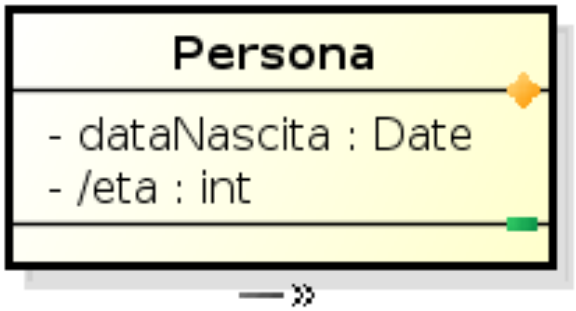
Lezione #12

Lunedì, 04 Novembre 2013

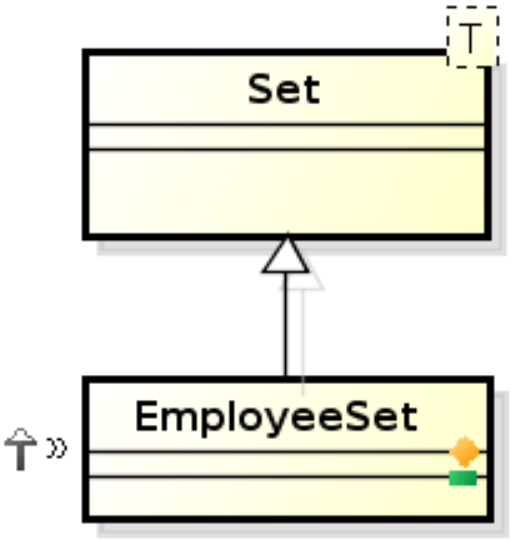
Christian Cardin, 09:30-11:15

Luca De Franceschi

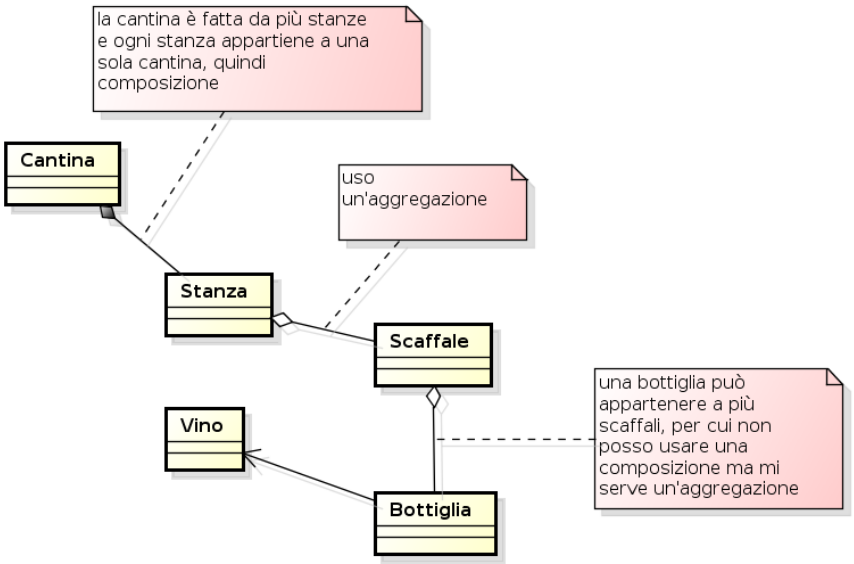
Caratteristiche varie sui diagrammi delle classi.
Attributi o operazioni statiche, non associati ad un oggetto ma al tipo; tutte le istanze la condividono. In UML basta sottolineare l'attributo o l'operazione. L'utilizzo di variabili e metodi statici va normalmente evitato, perchè si rischia di tornare alla programmazione procedurale.
Inserimento di un sacco di **parole chiave**, come `<<interface>>` o {`abstract`} o `<<enumeration>>`. I blocchi di una classe in verità sono più di 3, un blocco per le **eccezioni** e informazioni aggiuntive sulle responsabilità (commenti). Ma se la classe cambia spesso non ha senso aggiungere queste ultime due, altrimenti avrò problemi in fase di manutenzione. La documentazione deve restare il più possibile allineata con il *codice sorgente*.
Si possono inserire delle proprietà derivate nei diagrammi delle classi.



con la *slash* prima di età dico che età dipende dalla data di nascita. In questo modo evito di scrivere un campo dati in più e quindi scrivo meno codice.
Proprietà **read-only**, non deve fornire i servizi di scrittura su quella classe (niente metodi *setter*); **frozen** significa che è una costante, l'attributo non può essere modificato.
I linguaggi di programmazione forniscono i *template*, classi parametriche. In UML si rappresentano con un rettangolo con bordo tratteggiato in alto a destra della classe.

