

# Laboratorio di Sistemi Software

## UML per Design Patterns e Refactoring

Luca Padovani (A-L)  
Riccardo Solmi (M-Z)

### Indice degli argomenti

---

- **Introduzione alla notazione UML**
- **I diagrammi**
  - Class Diagram
  - Object Diagram
  - Sequence Diagram
- **Alcuni esempi di diagrammi**

## Bibliografia

---

- ***UML – Unified Modeling Language***
  - UML Reference Page (OMG)  
[www.omg.org/uml](http://www.omg.org/uml)
  - UML Quick Reference  
[www.holub.com/goodies/uml/index.html](http://www.holub.com/goodies/uml/index.html)
- ***Eclipse IDE + UML Plugin***
  - Ambiente di programmazione che supporta refactoring e UML  
[www.eclipse.org](http://www.eclipse.org) + [www.eclipseuml.com](http://www.eclipseuml.com)
- **Trovate tutto nelle pagine web relative a questo corso:**
  - [http://www.cs.unibo.it/~solmi/teaching/labss\\_2002-2003.html](http://www.cs.unibo.it/~solmi/teaching/labss_2002-2003.html)
  - <http://www.cs.unibo.it/~lpadovan>

## Ricevimento

---

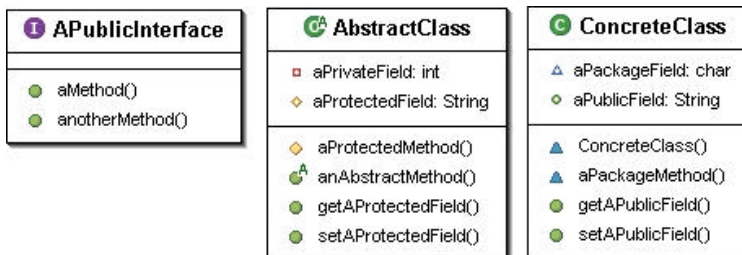
- ***Metodo 1: il newsgroup***
  - Avete a disposizione un newsgroup:  
[unibo.cs.informatica.paradigmiprogrammazione](mailto:unibo.cs.informatica.paradigmiprogrammazione)
  - Utilizzatelo il più possibile; se nessuno risponde in due-tre giorni, risponderà uno dei docenti
  - Valuteremo anche la partecipazione al newsgroup
- ***Metodo 2: subito dopo le lezioni***
  - siamo a vostra disposizione per chiarimenti
  - in aula e alla lavagna, in modo da rispondere a tutti
- ***Metodo 3: ricevimento su appuntamento***
  - Scrivete a [lpadovan@cs.unibo.it](mailto:lpadovan@cs.unibo.it) (A-L)  
o a [solmi@cs.unibo.it](mailto:solmi@cs.unibo.it) (M-Z)

# UML – Unified Modeling Language

- UML è una *notazione* per analizzare, specificare, visualizzare e documentare lo sviluppo dei documenti di progetto di un Sistema (Software)
- NB. UML non è un metodo; non definisce un processo di sviluppo.
- Modellare significa progettare
  - Un *modello* è una astrazione che permette di ragionare su un Sistema Software prima di implementarlo.
- UML fa uso di 12 tipi di diagrammi.
- Noi studiamo solamente:  
Class Diagram, Object Diagram, Sequence Diagram
  - I primi due descrivono la struttura di un modello; il terzo descrive le interazioni.
  - Sono usati nei cataloghi di Design Patterns e Refactoring
  - Supportano meglio la sincronizzazione tra modello e implementazione

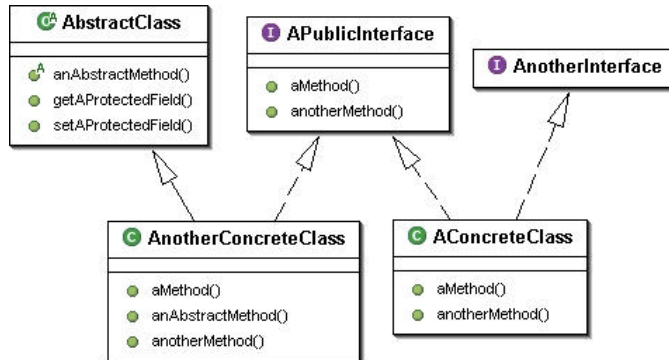
# Diagramma delle classi (Class diagram)

