Azzolini Riccardo 2019-10-21

Class diagram

1 Classi astratte

In UML, le classi astratte possono essere indicate:

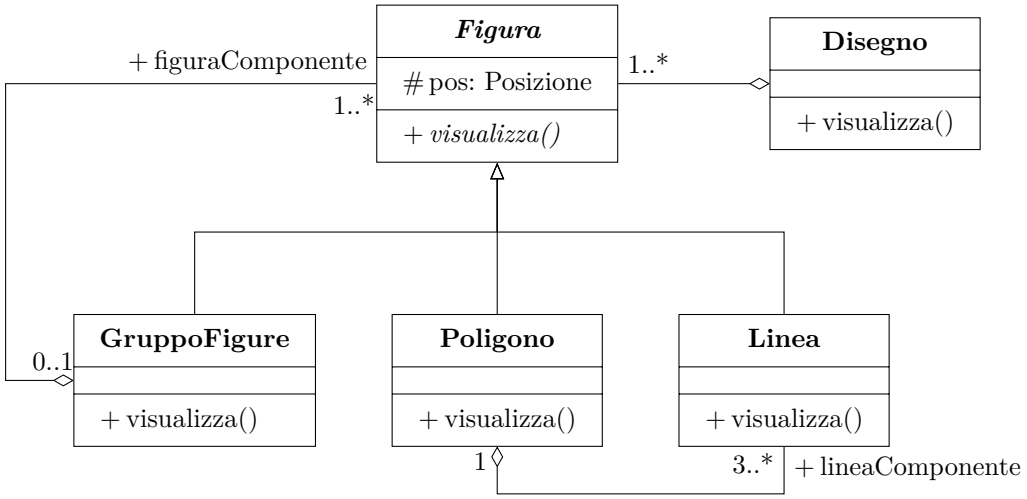
- con lo stereotipo «abstract»;
- scrivendo il nome in corsivo.

Anche le operazioni (metodi) astratte si scrivono in corsivo.

Una classe con almeno un'operazione astratta deve per forza essere astratta, ma può essere utile dichiarare come tale anche una classe che ha solo operazioni concrete, se non ha senso crearne delle istanze (ad esempio, perché è stata introdotta solo per raccogliere delle informazioni che altrimenti sarebbero state ripetute in varie altre classi).

1.1 Esempio e Composite Design Pattern

Si consideri un disegno composto da una o più figure. Per disegnare (visualizzare) se stesso, al disegno non serve sapere di che tipo sono le figure: è sufficiente che esso chiami il metodo "visualizza" di ciascuna figura.

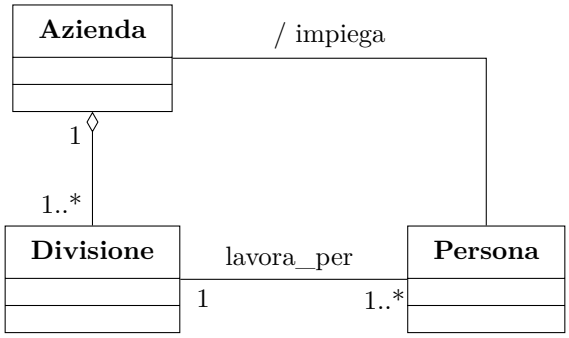


- Il metodo visualizza della classe Figura è astratto, perché ogni tipo di figura deve essere visualizzato in modo diverso.
- Un poligono è formato da 3 o più linee.
- Esiste un tipo particolare di figura, GruppoFigure, che permette di raggruppare altre figure, e quindi di realizzare una struttura ad albero, perché *aggrega più istanze della sua stessa superclasse*. Questa è un'applicazione del **Composite Design Pattern**. Un altro esempio potrebbe essere la rappresentazione di una struttura di file e cartelle.

2 Associazioni derivate

Alcune associazioni sono calcolabili in base ad altri legami e/o ai valori degli attributi. Esse sono chiamate **associazioni derivate**, e si indicano con il simbolo /, come gli attributi derivati.

