

UML

CLASS DIAGRAM

UML – CLASS DIAGRAM

- Analisi e progettazione
- Schede CRC
- Diagramma delle classi

Analisi vs. Progettazione

- L'analisi **modella** i concetti chiave del **dominio** del problema.
- La progettazione **adatta** il modello di analisi e lo **completa** affinché diventi **implementabile**.

In altre parole...

- L'analisi è vicina al **problema**.
- La progettazione è vicina alla **soluzione**.

Dal punto di vista di UML, non si usano primitive o diagrammi diversi, ma gli stessi tipi di diagramma con diversi livelli di dettaglio (i diagrammi di analisi sono più 'astratti' di quelli di progettazione).

Come procedere: Analisi

- Estrarre un insieme di classi di analisi dalla specifica del problema
- Ragionare su queste **classi**: quali **attributi** e quali **operazioni** devono fornire?
- Stendere una mappa delle classi e delle loro **relazioni** (associazioni).
- Modellare la dinamica delle **classi** con i *diagrammi di comportamento*.
- Procedere per **raffinamenti successivi** fino a quando il modello rappresenta efficacemente il dominio del problema.

Come procedere: Progettazione

- Si parte dal modello di analisi che contiene classi abbastanza generiche, e lo si raffina.
- I costrutti più **astratti** di UML vengono trasformati in altri più **concreti** che possono essere implementati in un linguaggio di programmazione OO.
- Finalmente si considerano i vincoli di **piattaforma** e **linguaggio**, e i requisiti **non funzionali**.
- Le classi di analisi si trasformano in classi di progettazione (non c'è corrispondenza 1 a 1)
- Ancora una volta si procede per **raffinamenti successivi**.
- Il risultato è un modello pronto per l'implementazione.

Come estrarre le classi di analisi: CRC cards

- Le carte CRC sono state proposte da Kent Beck e Ward Cunningham nel 1989 come strumento didattico per insegnare la progettazione Object-Oriented.
- Si tratta di un metodo di brainstorming iterativo che coinvolge sviluppatori, esperti, committenti.
- Servono per definire le classi principali e le loro interazioni.
- **Classe, Responsabilità, Collaborazione:**
 - **Classe:** gli oggetti più importanti
 - **Responsabilità:** compiti principali da eseguire
 - **Collaborazioni:** altri oggetti che cooperano per soddisfare una responsabilità
- Si usano post-it divisi in tre sezioni chiamate proprio in questo modo.