- E' un grafo che rappresenta le entità di un modello come *classi e interfacce* assieme ai loro contenuti (*campi e/o metodi*) e alle loro relazioni statiche.
  - Una classe/interfaccia è rappresentata da un rettangolo diviso in tre parti: nome, campi e metodi.
  - La *visibilità* degli attributi (public, protected, package, private) è mostrata con diversi simboli.
  - Viene usato per mostrare solo le informazioni rilevanti al fine del diagramma.



## Diagramma delle classi /2

- Una **generalizzazione** mostra una relazione di ereditarietà usando una freccia con la punta vuota.
  - L'ereditarietà di *implementazione* viene mostrata con una linea piena mentre l'ereditarietà di *interfaccia* con una linea tratteggiata.

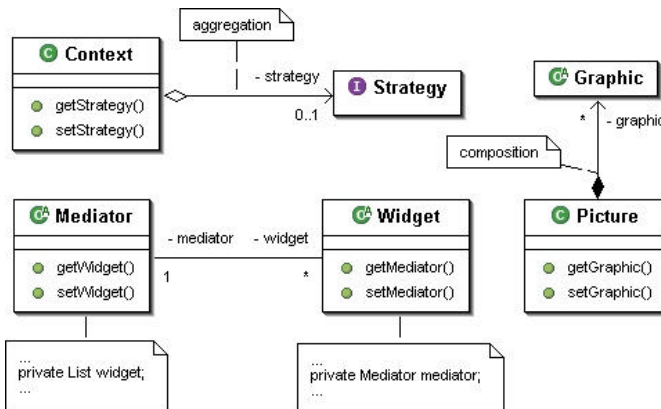


© 2002-2003 Riccardo Solmi

7

## Diagramma delle classi /3

- Una **associazione** rappresenta una relazione tra 2 classi/interfacce
  - Può essere *navigabile* in entrambe le direzioni o solo una.
  - Può essere di tipo *aggregazione* o *composizione*.
  - Può avere una *molteplicità* (0..1, 1, \*, 1..\*)



© 2002-2003 Riccardo Solmi

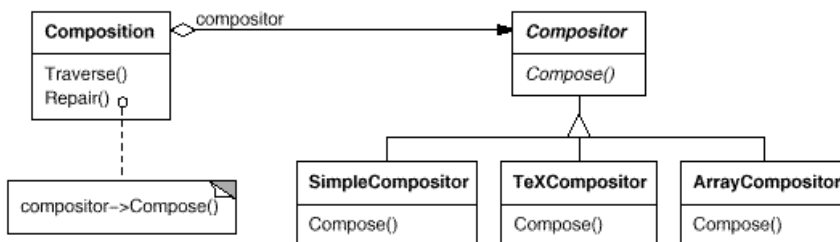
8

## Diagramma delle classi /4

- **Una associazione viene sempre tradotta in Java con un campo (due se navigabile in entrambe le direzioni).**
  - Il tipo del campo dipende dalla molteplicità.
  - Il tipo di associazione: aggregazione o composizione non influenza il codice prodotto in Java.
- **Aggregazione**
  - forma di associazione che specifica una relazione tutto-parte tra un aggregato (il "tutto") e una parte componente.
- **Composizione**
  - forma di aggregazione caratterizzata dalla forza della proprietà del tutto sulla parte e dalla coincidenza dell'esistenza temporale della parte con il tutto.

