

Diagrammi delle classi

Class diagrams

Diagrammi degli oggetti

Object diagrams

Diagramma delle classi

E' un diagramma che illustra una collezione di elementi dichiarativi (**statici**) di un modello come classi e tipi, assieme ai loro contenuti e alle loro relazioni.

Serve per individuare gli elementi di un sistema

Costruito, perfezionato ed utilizzato durante **tutto** il processo di sviluppo del sistema

Scopo:

- individua e specifica i concetti del sistema
- specifica le collaborazioni
- specifica gli schemi logici dei D.B.

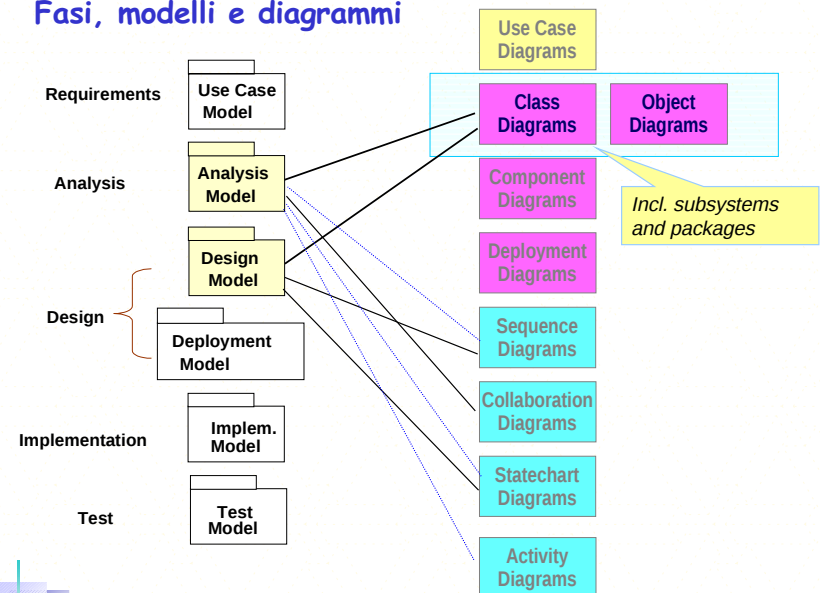
Utilizzato dagli analisti, progettisti e dai programmatori

Diagrammi delle classi e degli oggetti

Un **diagramma delle classi** è un grafo composto da classi e relazioni.

Un **diagramma degli oggetti** è un grafo composto da istanze di classi (oggetti) e relazioni; esso è *una istanza* del diagramma delle classi.

Fasi, modelli e diagrammi

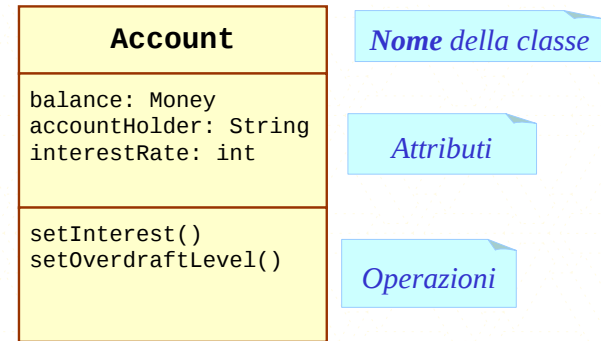


Classe

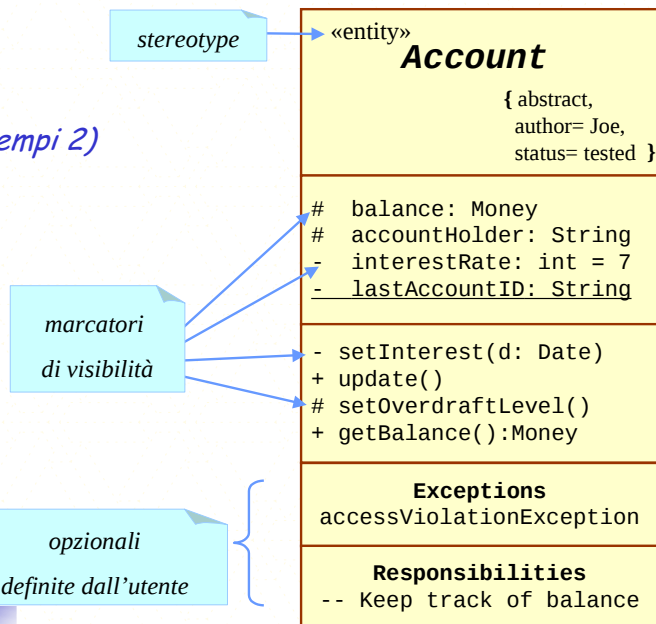
**“A class is the descriptor
for a set of objects with
similar structure, behavior,
and relationships”.**

dal manuale di riferimento di UML v.1.5

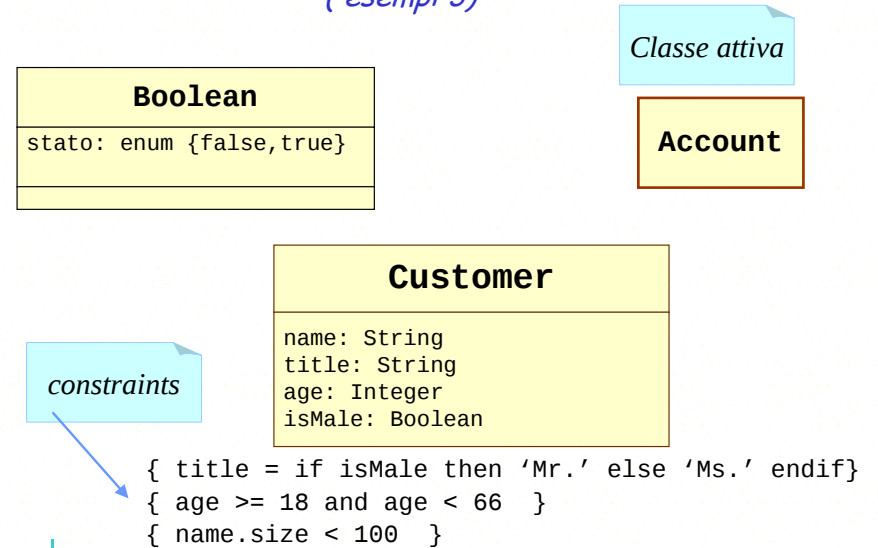
Sintassi di una classe (esempi 1)



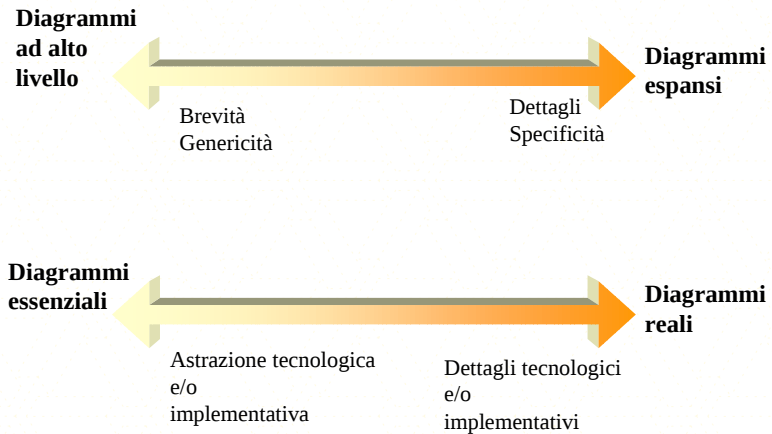
(esempi 2)



(esempi 3)



Livello di dettaglio nei diagrammi



Sintassi membri (in diagrammi espansi o reali)

Attributi

[visibility] name [multiplicity] [: type]
[= initial-value] [{property-string}]

changeable, addOnly, frozen

Operazioni (implementate da metodi, ...)

[visibility] name [(parameter-list)] [: return-type]
[{property-string}]

leaf, query, sequential, guarded, concurrent

parameter-list

[direction] name : type [= default-value]

direction

in, out, inout.

default

Marcatori di visibilità

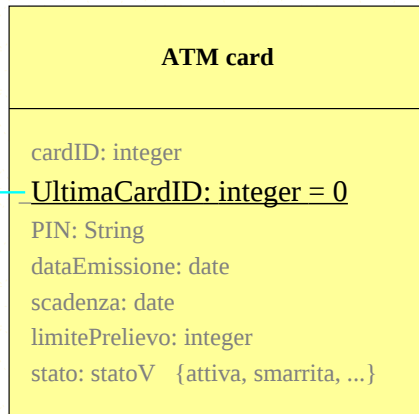
- +** *public* visibility
- *private* visibility
- #** *protected* visibility
- ~** *package* visibility

Altri marcatori

- /** *Derived*
- \$** *Static* (non standard
preferibile la sottolineatura)
- *** *Abstract* (non standard)

class-scope - instance-scope

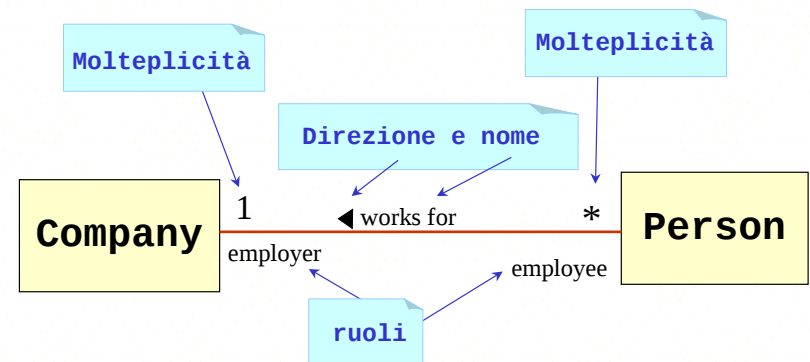
class-scope
(static)