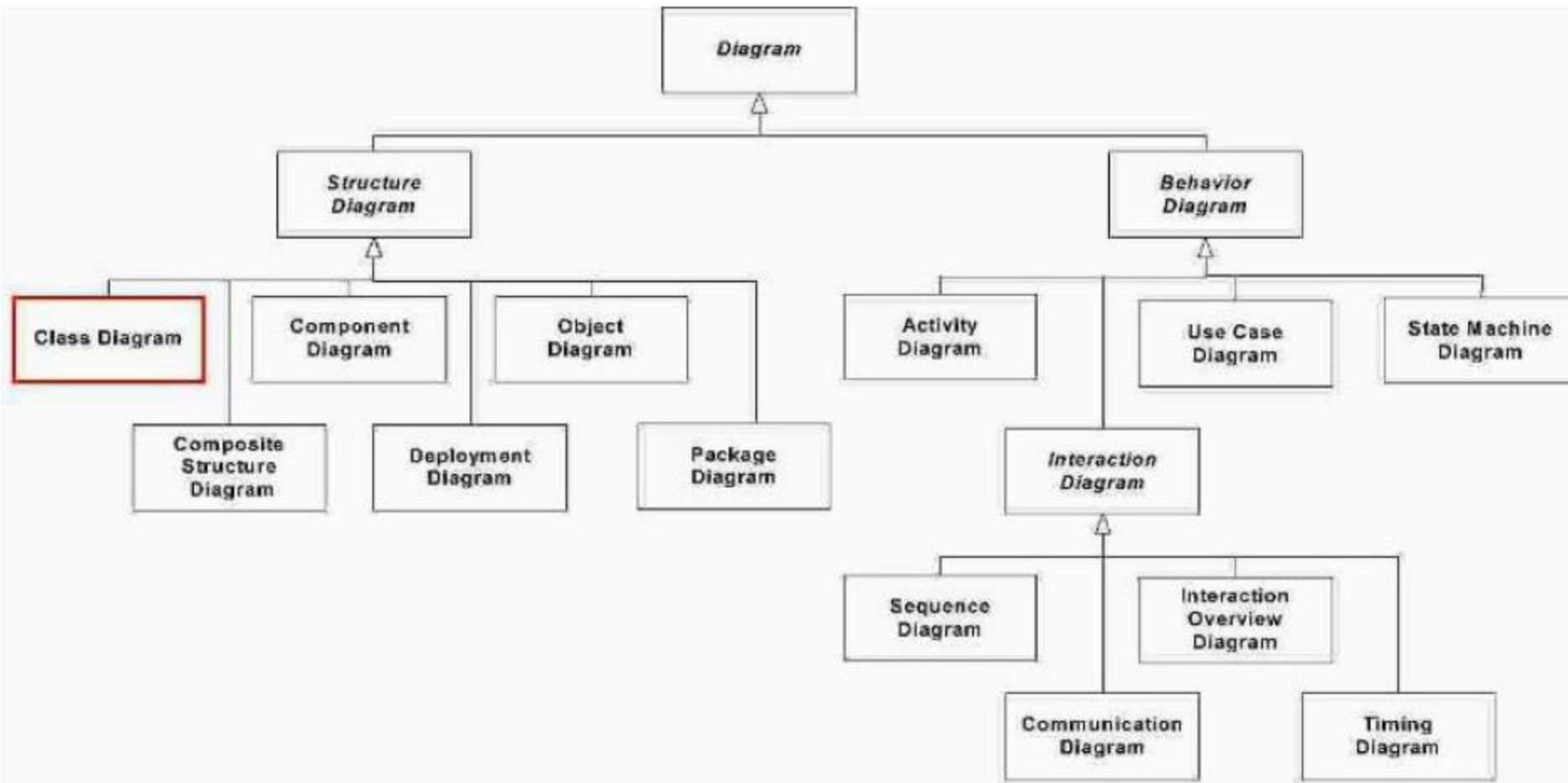
CRC cards: esempio

Class name:	Superclass:	Subclasses:
Responsabilities		Collaborations

CRC cards: esempio

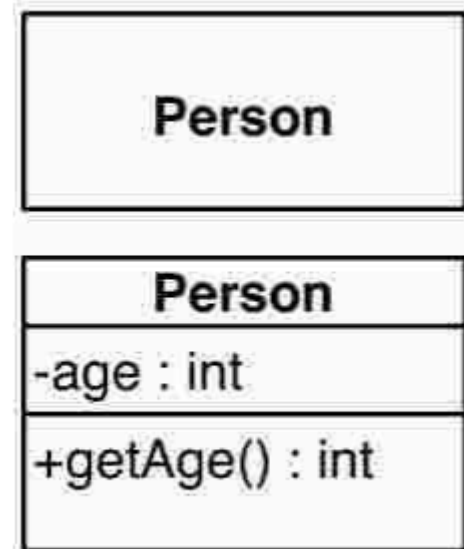
Class name:	Superclass:	Subclasses:
Responsabilities		Collaborations

UML – DIAGRAMMA DELLE CLASSI



L'icona di Classe in UML

- Ha una rappresentazione grafica in forma di un rettangolo diviso in tre parti
- Le classi possono avere fino a 3 slot:
 - nome e l'eventuale stereotipo (in UpperCamelCase - obbligatorio)
 - attributi (in lowerCamelCase - opzionale)
 - operazioni (in lowerCamelCase - opzionale)



Attributi e Operazioni: Signature

Attributo:

visibilità nome molteplicità:tipo=valoreIniziale

Operazione:







visibilità nome (nomeParametro:tipoParametro, . . .):
tipoRestituito

- Solo il nome è obbligatorio
- Le **classi di analisi** solitamente contengono solo quelli più importanti (quelli che risultano evidenti dall'analisi del dominio), e spesso specificano solo il nome.
- Assegnare un valore iniziale in una classe di analisi può evidenziare i vincoli di un problema.
- Le **classi di progettazione** forniscono una specifica completa (implementabile) della classe e dei suoi attributi.

