Azzolini Riccardo 2019-10-22

Class diagram

1 Interfacce

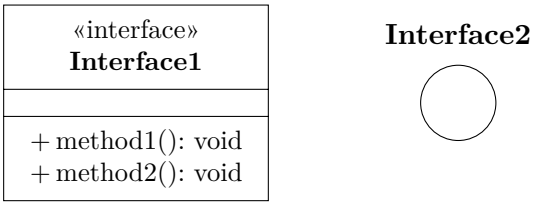
Un'**interfaccia** è una dichiarazione di un set di caratteristiche e obblighi pubblici. Essa specifica un contratto, che deve essere rispettato dalle classi che la implementano.

Un'interfaccia è simile a una classe, ma con alcune restrizioni:

- tutte le operazioni sono pubbliche e astratte (non hanno implementazioni di default);
- tutti i dati sono costanti.

In UML, un'interfaccia si può rappresentare:

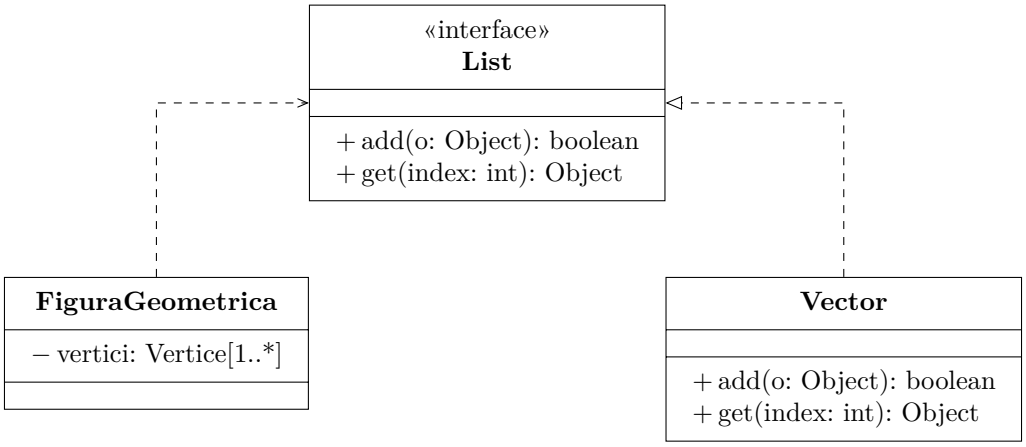
- come classe con lo stereotipo «interface»;
- come cerchio, indicando solo il nome, senza dettagli.



In relazione alle classi, esistono due tipi di interfacce:

- interfaccia **fornita**: la classe che espone l'interfaccia fornisce le operazioni dichiarate da quest'ultima;
- interfaccia **richiesta**: la classe che espone l'interfaccia ha bisogno delle operazioni che questa dichiara per poter svolgere i propri compiti.

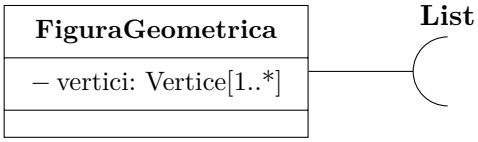
L'interfaccia richiesta corrisponde a una relazione di *dipendenza*, mentre l'interfaccia fornita corrisponde a una relazione di *realizzazione*. Ad esempio:



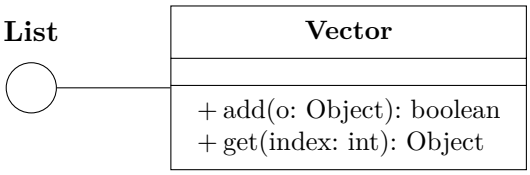
- FiguraGeometrica richiede l'interfaccia List;
- Vector fornisce l'interfaccia List.