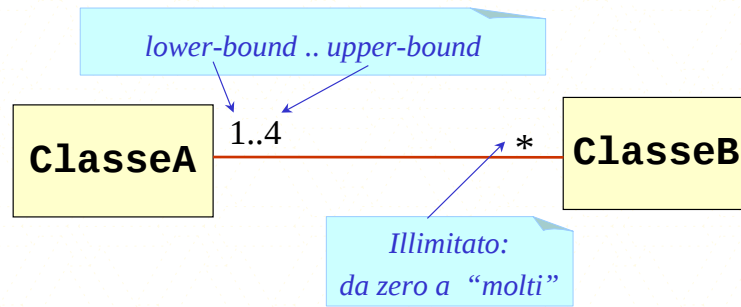
Relazioni (principali) tra le classi

- Associazioni (*association*) sono relazioni strutturali
- Generalizzazioni (*generalization*) sono relazioni di ereditarietà.
- Composizione e Aggregazione (*composition and aggregation*) speciali relazioni strutturali
- Dipendenza (*dependency*)
- Realizzazioni (*realization*) una relazione tra una specificazione e la sua implementazione

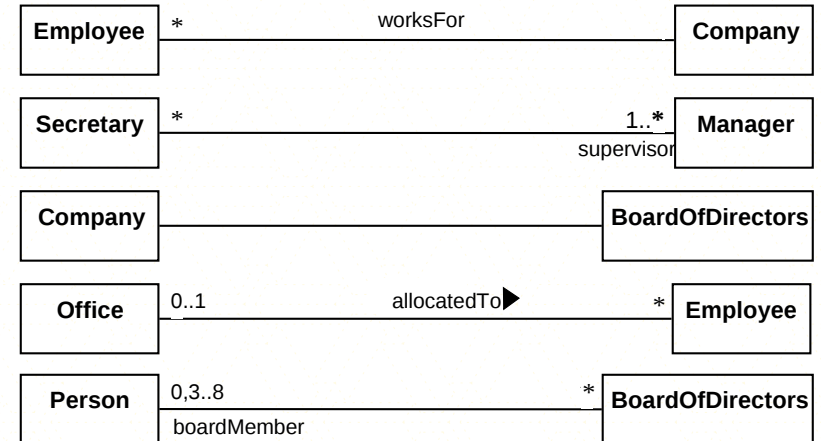
Associazioni



Molteplicità nelle associazioni (1)

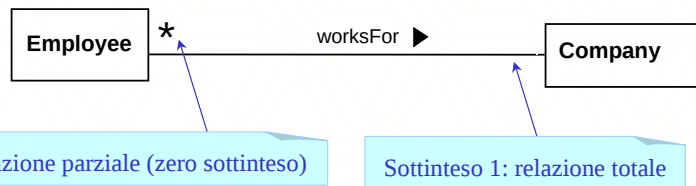


Molteplicità nelle associazioni (2)

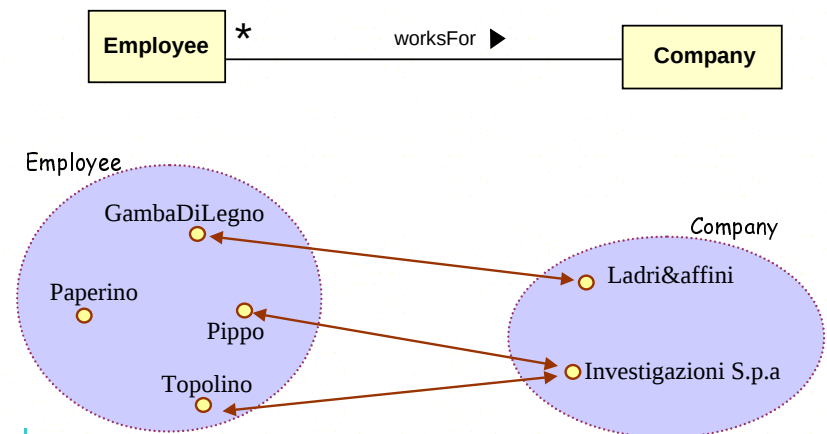


Molti-a-uno (Many-to-one)

- Una società ha molti dipendenti
- Un dipendente può lavorare solo per una società.
- Una società può avere zero dipendenti
- Non è possibile per un dipendente non "dipendere" da una società

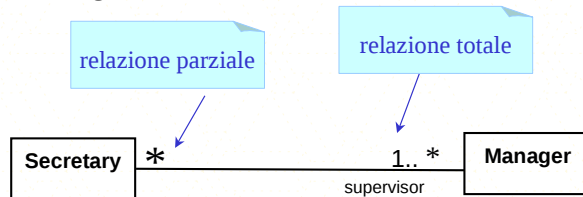


Molti-a-uno (significato con istanze)



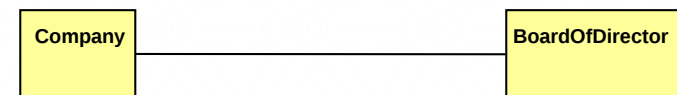
Molti-a-molti (Many-to-many)

- Una segretaria può lavorare per molti dirigenti
- Un dirigente può avere molte segretarie
- Alcuni dirigenti potrebbero avere nessuna segretaria.
- Una segretaria deve per forza avere almeno un dirigente



Uno-a-uno (One-to-one)

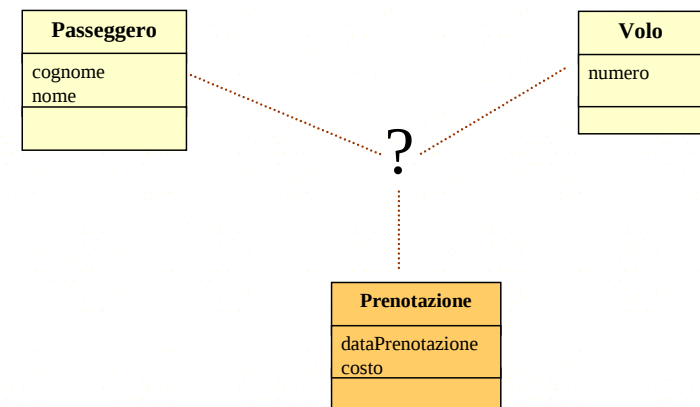
- For each company, there is exactly one board of directors
- A board is the board of only one company
- A company must always have a board
- A board must always be of some company



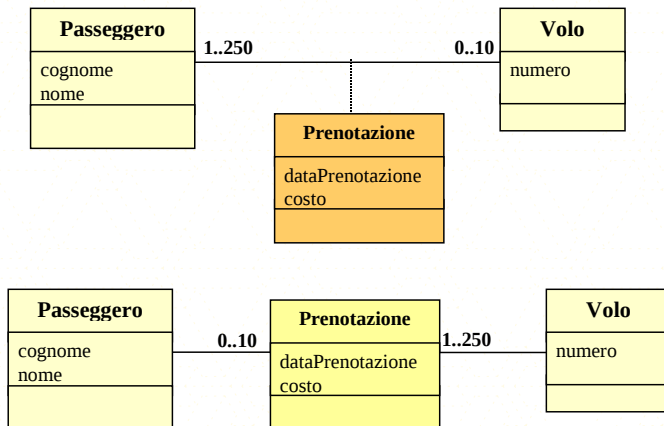
Esempio: voli e prenotazioni

- A booking is always for exactly one passenger
 - no booking with zero passengers
 - a booking could *never* involve more than one passenger.
- A Passenger can have ten Bookings
 - a passenger could have no bookings at all
- A Flight can have 250 Bookings
- A booking have a cost and a date

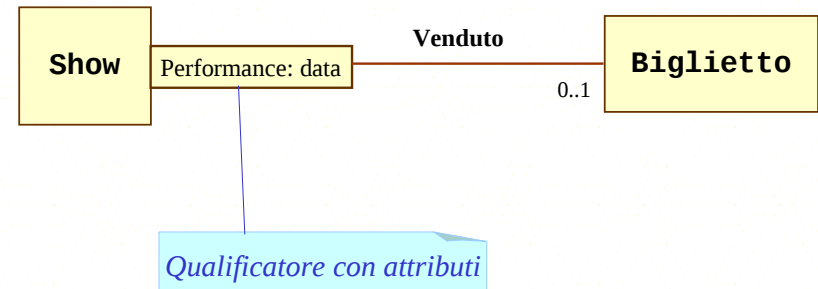
Voli e prenotazioni: soluzione ?



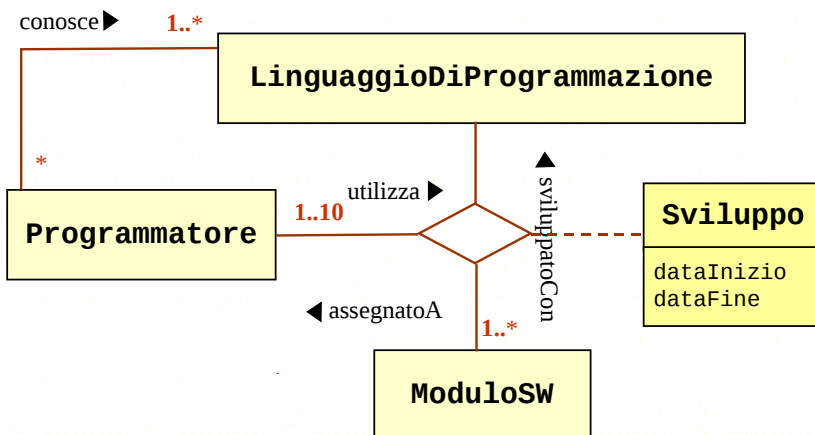
Classe d'associazione (association class)



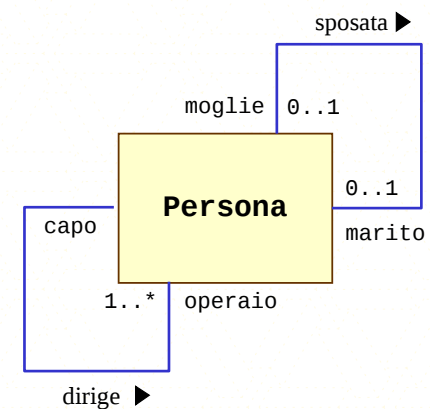
Associazioni qualificate (qualified associations)