Attributi e Operazioni: Tipi di Visibilità

- + public ogni elemento che può accedere alla classe può anche accedere a ogni suo membro con visibilità pubblica
- private solo le operazioni della classe possono accedere ai membri con visibilità privata
- # protected solo le operazioni appartenenti alla classe o ai suoi discendenti possono accedere ai membri con visibilità protetta
- ~ package ogni elemento nello stesso package della classe (o suo sottopackage annidato) può accedere ai membri della classe con visibilità package

Relazioni tra classi

- Generalizzazione 
- Realizzazione 
- Associazione 
- Dipendenza 
- Aggregazione 
- Composizione 

Associazione



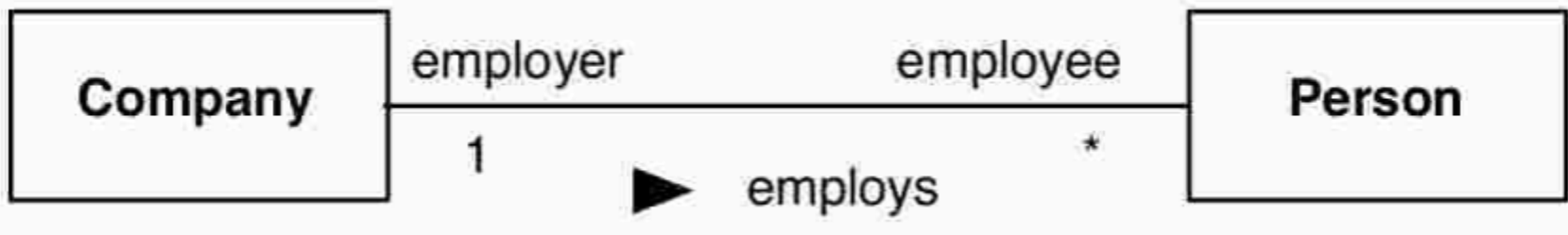
- Si tratta del tipo di relazione più generico: indica solo l'esistenza di collegamenti (link) tra le istanze delle classi.
- Rappresenta l'abilità di un'istanza di mandare messaggi a un'altra istanza.
- Può coinvolgere più di due classi e la stessa classe più di una volta.
- Tra le relazioni è anche la più flessibile e la meno vincolante.

Associazione

Per quanto UML consenta di rappresentare ogni tipo di associazione, spesso nei diagrammi delle classi ci si limita a considerare le sole **associazioni statiche** in cui una classe C1 ha uno o più attributi il cui tipo è la classe C2, oppure una collezione (per esempio un vettore) il cui tipo dei singoli elementi è la classe C2.

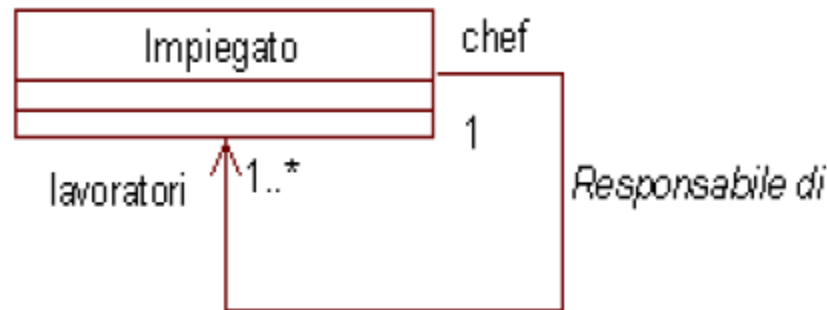
- C1 ha uno o più attributi di tipo C2 (o **collezioni** di oggetti di tipo C2).

Associazione: ornamenti



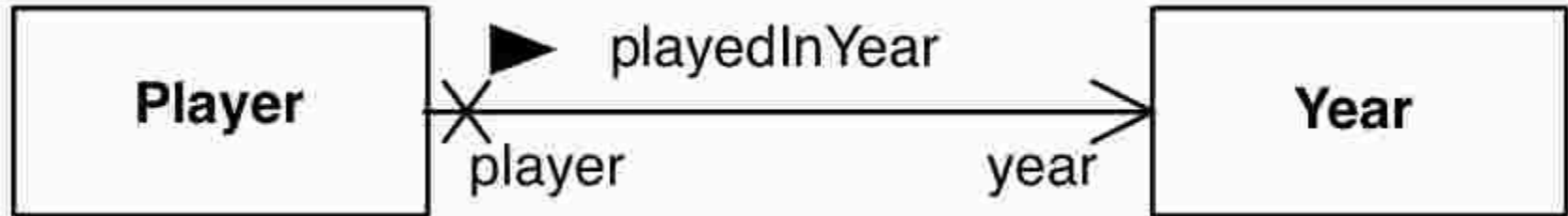
- Nome: opzionale.
- Triangolo direzionale: opzionale. Specifica la direzione in cui leggere l'associazione (aumenta la leggibilità).
- Ruoli: opzionali a ciascun estremo.
- ◆ Quando una classe si associa con un'altra, ognuna di esse gioca un ruolo all'interno dell'associazione. E' possibile mostrare questi ruoli sul diagramma,
- Molteplicità: opzionale a ciascun estremo.