Esiste anche una notazione più compatta, "a lecca-lecca":

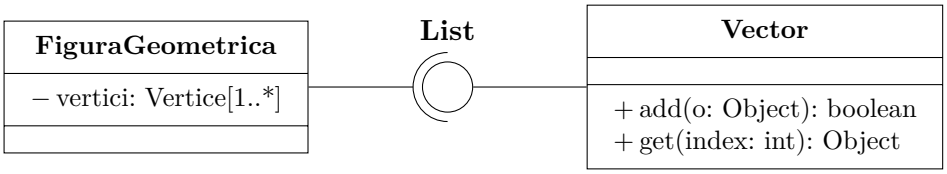
- un'interfaccia richiesta si rappresenta con un semicerchio;



- un'interfaccia fornita si rappresenta con un cerchio;

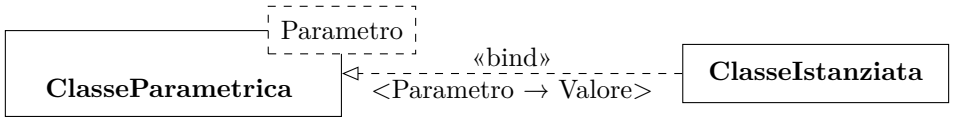


- un'interfaccia richiesta da una classe e fornita da un'altra si rappresenta come un semicerchio "connesso" a un cerchio.

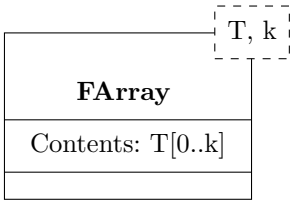


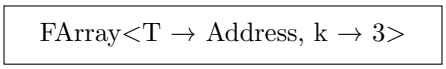
2 Classe parametrica

Una **classe parametrica** (**template**) è caratterizzata da uno o più tipi parametrici. Essa non è in realtà una classe, ma invece definisce una famiglia di classi, che si ottengono istanziando i parametri. Queste classi istanziate (*bound class*) sono legate alla classe parametrica da una relazione di **binding**, raffigurata come una realizzazione con lo stereotipo «bind».

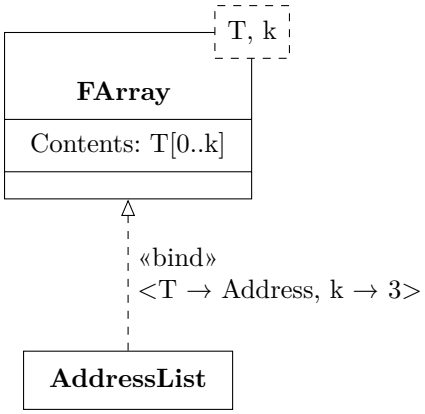


Si possono definire anche classi istanziate anonime. In tal caso, la relazione di binding è implicita. Ad esempio,





equivale a:

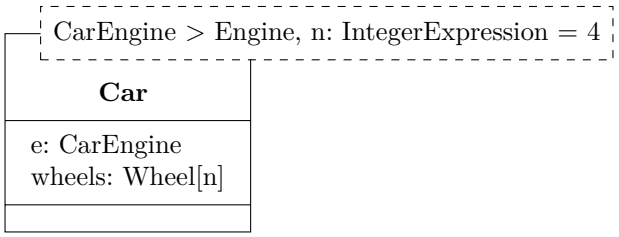


Per i parametri, è possibile specificare:

- un valore di default;
- che i valori assegnati devono essere compatibili con un certo tipo.

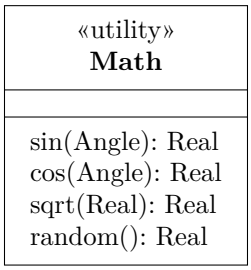
Ad esempio, nel diagramma seguente:

- la classe usata come parametro CarEngine deve essere compatibile con la classe Engine;
- se una classe istanziata non specifica un valore per il parametro n, esso assume 4 come valore di default.



3 Classi utility

Le **classi utility** raggruppano variabili e procedure, che di fatto verranno viste come globali.