Associazione ternaria con classe d'associazione

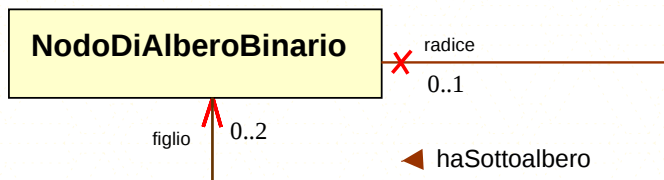


Associazioni ricorsive (o riflesse)



Navigabilità nelle associazioni

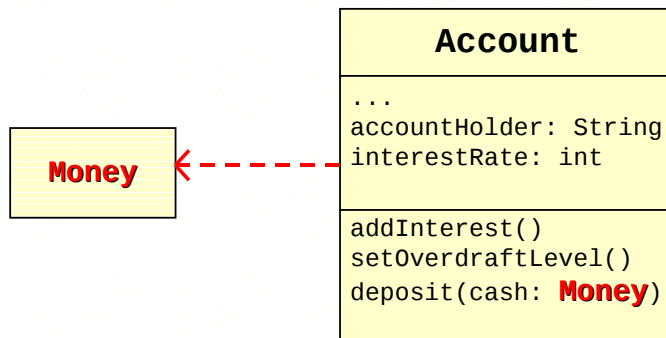
- Le associazioni sono per default con “*direzione non specificata*” per i diagrammi ad alto livello o essenziali.
- E’ possibile indicare la direzione di una associazione (navigabilità) aggiungendo una freccia alla estremità della linea.
- Se si vuol indicare l’impossibilità della navigazione in una direzione si inserisce un simbolo di blocco



Dipendenze (dependency)

Dipendenze (dependency)

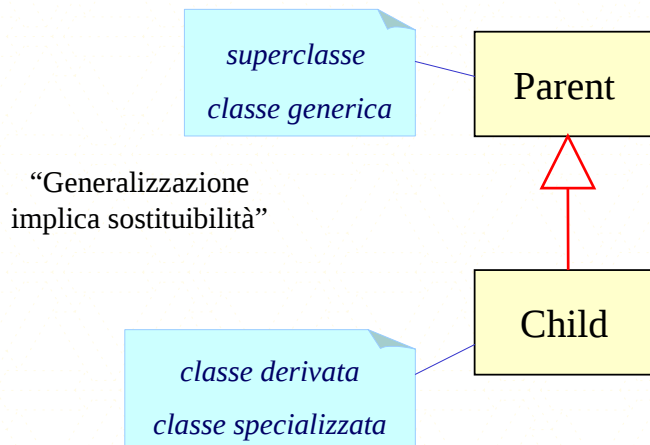
Una dipendenza mostra che una classe **usa** un'altra classe. Un cambiamento nella classe indipendente influirà l'altra.



Generalizzazione

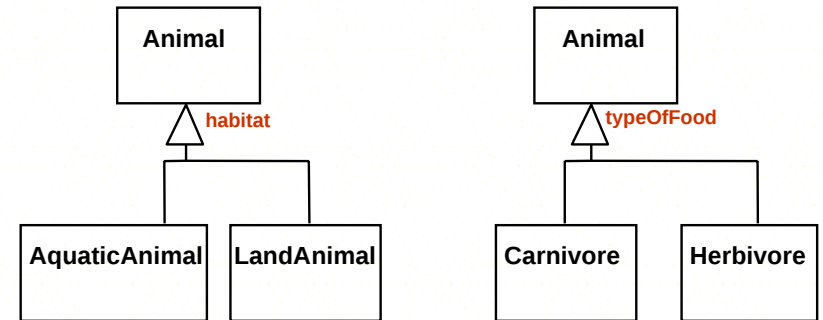
Generalizzazione (specializzazione, ereditarietà) (1)

Una relazione tassonomica tra un elemento più generale ed uno più specifico

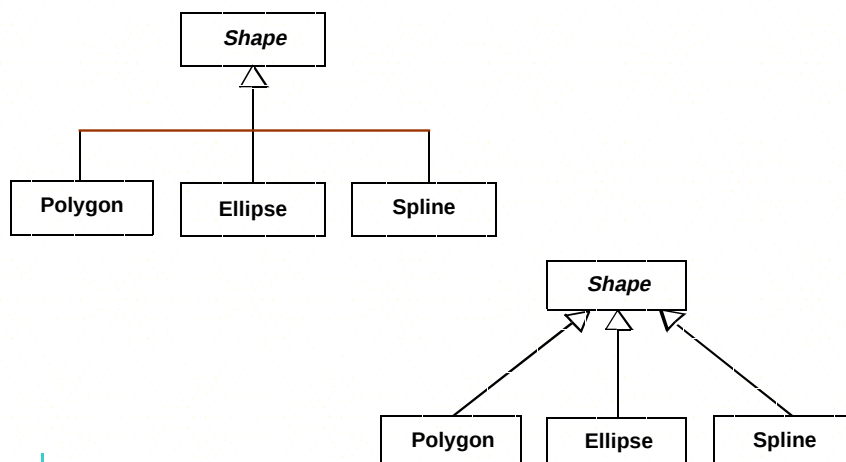


Specializzazione (2)

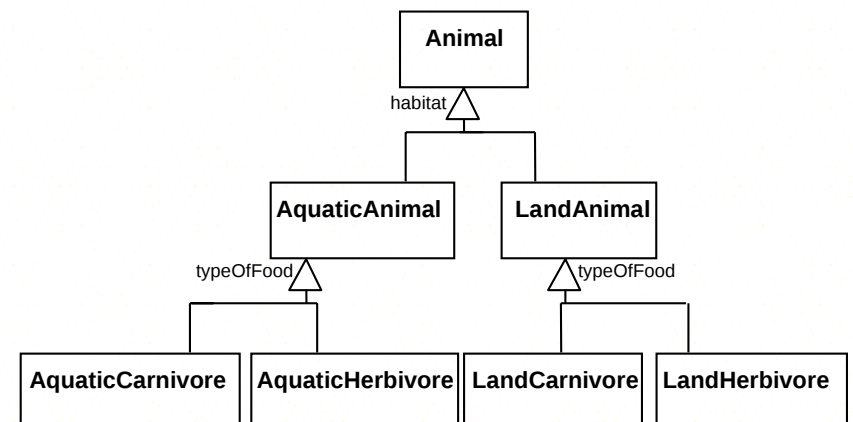
- Il **discriminatore** (*discriminator*) è una etichetta che descrive il criterio utilizzato per la specializzazione.



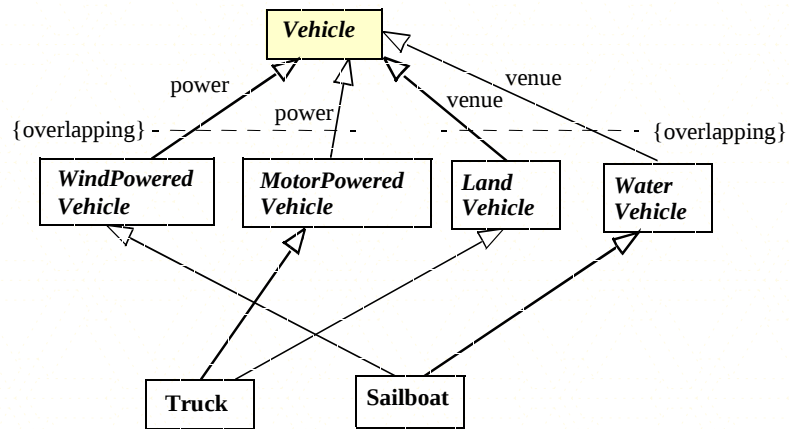
Specializzazione: stili grafici



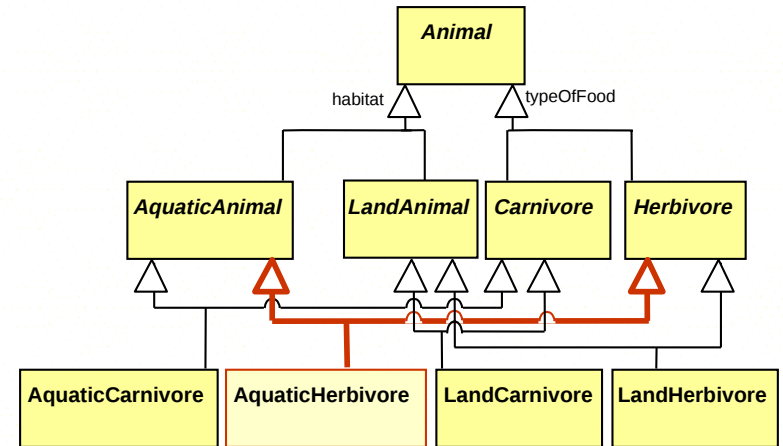
Discriminatori multipli



Ereditarietà multipla (1)



Ereditarietà multipla (2)



Classi astratte, tipi, interfacce

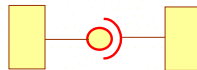
Abstract classes, Types, Interfaces

«type»
SomeType

I tipi non hanno implementazione
Usati per i "built in types" come `int`.