Associazione: ornamenti



- ◆ Una classe può trovarsi in associazione con se stessa. Ciò si verifica quando una classe contiene oggetti che possono giocare svariati ruoli. Tali associazioni sono chiamate: "Associazioni Riflessive".

Associazione: Navigabilità



- Specifica se gli altri estremi dell'associazione possono sapere a quali istanze sono associati.
- La freccia indica navigabilità.
- La croce indica assenza di navigabilità.
- La mancanza di entrambe significa navigabilità non specificata (tipico della fase di analisi).
- Un oggetto di tipo **Player** sa in quali anni ha giocato, un oggetto di tipo **Year** non sa quali giocatori giocarono quell'anno.

Navigabilità: in pratica...

- In pratica, si tende a non specificare la navigabilità di ogni estremo di ogni relazione.
- Un metodo diffuso è il seguente:
 - non si usano le croci
 - l'assenza di frecce indica navigabilità in entrambe le direzioni
 - una freccia indica navigabilità in quella direzione e assenza di navigabilità nell'altra
- Doppia navigabilità risulta indistinguibile da navigabilità non specificata, ma non è un problema in pratica

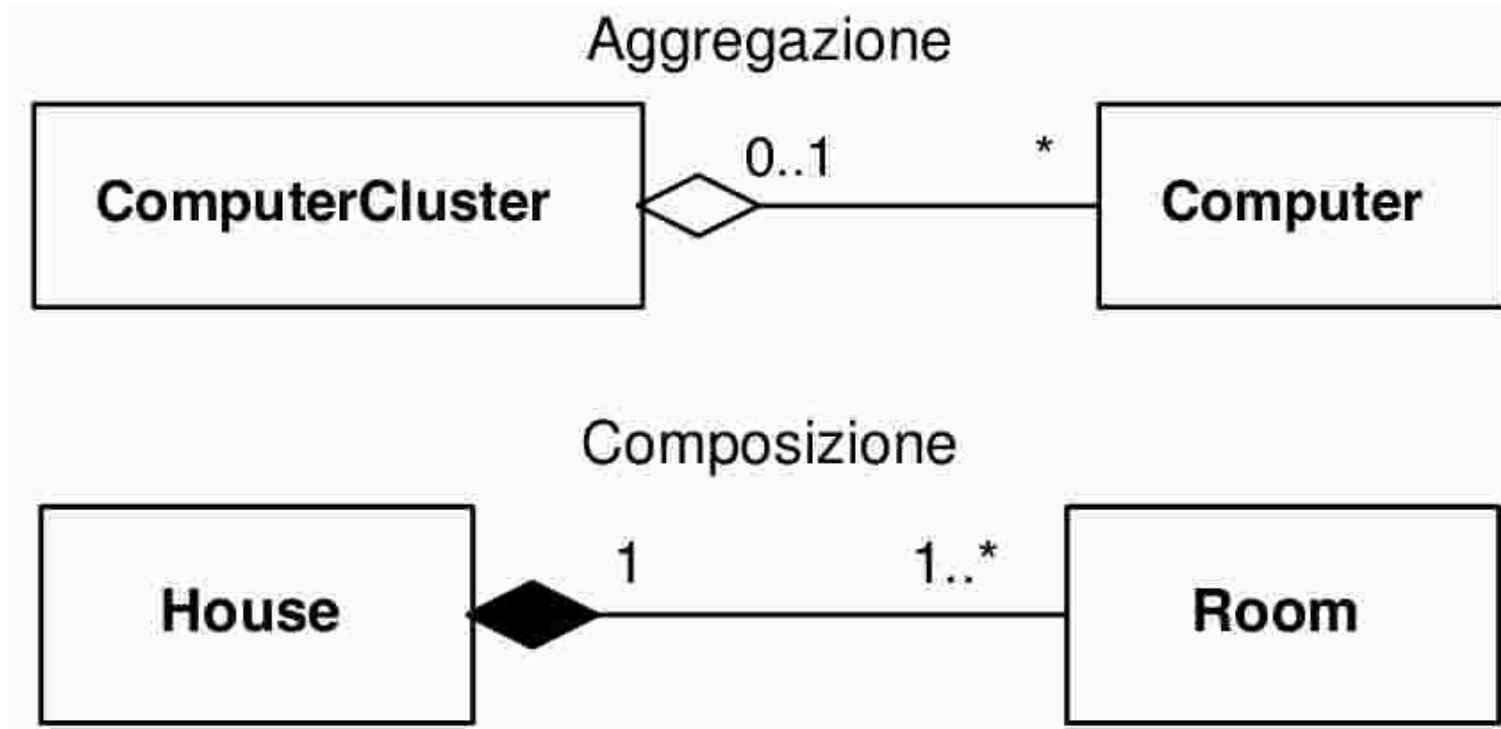
Aggregazione e Composizione

Si tratta di particolari forme di associazione che rappresentano la relazione whole-part (tutto-parte) tra un aggregato e le sue parti.

- **Aggregazione:** relazione poco forte (a. le parti esistono anche senza il tutto; b. Le parti possono appartenere a più aggregazioni. Es. i computer e il loro cluster).
- **Composizione:** relazione molto forte (le parti dipendono dal tutto e non possono esistere al di fuori di esso; es. le stanze e la casa).

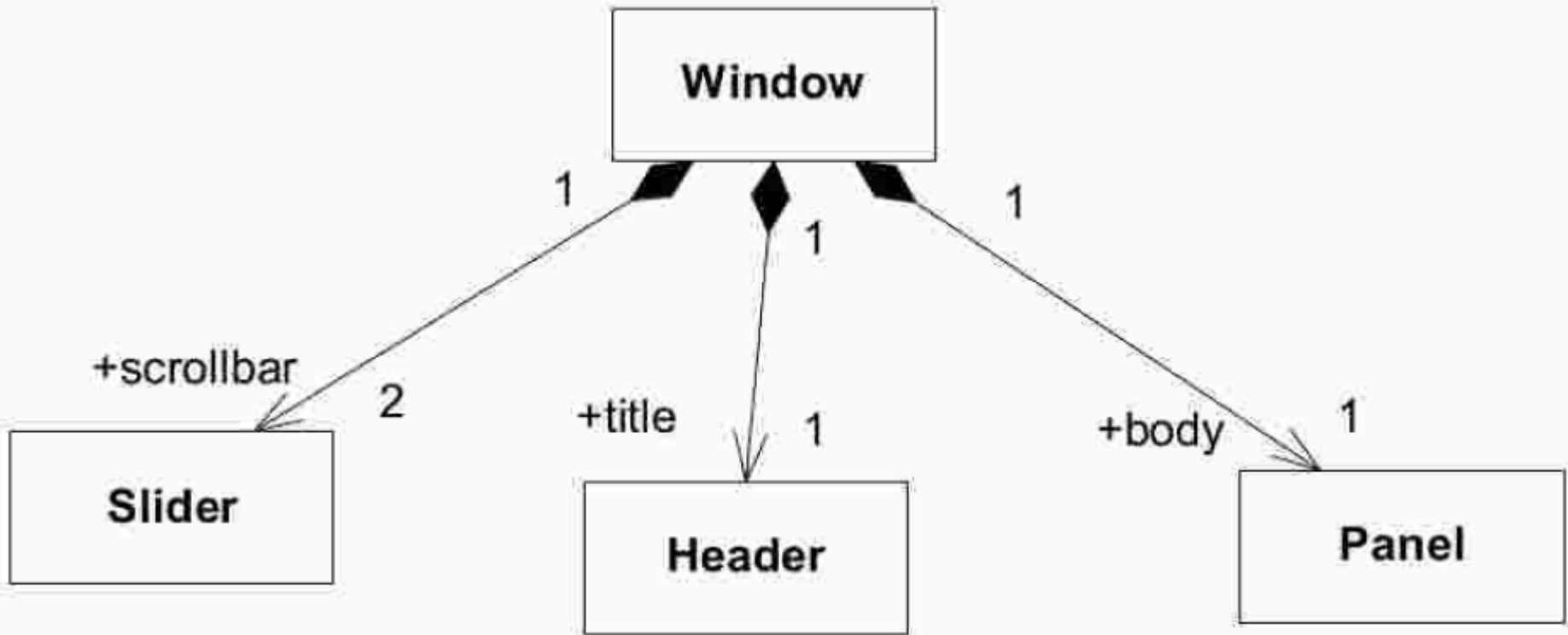
Non è sempre semplice capire quale delle due modelli meglio una situazione.