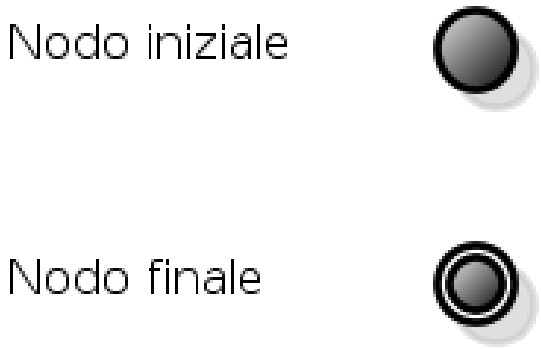


Le **classi attive** sono classi che possono essere eseguite (in Java le classi che estendono *Thread*).
I diagrammi delle classi sono tra i più ricchi in assoluto, sono stati arricchiti sempre di più, noi ne abbiamo visto solo una parte. Meglio prima partire con diagrammi semplici per prendere confidenza.
I **diagrammi a oggetti** sono molto simili, fotografano il software in un certo istante.
Esercizio: cantina formata da più locali, ogni stanza ha scaffali e su questi ci sono diverse bottiglie. Le bottiglie vengono portate da una stanza all'altra ma vogliamo mantenere uno storico di dove è stata quella bottiglia.