«interface»
SomeFace

Stabiliscono un **contratto** (un obbligo) col cliente
Le interfacce hanno solo dichiarazioni pubbliche



SomeClass
{abstract}

Le classi astratte non hanno istanze
(Le classi astratte pure C++ sono simili alle interfacce Java)

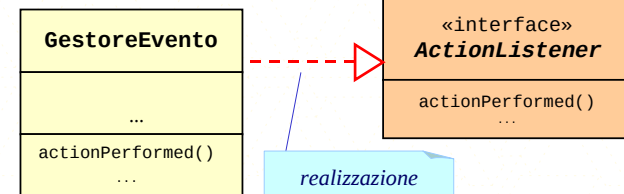
Il nome della classe in corsivo qualifica la classe astratta

«abstract»
SomeClass

Interfacce

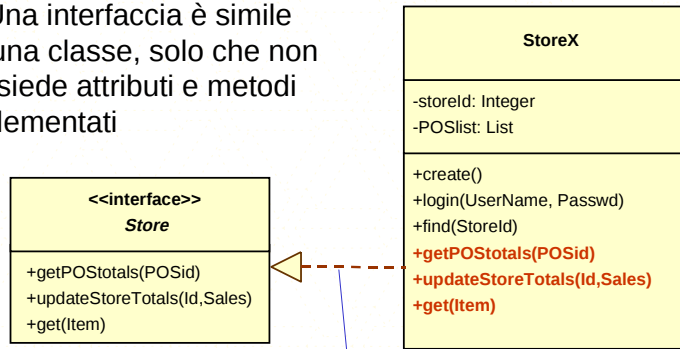
GestoreEvento

ActionListener



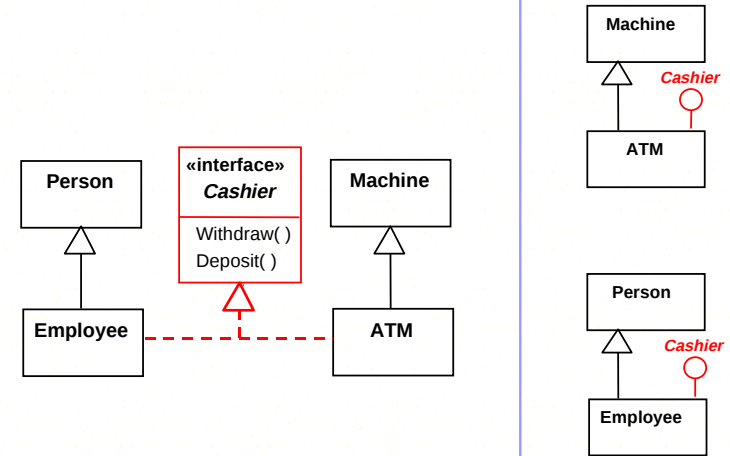
Interface

- Una interfaccia descrive una porzione del comportamento visibile di un insieme di oggetti
- Una interfaccia è simile ad una classe, solo che non possiede attributi e metodi implementati

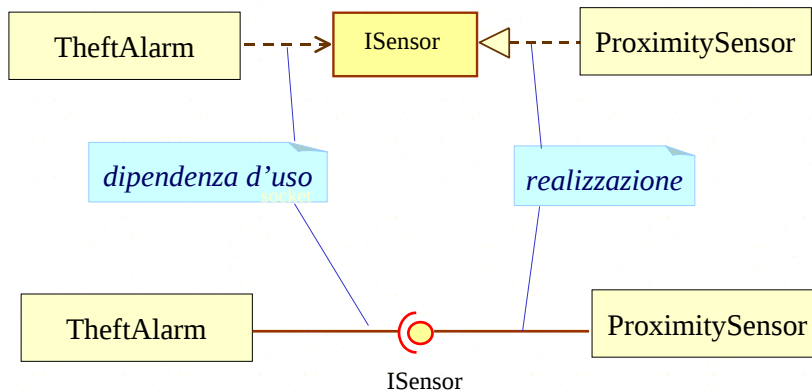


realizzazione

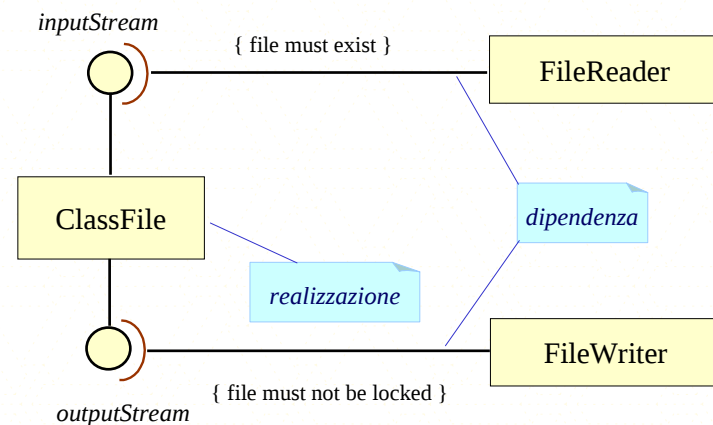
Interfacce: notazioni equivalenti



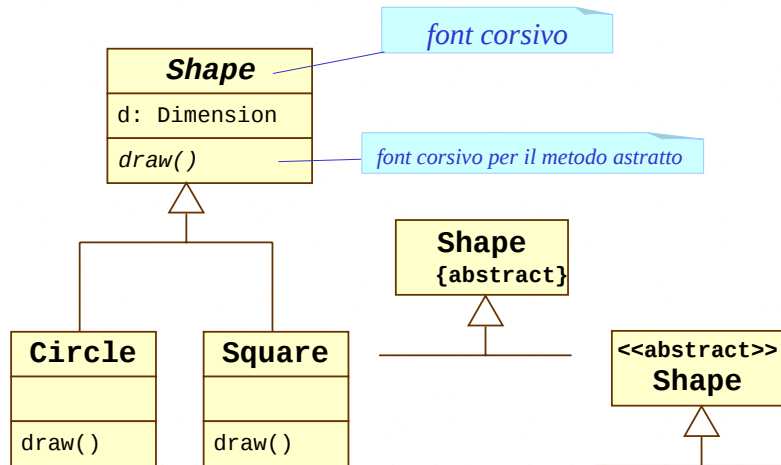
Interfacce: esempio con dipendenze



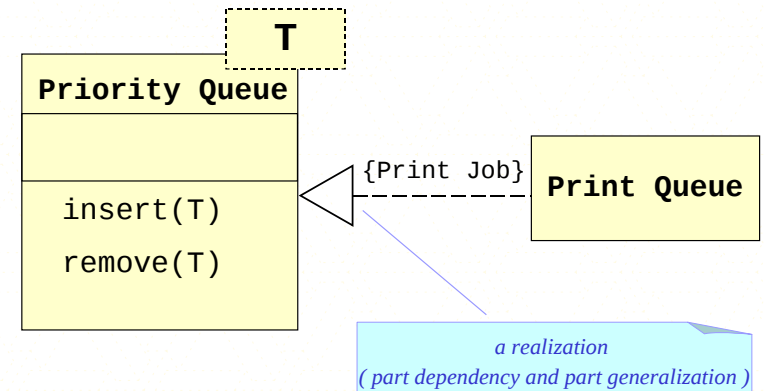
Interfacce: esempio con dipendenze



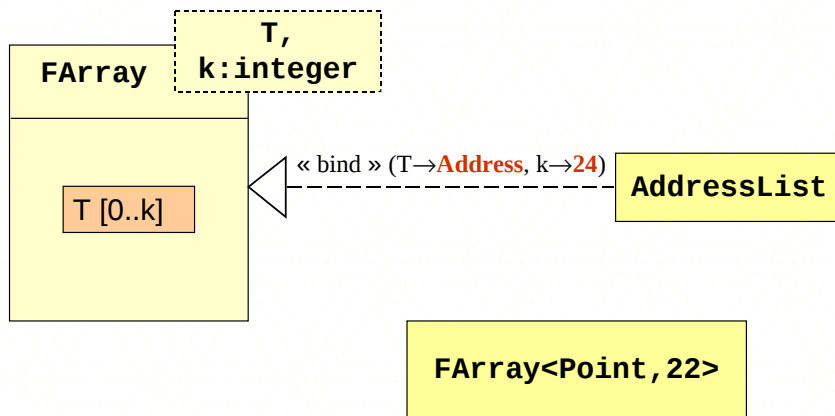
Classi astratte (notazioni equivalenti)



Classi generiche (template)

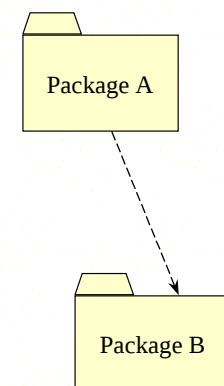


Classi generiche: diverse realizzazioni

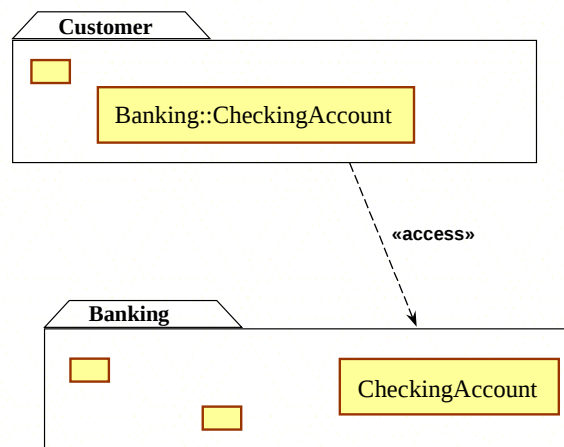


Package

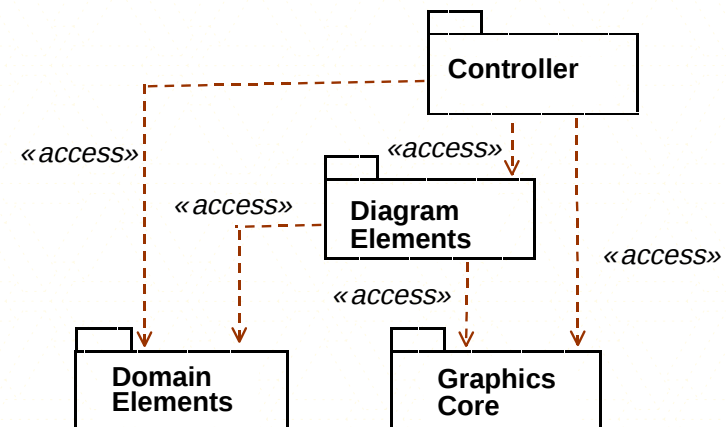
- Un package è un meccanismo generale per organizzare elementi in gruppi omogenei.
- Un packages può contenere altri package.
- Dipendenze tra package si indicano con una freccia (vedi figura).
- Vi sono delle regole di visibilità per i componenti dei package.



