello stesso tipo.
  - Il figlio è sostituibile al genitore dovunque appaia, e ne condivide struttura e comportamento.



Dipendenze e realizzazione

- Una dipendenza è una relazione semantica: indica che il client dipende, semanticamente o strutturalmente, dal supplier (variazioni alla specifica del supplier possono cambiare quella del client).
- Anche la realizzazione è una relazione semantica: il supplier fornisce una specifica, il client la realizza (es: implementazione di interfacce, templates, etc.)





Frecce e stereotipi

- Un metodo mnemonico per ricordarsi il verso delle frecce in UML:
- Tutte le frecce vanno da chi sa verso chi non sa (dell'esistenza dell'altro). In una generalizzazione, il figlio sa di estendere il genitore, ma il genitore non sa di essere esteso. In una dipendenza, chi dipende sa da chi dipende, ma non vice-versa. In una realizzazione, chi implementa conosce la specifica, ma non il contrario.
- Uno stereotipo è una parola chiave tra virgolette e abbinata ad un elemento del modello. Es. «import», «utility», «interface».
- Un'altra primitiva comune ad ogni diagramma, che lo rende più informativo arricchendo la semantica dei costrutti UML.
- Gli stereotipi forniscono significato aggiuntivo ai costrutti UML.
- Possono essere usati per adattare UML a particolari ambiti e piattaforme di sviluppo.
- Stereotipi, vincoli e regole aggiuntivi vengono raccolti in profili, che costituiscono uno dei principali meccanismi di estensione di UML.

Esempio stereotipi