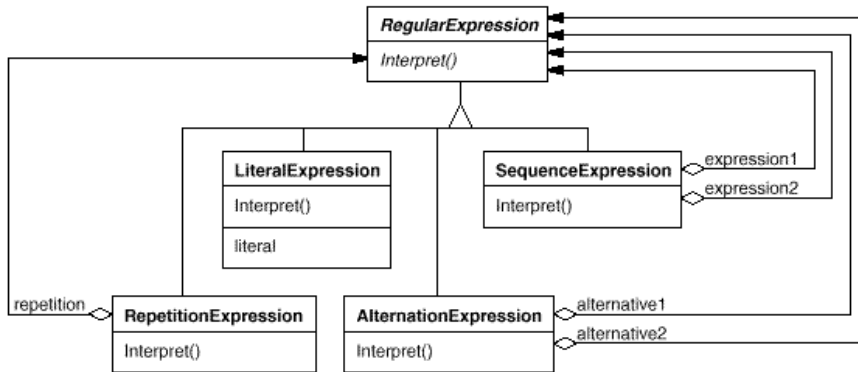
## Diagramma delle classi /5

- **EsempioStrategy**
  - NB Notazione GoF in questo e nei seguenti lucidi



## Diagramma delle classi /6

- **Esempio Interpreter**



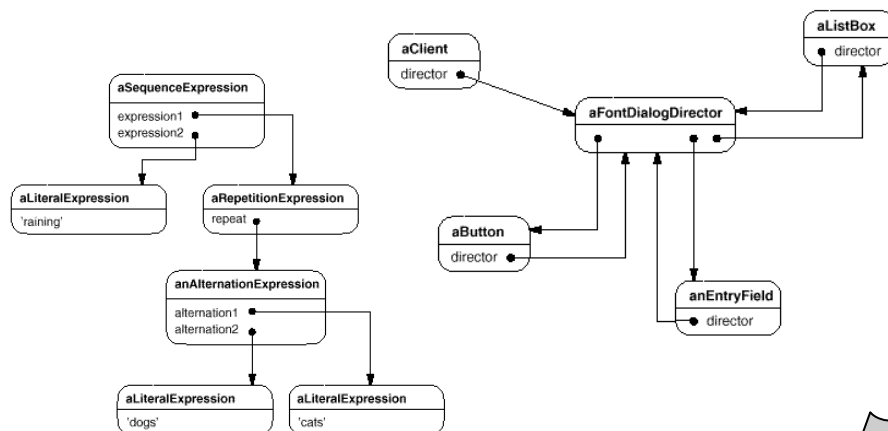
© 2002-2003 Riccardo Solmi

11

## Diagramma degli oggetti (Object diagram)

- **E' un grafo di istanze di oggetti**

- Mostra un insieme di oggetti e le loro relazioni in un certo momento.
- E' una semplice istanza del diagramma delle classi.



© 2002-2003 Riccardo Solmi

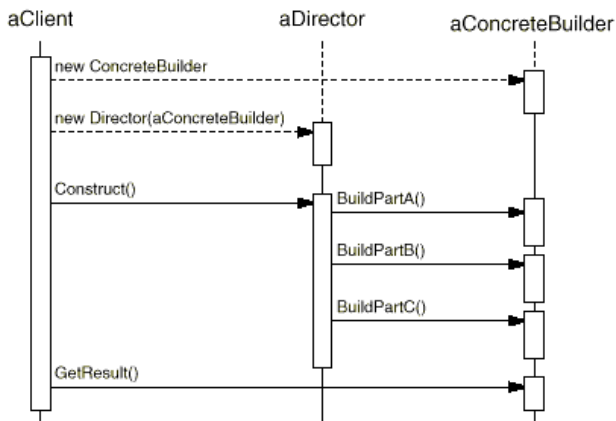
12

## Diagramma di sequenza (Sequence diagram)

- **Un diagramma che illustra le *interazioni* tra gli oggetti disponendole lungo una *sequenza temporale*.**
  - Tutti gli oggetti che partecipano all'interazione più eventuali clienti vengono disposti orizzontalmente in alto
  - Dall'alto al basso è possibile seguire la sequenza delle interazioni tra i partecipanti.
  - Per ogni partecipante vi è disegnata una linea verticale tratteggiata prima della creazione dell'oggetto poi piena.
  - Le interazioni tra partecipanti consistono in invocazioni di metodi (con i parametri indicati) e sono disegnate come frecce orizzontali.
  - Accorgimenti grafici sono usati per denotare chiamate sullo stesso oggetto, iterazioni, chiamate condizionali, ecc ...

## Diagramma di sequenza /2

- **Esempio Builder**



## Diagramma di Sequenza /3

- **Esempio Observer**