Aggregazione e Composizione: Notazione 1



Entrambe sono rappresentate da linee con un diamante (pieno o vuoto) vicino alla classe che rappresenta l'intero

Aggregazione e Composizione: Notazione 2



Aggregazione e composizione possono essere combinate con le altre notazioni per le associazioni.

Esercizio

Rappresentare il seguente testo tramite un diagramma UML delle classi:

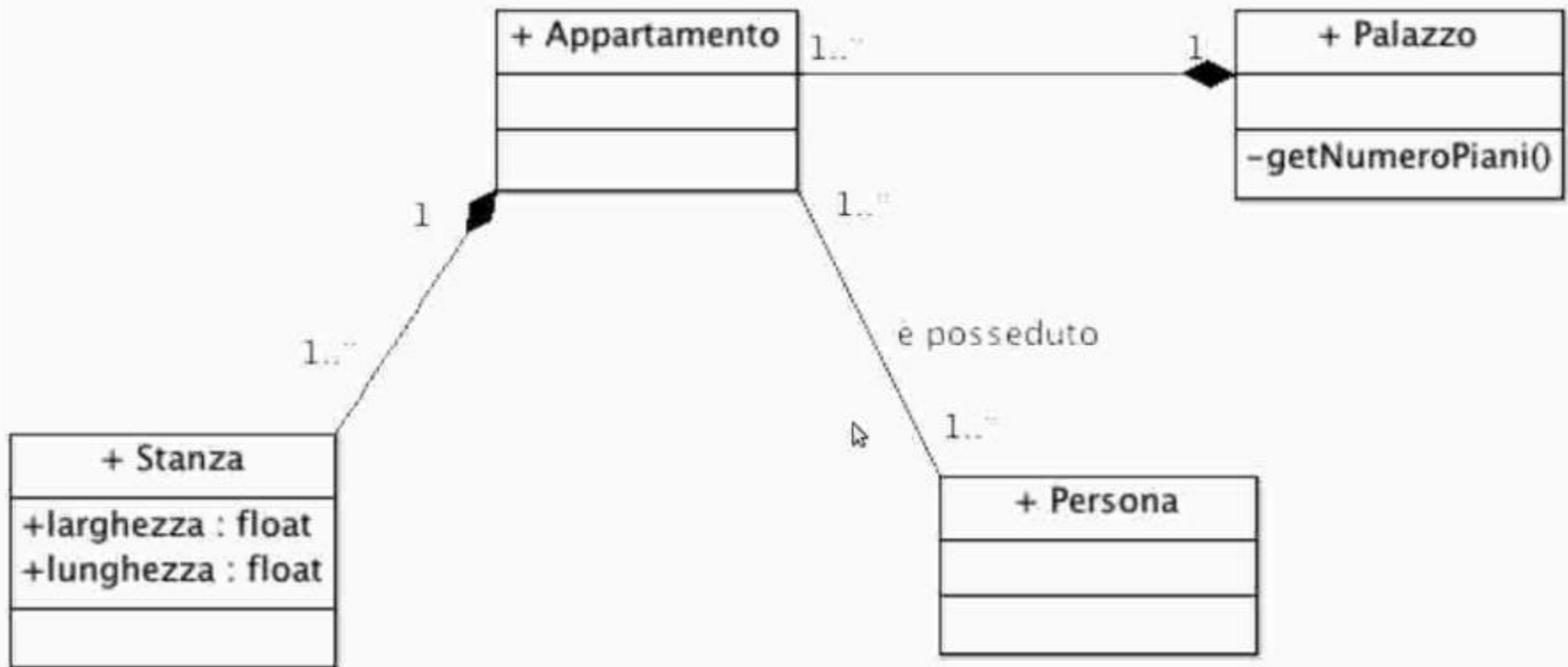
- Un appartamento è composto da una o più stanze, ciascuna delle quali ha una lunghezza e una larghezza. Un appartamento è posseduto da uno o più persone. Un palazzo ha un certo numero di piani ed è composto da uno o più appartamenti.