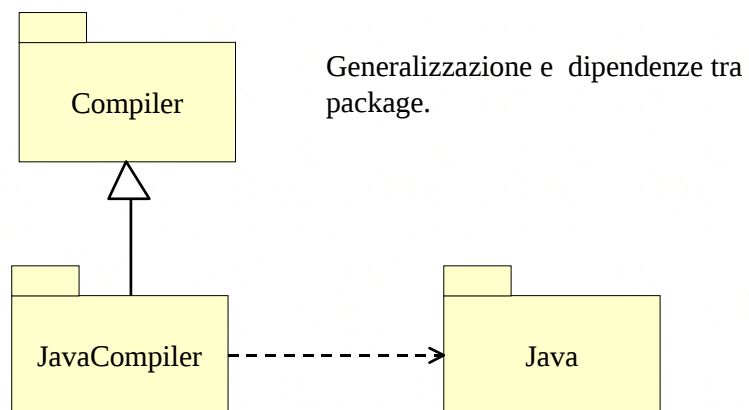
Package: dipendenze



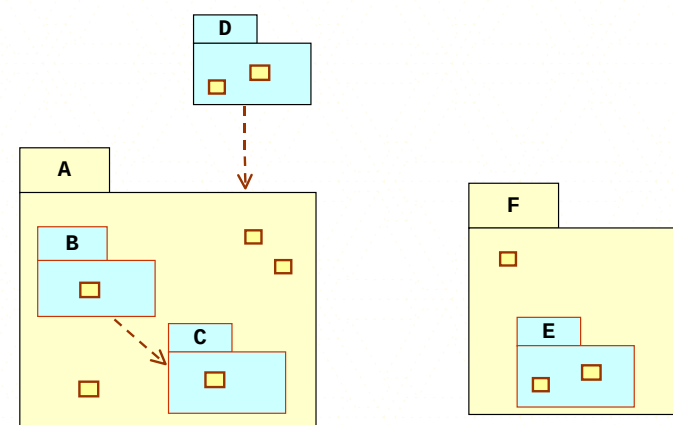
Package: dipendenze



Package



Visibilità tra package



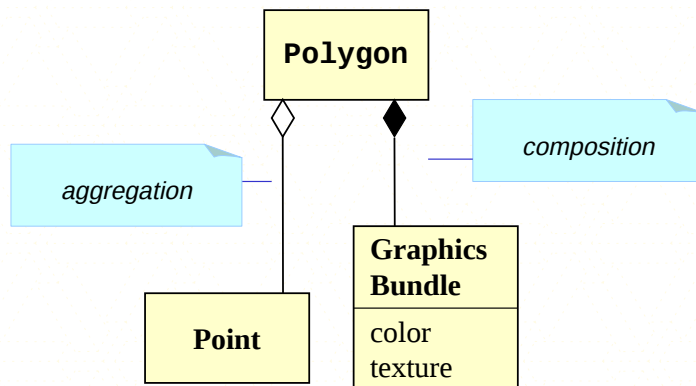
Associazione e generalizzazione: commenti

- Le associazioni descrivono relazioni che esistono tra istanze a *run time*.
 - Quando si disegna un diagramma degli oggetti, generato da un diagramma delle classi, ci deve essere un'istanza per entrambe le classi congiunte dalla associazione.
- Le generalizzazioni descrivono relazioni tra classi nei diagrammi delle classi.
 - Queste non appaiono affatto nei diagrammi degli oggetti.

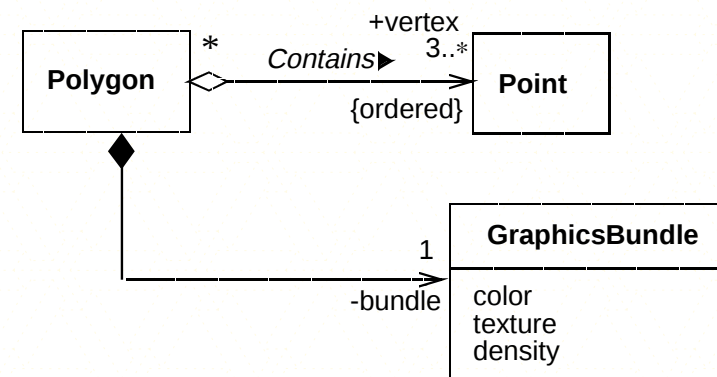
Aggregazione e Composizione

Aggregazione e Composizione

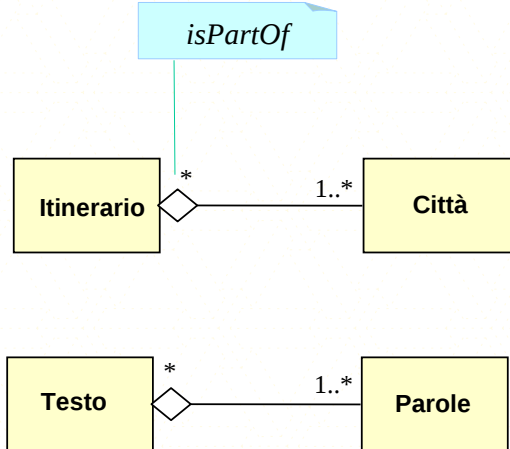
Una speciale forma di associazione che specifica una relazione tra la parte intera (*aggregato*) ed i suoi componenti (*parti*)



Aggregazione e Composizione (2)

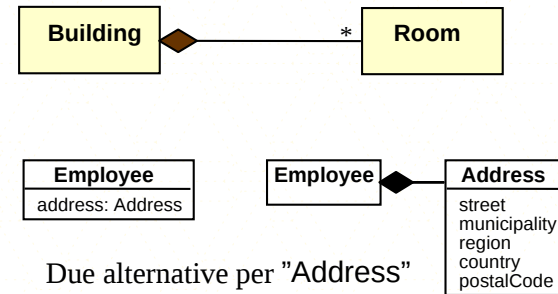


Aggregazione (Aggregation)



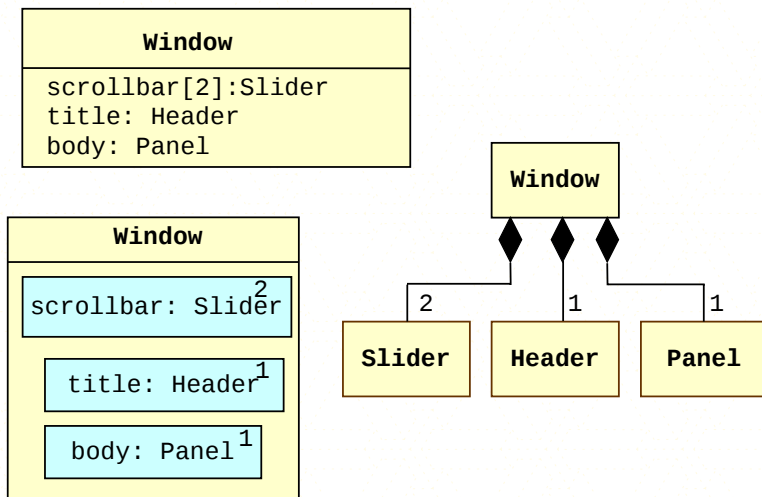
Composizione (Composition)

- Una composizione è una forma forte di aggregazione
 - Se l'aggregato viene distrutto, anche le sue parti vengono distrutte

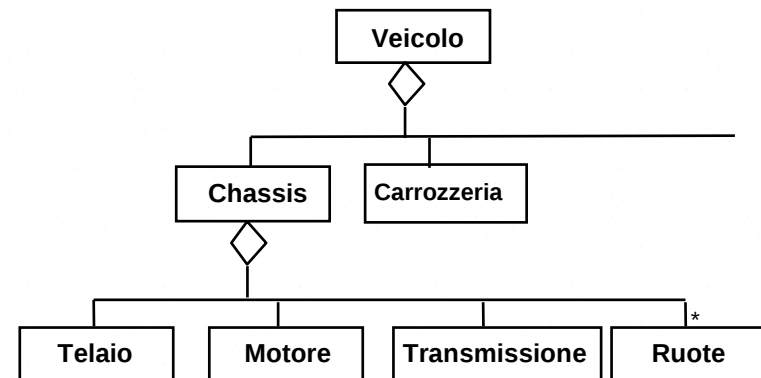


Due alternative per "Address"

Composizione: alcune notazioni equivalenti



Una gerarchia di aggregazioni (... o composizioni?)

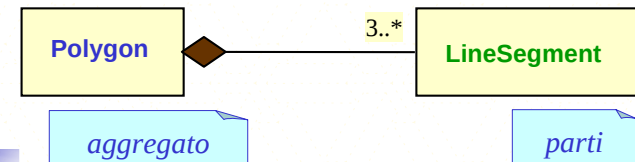


Quando usare una aggregazione od una composizione

- Come regola generale, usa associazione è una aggregazione/composizione se è vero che:
 - si può stabilire
 - le parti sono “pezzi” dell’aggregato
 - oppure l’aggregato è “composto” di parti
 - quando qualcosa che possiede o controlla l’aggregato, allora controlla anche le sue parti
- Si tratta di una aggregazione se le parti possono esistere anche se l’aggregato viene a mancare
- Si tratta di una composizione se le parti cessano di esistere quando l’aggregato viene distrutto

Propagazione

- Un meccanismo dove un’operazione su un **aggregato** è implementata facendo eseguire quella operazione sulle sue **parti**
- Allo stesso tempo, le proprietà delle **parti** si propagano spesso indietro verso l’**aggregato**
- La propagazione sta all’aggregazione come l’ereditarietà sta alla sua generalizzazione.
 - La maggior differenza è:
 - l’ereditarietà è un meccanismo implicito
 - la propagazione deve essere programmata quando richiesto



Composizione e classi annidate: differenze

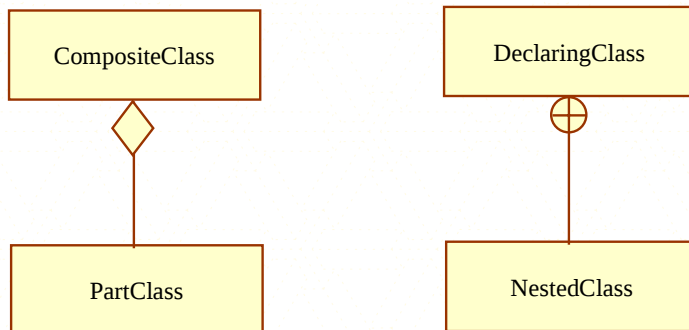


Diagramma degli oggetti

Object Diagram or Instance Diagram

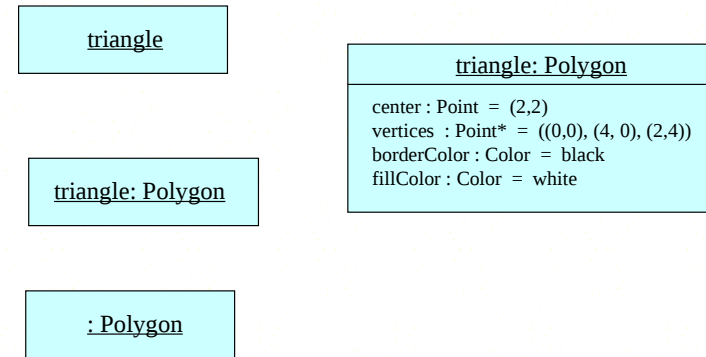
An object diagram is a graph of instances, including objects and data values.

