

Diagrammi delle classi: concetti avanzati

Lezione 15 di Ingegneria del software

Docente: Giuseppe Scollo

Università di Catania, sede di Comiso (RG)
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Studi in Informatica applicata, AA 2008-9

Indice

1. Diagrammi delle classi: concetti avanzati
2. stereotipi e parole chiave
3. aggregazioni e composizioni
4. proprietà derivate
5. interfacce e classi astratte
6. proprietà di attributi
7. associazioni qualificate
8. classificazione multipla
9. classificazione dinamica
0. classi di associazione
- .1. classi parametriche

stereotipi e parole chiave

notazione per **estensioni standard** di UML

stereotipi e parole chiave hanno simile **sintassi** :

`<<stereotipo>>`, `<<parola chiave>>`, ma talvolta anche `{parola chiave}`

però usi **diversi** :

stereotipi : estensioni per specifici domini applicativi: **profili UML**

parole chiave : estensioni di uso generale, indipendenti dal dominio

perché una sola notazione per le estensioni di UML?

economia di notazioni grafiche ⇒ **apprendibilità**

riuso di un grafismo per concetti semantici diversi ma correlati, ad es.:

`<<interface>>`, `{abstract}` : riuso del grafismo di classe

`<<complete>>`, `<<bind>>` : riuso del grafismo di generalizzazione

aggregazioni e composizioni

sono associazioni che modellano la relazione "parte di" (un "aggregato" o "composto") : qual è la differenza?

molteplicità :

composizione : la "parte" lo è di **un solo** composto (al più)

aggregazione : la "parte" è **condivisibile** da più aggregati

tempo di vita :

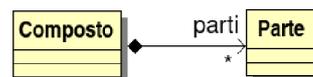
composizione : la "parte" **non sopravvive** al composto

aggregazione : la "parte" **può sopravvivere** all'aggregato...

che peraltro può non essere unico!

notazione :

composizione :



aggregazione :



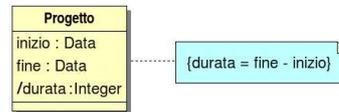
proprietà derivate

proprietà il cui valore può essere **calcolato** in termini dei valori di altre
notazione: **/proprietà --- {vincolo}**

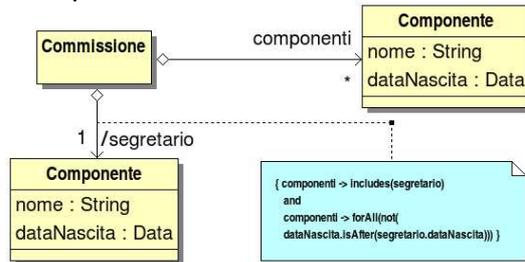
"/" specifica che la proprietà è derivata

l'eventuale {vincolo} specifica come si calcola il suo valore

esempio di attributo derivato :



esempio di associazione derivata :



interfacce e classi astratte

interfaccia : ha solo operazioni astratte (cioè dichiarazioni di operazioni)
notazione : parola chiave **<<interface>>**

classe astratta : può avere operazioni astratte

i metodi di quelle concrete possono essere sovrascritti nelle estensioni

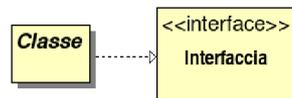
notazione : vincolo **{abstract}** sui nomi della classe e delle operazioni astratte

...

... oppure: nomi in corsivo

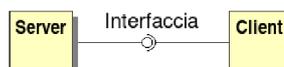
conformità di una classe (astratta o concreta) a un'interfaccia

notazione :



una classe può **fornire** (realizzare, essere **server** di)
o **richiedere** (usare, essere **client** di) un'interfaccia

notazione **lollipop/socket** :



proprietà di attributi

oggetti riferimento e oggetti valore: qual è la differenza?