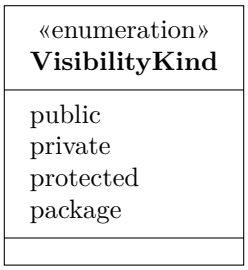
Spesso, queste classi sono "senza memoria":

- i dati sono costanti;
- le operazioni restituiscono sempre gli stessi valori se invocate con gli stessi parametri.[1]

[1] Agli effetti esterni, anche una procedura random si può considerare senza memoria, perché restituisce ogni volta un numero casuale che non è correlato ai precedenti, nonostante abbia internamente uno stato che cambia con ogni chiamata.

4 Enumerazioni

Un'**enumerazione** definisce un elenco di valori. Ad esempio:



5 Stereotipi

Gli **stereotipi** sono uno dei meccanismi per estendere la notazione UML. Essi specializzano il significato di elementi UML predefiniti (classi, associazioni, ecc.: si possono applicare in tutti i tipi di diagrammi). Un elemento stereotipato può essere utilizzato in tutte le situazioni nelle quali può essere usato l'elemento originale.

Uno stereotipo può essere rappresentato in modo testuale, aggiungendo «NomeStereotipo» all'elemento originale, oppure graficamente, usando un simbolo diverso da quello dell'elemento originale.

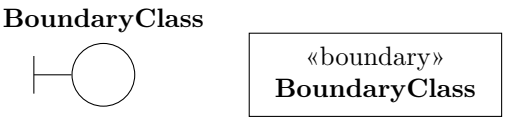
Con gli stereotipi è possibile definire "dialetti" di UML, detti *profili* o *estensioni*, che sono specializzazioni del linguaggio per la modellizzazione di particolari domini applicativi. Gli stereotipi usati in un certo ambito non possono però essere riutilizzati con significati diversi in altre circostanze: essi costituiscono appunto un dialetto, non un linguaggio indipendente.

5.1 Esempio

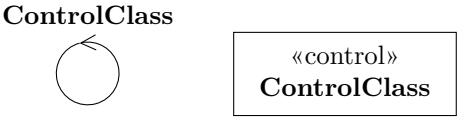
Esistono alcuni stereotipi per rappresentare certi tipi di classi che sono particolarmente comuni in molti sistemi.

Entity: rappresenta i dati. È una classe *passiva*, le cui istanze non sono mai iniziatori di interazioni (cioè non eseguono operazioni "spontaneamente").

Boundary: delimita il sistema, facendo da interfaccia con l'esterno. Tutti gli elementi ancora più esterni rispetto alle classi boundary non sono parti del sistema, e vengono quindi modellati diversamente dai componenti interni al sistema.



Control: controlla le interazioni tra gli altri componenti (entity e boundary).



Questi tre tipi di classi sono i componenti dell'**MVC** (Model-View-Control) Design Pattern.

6 Dipendenza

La relazione di **dipendenza** può esistere tra vari tipi di elementi UML: non solo tra classi, ma anche tra package, use cases, ecc.

In generale, essa individua una connessione "semantica" tra due elementi: una modifica nell'elemento indipendente ha effetto su quello dipendente.

UML definisce alcuni stereotipi che permettono di specificare in modo più preciso la natura di una relazione di dipendenza tra due classi («instantiates», «calls», «friend» e «usage»), ma è comunque possibile utilizzare altri stereotipi "non ufficiali", specializzati per particolari domini applicativi.

7 Annidamento

Una relazione di **annidamento** (**nesting**) indica, in generale, che un elemento UML è definito all'interno di un altro. In particolare, tra classi, l'annidamento indica una classe interna (che solitamente è privata).

L'annidamento si distingue dall'aggregazione perché è una relazione *tra classi*, mentre l'aggregazione è *tra oggetti*: una classe può avere classi interne, mentre un'istanza di una classe può aggregare istanze di altre classi.