Ad esempio, nel diagramma seguente, l'associazione "impiega" è calcolabile perché si ricava transitivamente dall'aggregazione e dall'associazione "lavora_per":

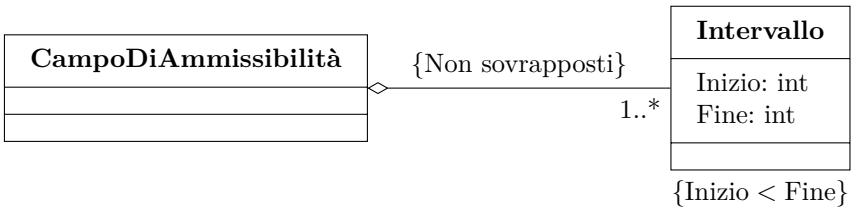


Le associazioni derivate non devono essere segnate sempre (perché ce ne sarebbero troppe), ma solo quando sono importanti per il modello (ad esempio, l'associazione "impiega" potrebbe essere segnata se Azienda avesse bisogno di un elenco generale del personale, oltre agli elenchi per ciascuna divisione).

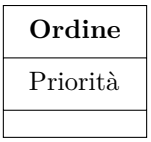
Le associazioni derivate sono di fatto dati duplicati, quindi il codice che le gestisce deve assicurare che esse siano sempre aggiornate al variare dei legami/attributi in base ai quali sono calcolate.

3 Vincoli

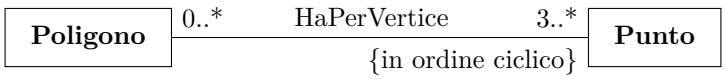
Si possono porre **vincoli** sui valori degli attributi e sulle relazioni, scrivendoli tra parentesi graffe a margine delle classi/associazioni.



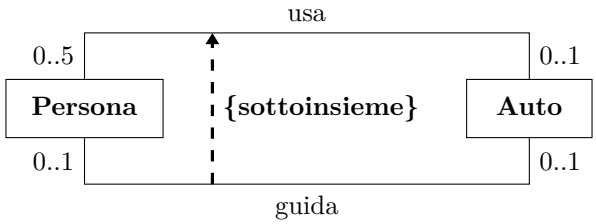
Alcuni vincoli sono specificati da UML, ma è ammesso anche testo libero. I vincoli possono porre restrizioni sia sullo stato di attributi/associazioni in un singolo istante, sia sulla loro evoluzione nel tempo, come ad esempio:

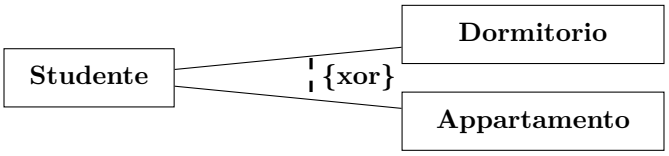


Nelle associazioni "a molti", i vincoli permettono anche di fissare un certo tipo di ordinamento per i "molti" oggetti, che normalmente sarebbero invece considerati come un insieme non ordinato.



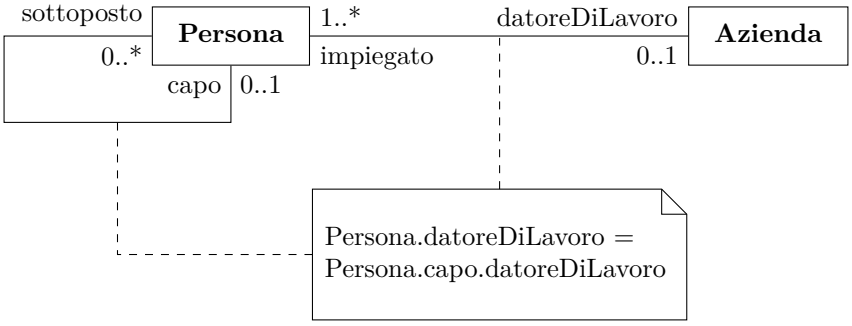
Un altro uso dei vincoli è per condizionare un'associazione rispetto a un'altra. Ad esempio:





- {sottoinsieme} indica che gli oggetti legati dall'associazione "guida" devono essere anche legati dall'associazione "usa" (la Persona che guida un'Auto la sta anche usando);
- {xor} indica che uno Studente non può essere legato sia a un Dormitorio che a un Appartamento.

Infine, alcuni vincoli tra le associazioni si possono esprimere come particolari commenti. Ad esempio:



Questo vincolo, il quale specifica che una Persona e il suo capo lavorano per la stessa Azienda, è scritto in *Object Constraint Language* (OCL), un linguaggio nato per rendere più rigorosa la specifica dei vincoli in UML, che però, in pratica, non ha avuto molto successo. Lo stesso vincolo potrebbe invece essere espresso in linguaggio naturale.