tutti gli oggetti sono dotati di **identità**, però:

l'eguaglianza di oggetti riferimento è **eguaglianza dell'identità** degli oggetti

devono essere la stessa istanza della classe

l'eguaglianza di oggetti valore è **eguaglianza del valore** degli oggetti

possono essere istanze multiple della classe

gli oggetti valore dovrebbero essere **immutabili** :

il valore non può cambiare nel tempo di vita dell'oggetto

ciò può indicarsi con il vincolo **{frozen}**

il vincolo **{readOnly}** indica invece

una proprietà che non può essere modificata da oggetti client

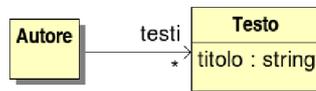
associazioni qualificate

quando una collezione di oggetti è associata a ciascuna istanza, si può usare una loro proprietà per **indicizzare** l'associazione

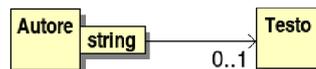
*il tipo di tale proprietà è il **tipo indice**, o **qualificatore**, dell'associazione*

ad esempio, si può qualificare l'associazione di **testi** presenti in una biblioteca ad un **autore** adoperando il **titolo del testo** come qualificatore:

associazione non qualificata :



associazione qualificata :



attenzione al significato della **molteplicità** in un'associazione qualificata:

*fa riferimento al numero di istanze associate per ciascun valore del **qualificatore***

classificazione multipla

classificazione = assegnamento di tipo (a oggetti)

diversa dalla generalizzazione: relazione di sottotipo (fra tipi)

attenzione: la locuzione "è un" si usa in entrambi i sensi!

classificazione multipla : assegnamento di più tipi a un oggetto

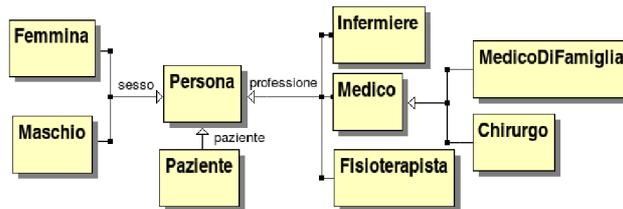
indipendentemente dall'ereditarietà

di solito **non ammessa** nei linguaggi di programmazione OO: ogni oggetto deve avere un tipo, anche se questo ha più supertipi (ereditarietà multipla)

si ha classificazione multipla specializzando un tipo in più modi (generalizzazioni): ogni generalizzazione va etichettata con il nome dell'insieme di generalizzazione

("discriminante" in UML 1) per la deduzione delle combinazioni legali di sottotipi

ad esempio :



classificazione dinamica

possibilità che un oggetto cambi tipo a tempo di esecuzione

- nei fatti un possibile cambiamento di stato
- ad esempio : il mutamento di segno del saldo di un conto corrente può rendere indisponibili alcune operazioni
- notazione : parola chiave `{dynamic}`

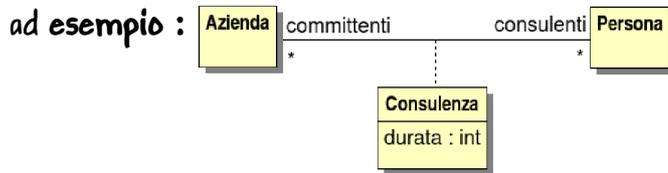
classificazione multipla, dinamica: quando servono?

utili soprattutto nella prospettiva concettuale

classi di associazione

è possibile arricchire un'associazione con **caratteristiche di classe**
attributi, operazioni, etc.

ciò è utile per **caratteristiche non attribuibili a una delle istanze delle due classi ai capi dell'associazione, bensì alla coppia**



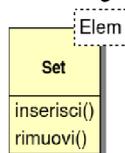
N.B. come **istanza** di una classe di associazione si intende sempre **una sola coppia di istanze** delle classi associate

anche con le molteplicità come nell'esempio in figura

classi parametriche

template di classe : classe **generica** , dotata di un **parametro** (una variabile sui tipi)

notazione :



derivazione di una classe da un template: `Set<Elem::Persona>`

oppure, con **ridenominazione del tipo derivato**:

la derivazione è una **specializzazione ristretta** :

*non si possono aggiungere caratteristiche al tipo
si può solo legare il parametro a un tipo specificato*