A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time.

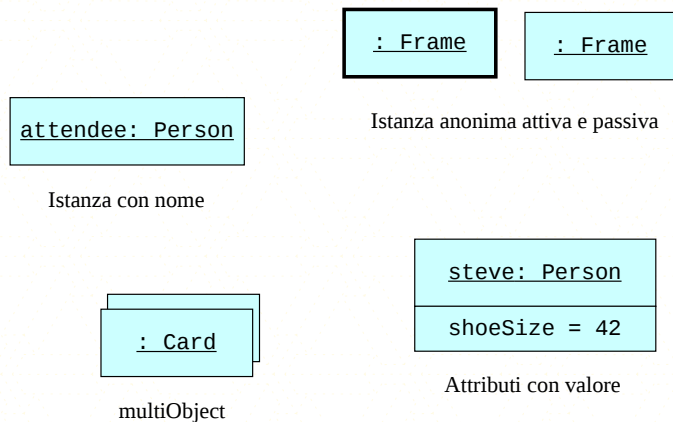
The use of object diagrams is fairly limited, mainly to show examples of data structures.

dal manuale di riferimento di UML v.1.5

Istanze o oggetti (1)



Istanze o oggetti (2)



Istanze e ruoli (notazione standard?)

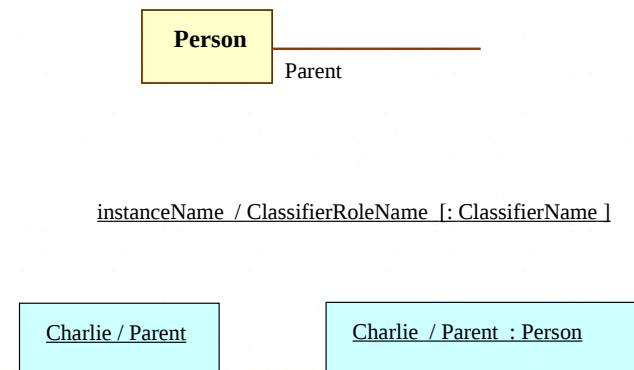


Diagramma degli oggetti

Un **link** è una istanza di una associazione (nello stesso modo in cui affermiamo che un oggetto è una istanza di una classe)

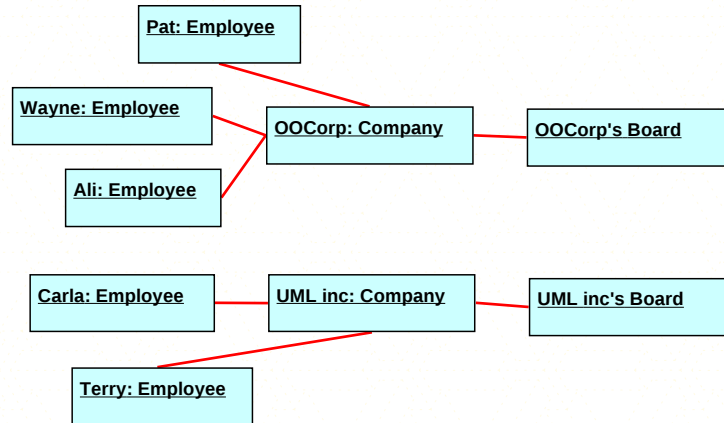
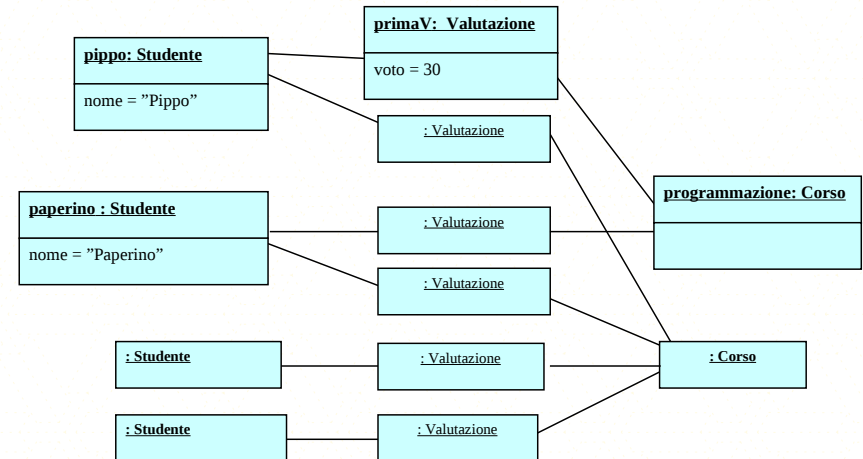
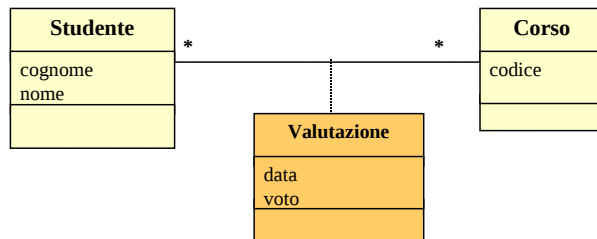


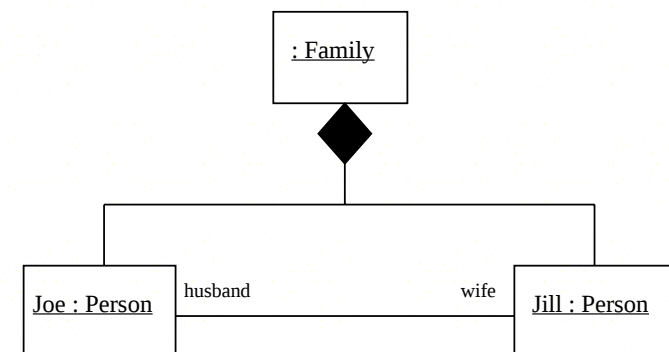
Diagramma degli oggetti ...