Esercizio

Rappresentare il seguente testo tramite un diagramma UML delle classi:

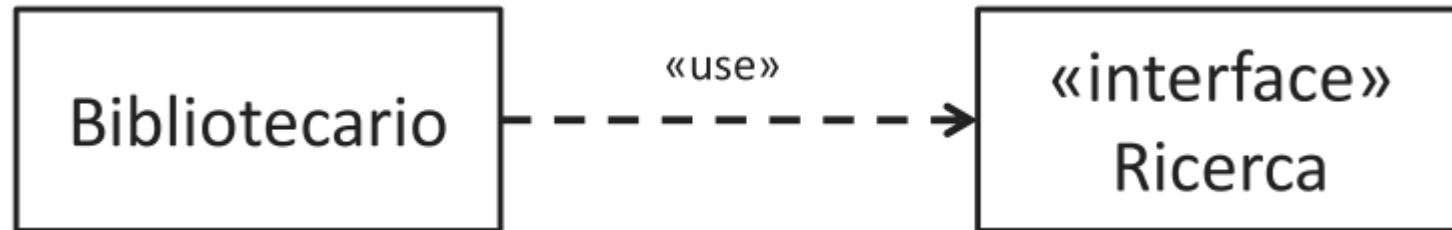
- Un **appartamento** è composto da una o più **stanze**, ciascuna delle quali ha una lunghezza e una larghezza. Un appartamento è posseduto da uno o più **persone**. Un **palazzo** ha un certo numero di **piani** ed è composto da uno o più appartamenti.

Esercizio



- Aggregazione o composizione?
- Aggiungere la classe Piano? Come cambia il diagramma?

Dipendenza



- Una dipendenza è una forma più debole di relazione: tra un cliente ed un fornitore di servizio (es. tra classi e operazioni).
- Due ruoli: **supplier** (fornitore) e **client** (cliente); entrambi possono essere insiemi di elementi.
- La freccia va dal cliente verso il fornitore (e può essere indicato il tipo di dipendenza).
- Una dipendenza significa che il cliente richiede il fornitore per la propria specifica o implementazione.
- Il cliente dipende strutturalmente o semanticamente dal fornitore, e se la specifica del fornitore cambia può cambiare anche quella del cliente.

Generalizzazione

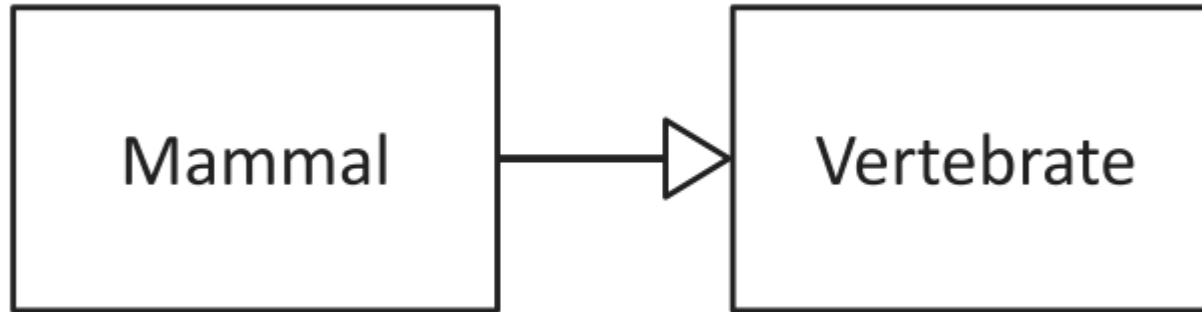
Tutte le associazioni statiche sono relazioni tra classi che possono essere sintetizzate dalla dichiarazione **«gli oggetti di classe C1 HANNO uno o più oggetti di classe C2»**.