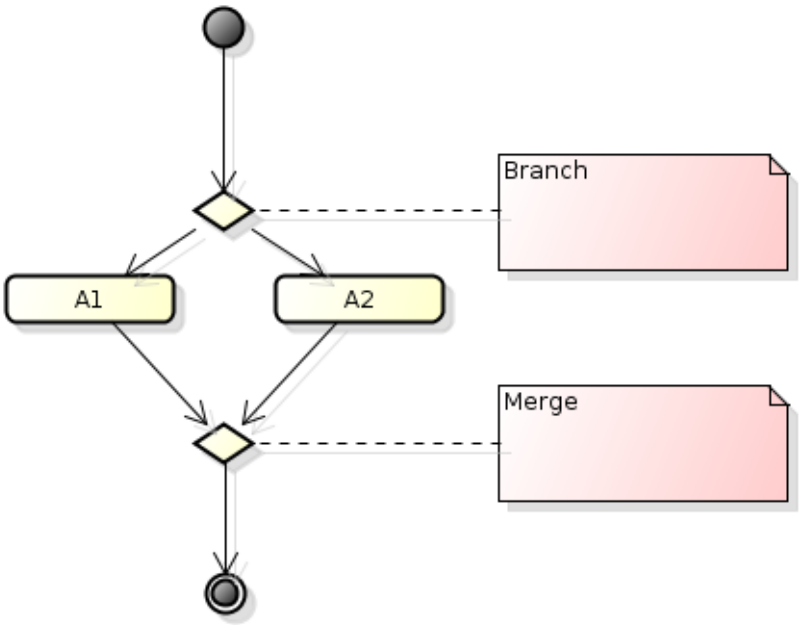


Diagrammi delle attività

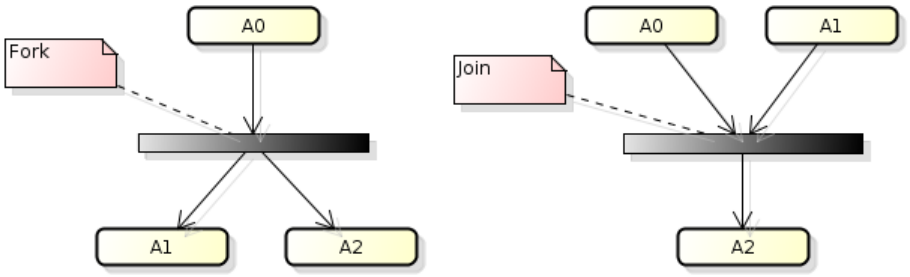
Finora abbiamo visto il punto dell'analista (*casi d'uso*) e quello del progettista (*classi*). Non abbiamo ancora nessuna tipologia di diagramma che mi permetta di descrivere gli aspetti dinamici del software. Li andremo a scoprire con i **diagrammi delle attività**. Essi vengono usati sia nella prima parte che nella seconda. Assomiglia molto ad un diagramma di flusso, è una serie di azioni che devono essere effettuate per un processo. Ciascun processo ha un **nodo iniziale** e un **nodo finale**.



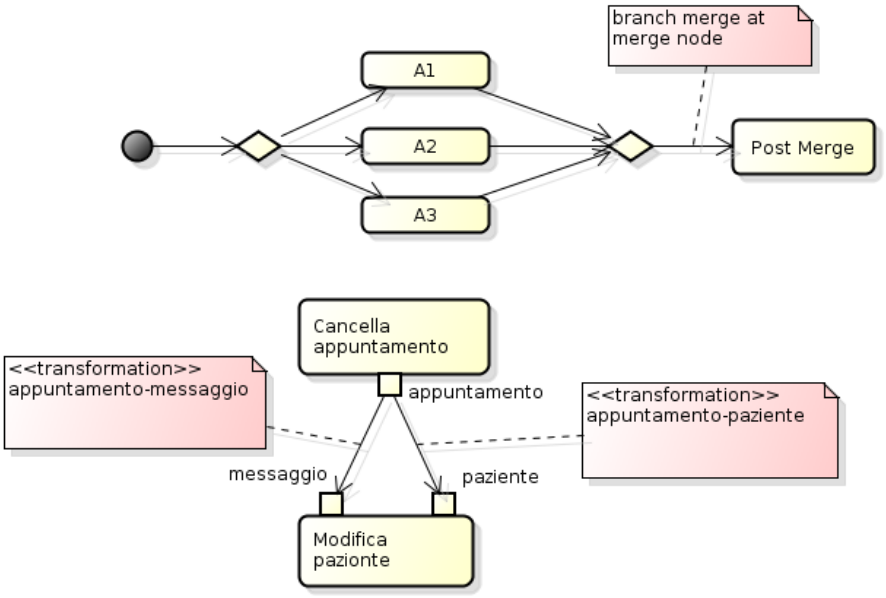
Da questi due nodi dobbiamo costruire un percorso per andare dallo stato iniziale a quello finale. Questo percorso descrive come l'attività debba essere eseguita. Ogni azione è descritta da un rettangolo con i bordi smussati e ha una descrizione interna. Una decisione, o **branch**, serve per modellare le condizioni (gli *if*). La condizione si chiama **guardia**. L'insieme dei rami deve dare la totalità delle condizioni. Quando due o più branch si riuniscono abbiamo un **merge** che raggruppa il flusso.



Questo diagramma mi permette di modellare anche il parallelismo. Questa fase parallela è detta **fork**. Il fork sdoppia un token in tante azioni che vengono eseguite in parallelo, o in sequenza se non ci interessa l'ordine di esecuzione dei rami. Per ricollegare più flussi paralleli usiamo la **join**.



La join potrebbe avere delle espressioni booleane che consentono di proseguire solo se queste condizioni sono soddisfatte.
Può succedere che uno dei flussi, per qualche condizione, **muoia**, ovvero non abbia delle condizioni di terminazione. In questo caso non termina l'intera attività ma solo quel ramo.
Può esserci il caso, soprattutto all'inizio, in cui ho bisogno di definire delle **sottoattività** che saranno un'implementazione di un'azione (farla vedere al dettaglio). In un diagramma di attività, tra un'azione e un altro passo passano degli oggetti. Un'azione prende un oggetto in input e restituisce un oggetto in output.



Le **partizioni** forniscono una responsabilità all'esecuzione delle azioni.