... relativo diagramma delle classi

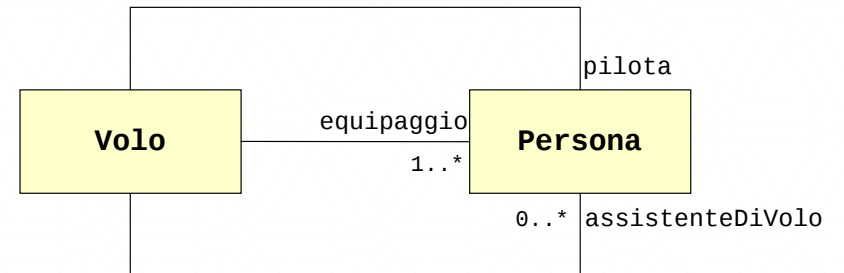


Esempio diagramma degli oggetti con aggregazione

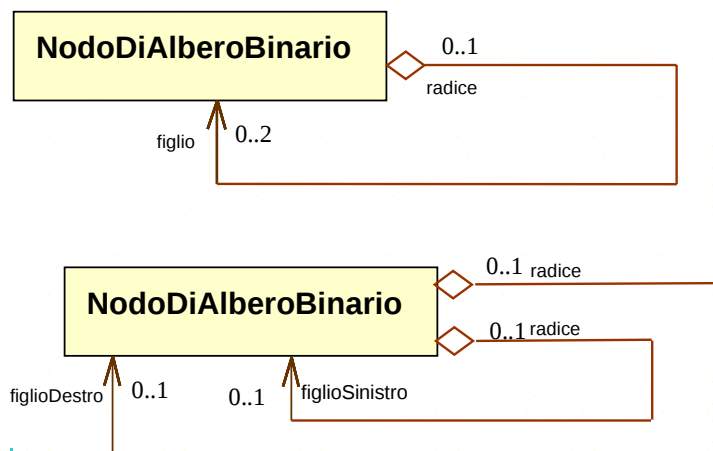


Esempi riassuntivi

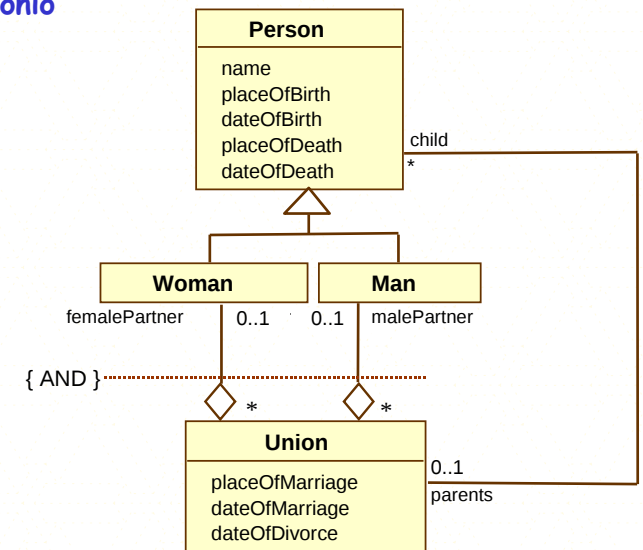
Persone e voli



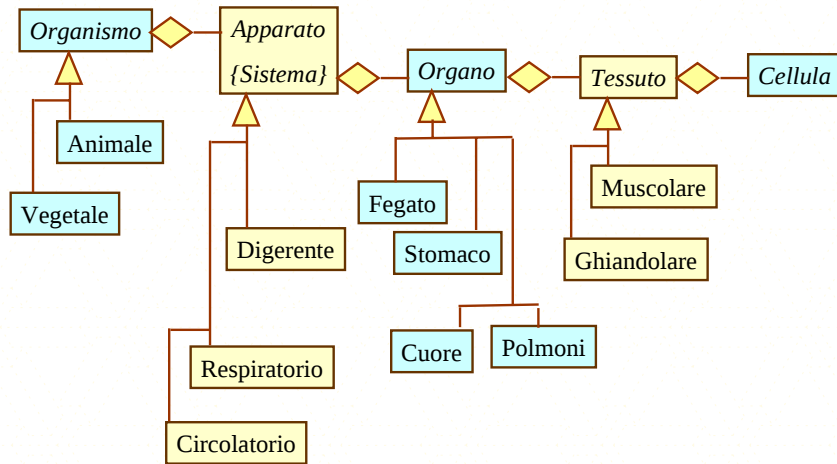
Alberi binari



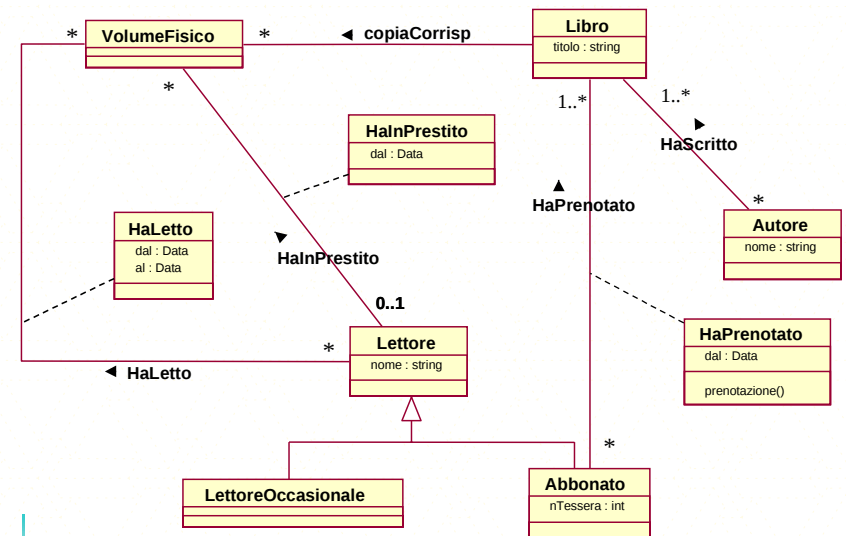
Matrimonio



Organismo

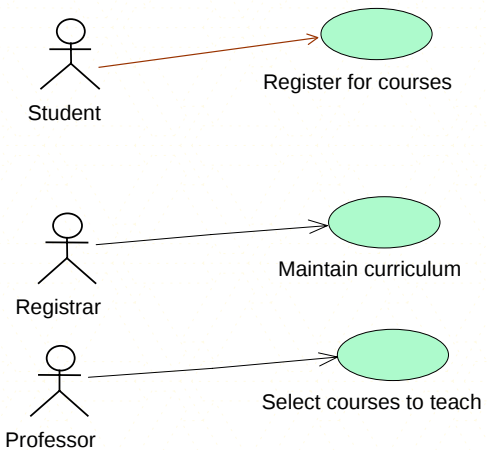


Biblioteca

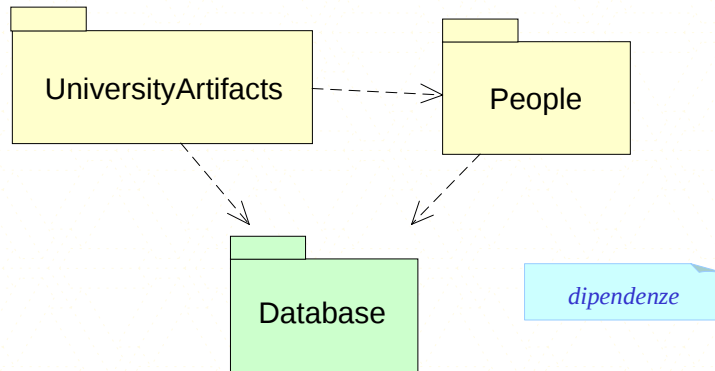


Sistema universitario: registrazioni

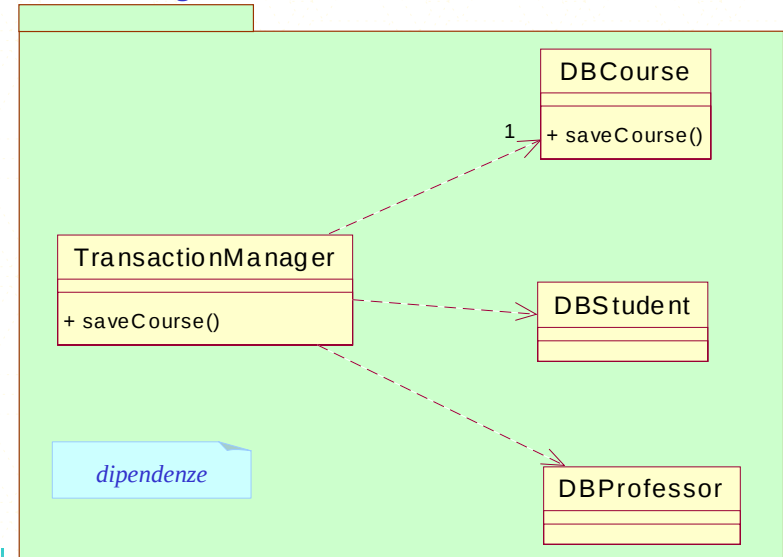
Sistema universitario: use case



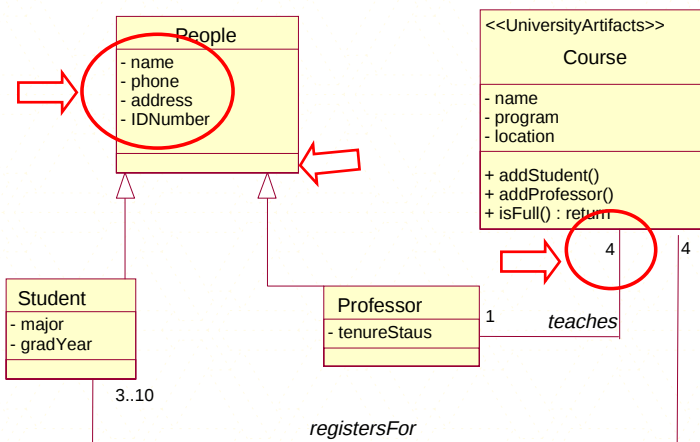
Sistema di registrazione universitario (vari package)



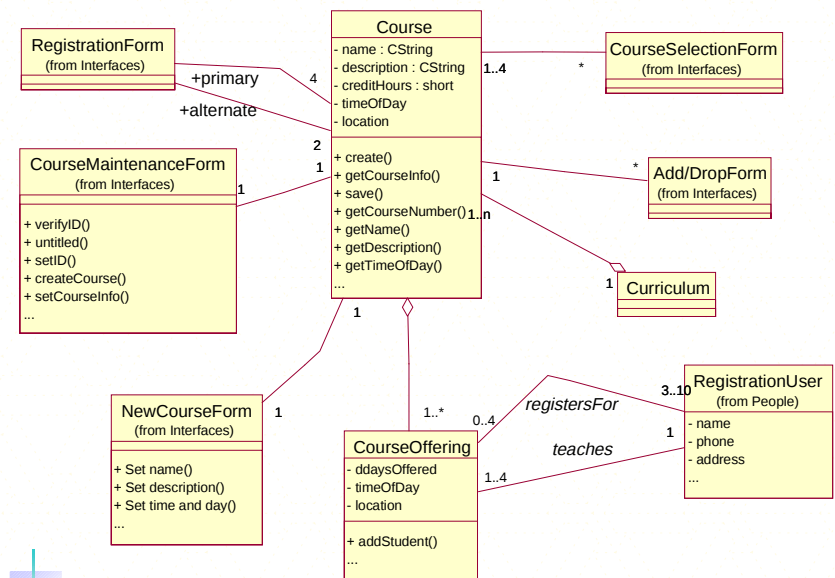
Sist. di registrazione universitario: Data Base



Sist. di registrazione universitario: People (presenti alcuni difetti)



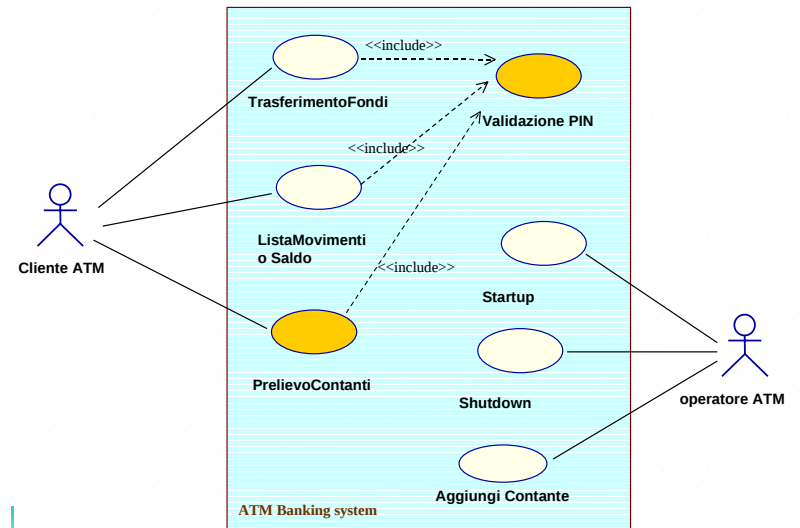
Sist. di registrazione universitario: UniversityArtifacts