La relazione tra classi definita dalla dichiarazione **«gli oggetti di classe C1 SONO anche oggetti di classe C2»** è realizzata dalla generalizzazione che è implementata dai linguaggi di programmazione OO come derivazione di una classe a partire da una classe base (ereditarietà).

La simbologia grafica della relazione di generalizzazione nei diagrammi UML delle classi è:



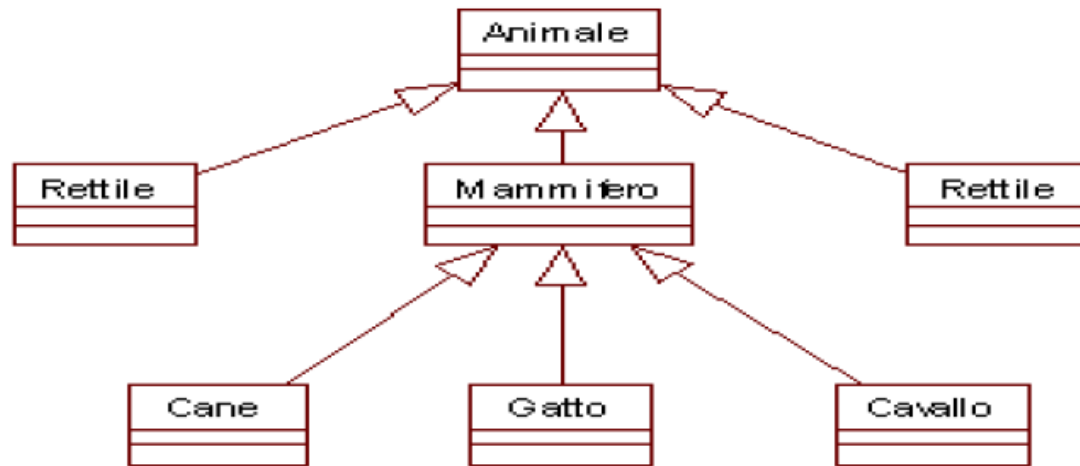
Generalizzazione



- Relazione tassonomica tra un elemento più generale e uno che lo specifica.
- La freccia parte dall'elemento specifico e punta verso quello più generale.
- Si tratta dell'ereditarietà in UML.
- Tra tutte le relazioni, questa è la più forte e vincolante.

Generalizzazione - Ereditarietà

Ereditarietà

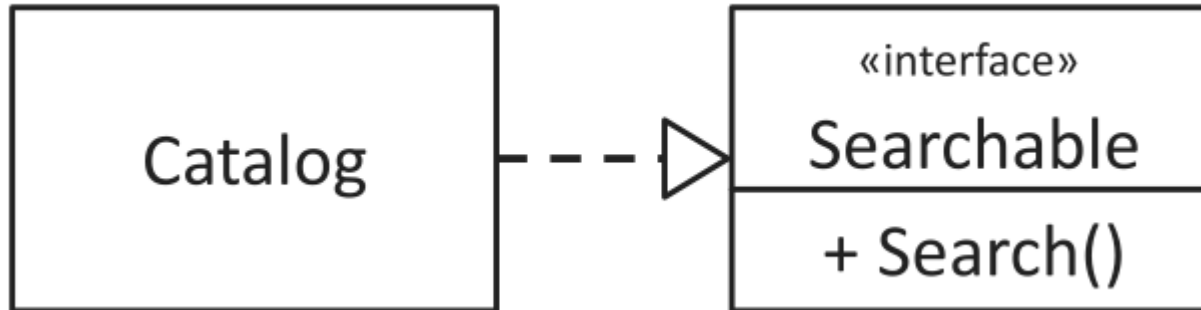


Un esempio di ereditarietà

Una classe figlia (o sottoclasse) può ereditare gli attributi e le operazioni da un'altra classe (che viene definita classe padre o super classe) che sarà sempre più generica della classe figlia.

Nella generalizzazione, una classe figlia può rappresentare un valido sostituto della classe padre. Cioè, in qualunque posto appaia la classe padre, sarebbe possibile far apparire la classe figlia. Il viceversa non è, invece, vero.

Realizzazione



- Si tratta di una relazione semantica in cui il fornitore presenta una specifica, e il cliente la realizza (implementandola e eseguendola).
- L'esempio canonico di realizzazione è quello in cui il fornitore è un'interfaccia, e il cliente è la classe che la implementa.

Diagramma delle classi: esempio