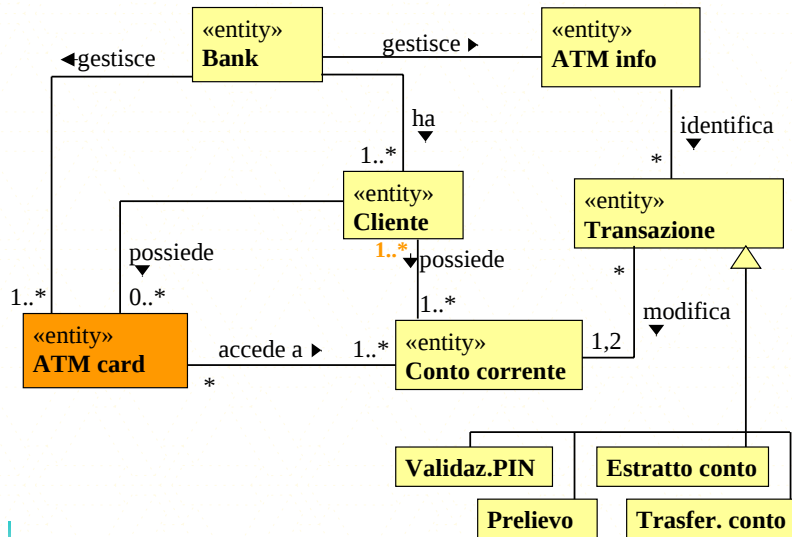
Sistema bancomat (ATM)

Use case sistema bancomat (ATM) livello -

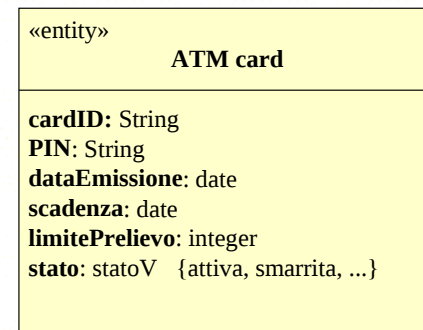
- alto



Banking System

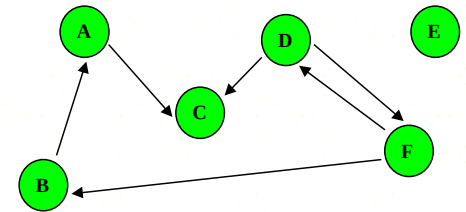


Dettagli per la classe ATM card (soli attributi)

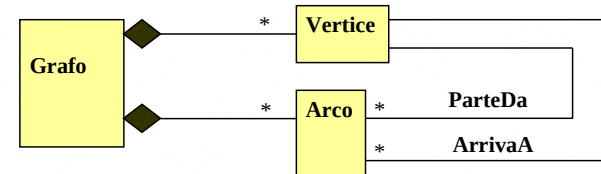


Struttura dati "Grafo"

Studio della struttura dati "Grafo" (1)

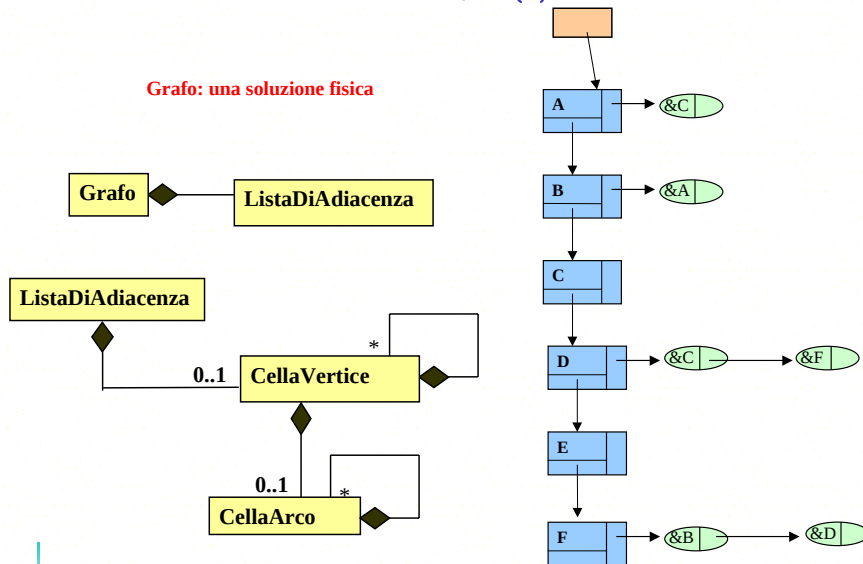


Grafo: schema dal punto di vista logico



Studio della struttura dati "Grafo" (2)

Grafo: una soluzione fisica



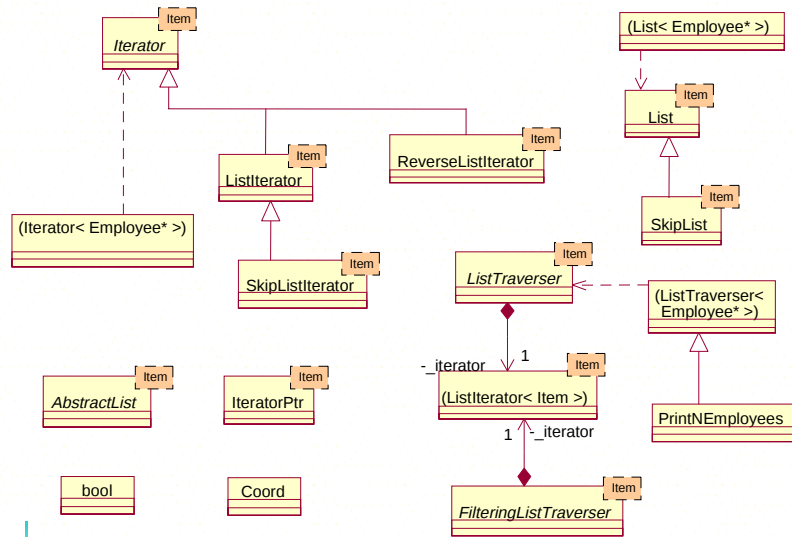
Design pattern iteratore

tratto dal testo

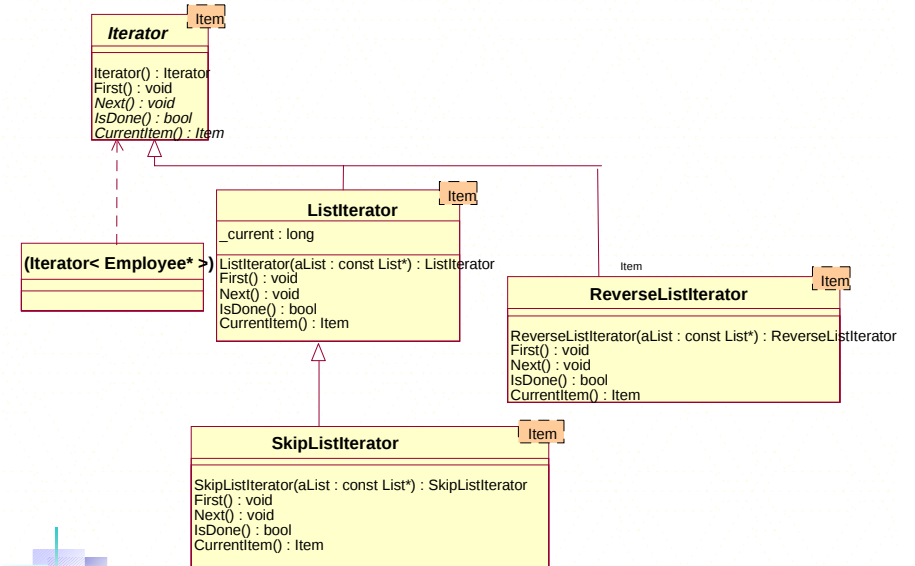
"Design Patterns": Gamma, Helm, Johnson, Vlissides

da un reverse engineering
ricavato dal codice associato al testo

Design pattern iteratore (da un reverse engineering)



Design pattern iteratore (particolare)



Design pattern iteratore (codici C++)

```
template <class Item>
class ListIterator : public Iterator<Item>
{
public:
    ListIterator(const List<Item>* aList);
    virtual void First();
    virtual void Next();
    virtual bool IsDone() const;
    virtual Item CurrentItem() const;

private:
    const List<Item>* _list;
    int _current;
};
```

```
template <class Item>
class Iterator
{
public:
    virtual void First() = 0;
    virtual void Next() = 0;
    virtual bool IsDone() const = 0;
    virtual Item CurrentItem() const = 0;
protected:
    Iterator();
};
```

Riassunto delle relazioni

Construct	descrizione	sintassi
Associazione <i>Association</i>	Una relazione tra due o più classificatori che implica una connessione tra le loro istanze	
Composizione e Aggregazione <i>Composition and Aggregation</i>	Una speciale forma di associazione che specifica una relazione tra la parte intera (<i>aggregato</i>) ed i suoi componenti (<i>parti</i>)	 
Annidamento <i>Nesting</i>	Una relazione tra due o più classi: all'interno di una classe vengono dichiarate altre classi	
Generalizzazione <i>Generalization</i>	Una relazione tassonomica tra un elemento più generale ed uno più specifico	
Dipendenza <i>Dependency</i>	Una relazione tra due elementi, nei quali il cambiamento nell'elemento indipendente può influire nell'elemento dipendente	
Realizzazione <i>Realization</i>	Una relazione tra una specificazione e la sua implementazione	

Bibliografia

Grady Booch, James Rumbaugh, Ivar Jacobson.
The Unified Modeling Language User Guide, Addison Wesley , (1999).

Grady Booch, James Rumbaugh, Ivar Jacobson
The Unified Modeling Language Reference Manual ,
Addison Wesley, (1999).

Ivar Jacobson, Grady Booch, James Rumbaugh
The Unified Software Development Process,
Addison Wesley, (1999).

Riferimenti nel Web

OMG UML - www.omg.org/uml/

- Reference manual UML 1.5
- UML 2.0 Superstructure Specification (ptc/03-08-02)

UML: tool, demo, doc
www.rational.com

UML: Tutorial e link:
www.kobryn.com