Luca Cabibbo

Analisi e Progettazione del Software

Analisi e progettazione orientata agli oggetti

Capitolo 1

marzo 2017

Il tempo è un grande professore,
ma sfortunatamente
uccide tutti i suoi allievi.

Hector Berlioz

1.1 Contenuto del corso e sua utilità

Questo corso è basato sul libro di **Craig Larman**

- **Applicare UML e i pattern**
  analisi e progettazione orientata agli oggetti
  - quarta edizione, 2016
  - Pearson Education Italia, ISBN 9788891901033
- è un'introduzione all'analisi e alla progettazione orientata agli oggetti (OOA/D) basata sull'*applicazione* (nel senso di *azionare*, *mettere in pratica*) di
  - UML – per la modellazione del software
  - pattern – principi di analisi e progettazione
  - un processo iterativo – come contesto per lo sviluppo del software

A P S Analisi e progettazione a oggetti

Sviluppare software è divertente – ma sviluppare software di qualità è un'attività complessa

- infatti un sistema software deve
  - implementare un insieme di funzionalità richieste
  - fornire alcune qualità desiderate
- come dominare questa complessità?

Alcuni problemi pragmatici nello sviluppo di software OO

- quali sono gli oggetti? quali le classi?
- che cosa deve conoscere ciascun oggetto? che deve saper fare?
- come collaborano gli oggetti?

A P S Analisi e progettazione a oggetti

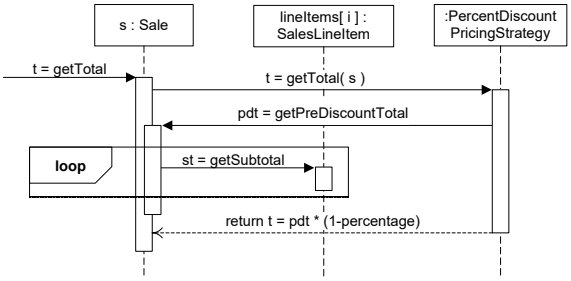
Lo sviluppo di software OO può essere sostenuto dallo svolgimento delle seguenti attività (semplificate)

- analisi (requisiti) – identifica i requisiti funzionali e di qualità del sistema software di interesse (non è OO)
- analisi (OOA) – definisci un modello (a oggetti) del dominio del problema
- progettazione (OOD) – scegli gli elementi software (oggetti) e allocagli delle responsabilità (operazioni e dati)
- inoltre, queste attività possono essere guidate dall'applicazione di opportuni principi e pattern
- **l'analisi e la progettazione del software sono attività altamente creative – ma, fortunatamente, le sue fondamenta possono essere studiate e apprese**

Modelli e modellazione

Un'attività fondamentale nello sviluppo del software – soprattutto nell'analisi e progettazione di sistemi software complessi – è la **modellazione**

- un **modello** è una semplificazione della realtà che descrive completamente un sistema da un particolare punto di vista



Gestisci Restituzione
Scenario principale di successo:
Un cliente arriva alla cassa con alcuni articoli da restituire. Il cassiere usa il sistema POS per registrare ciascun articolo restituito …
Scenari alternativi:
Se il cliente aveva pagato con la carta di credito, ...
Se ...

Modelli e modellazione

Un'attività fondamentale nello sviluppo del software – soprattutto nell'analisi e progettazione di sistemi software complessi – è la **modellazione**

- un **modello** è una semplificazione della realtà che descrive completamente un sistema da un particolare punto di vista
- la modellazione è importante perché può favorire i ragionamenti e la comunicazione
- **l'importanza di un modello non è in un particolare diagramma, ma è nell'idea che il diagramma intende comunicare**

Applicare UML

Unified Modeling Language (**UML**)

- è una notazione standard, visuale per scrivere (disegnare) modelli – relativi soprattutto al software OO

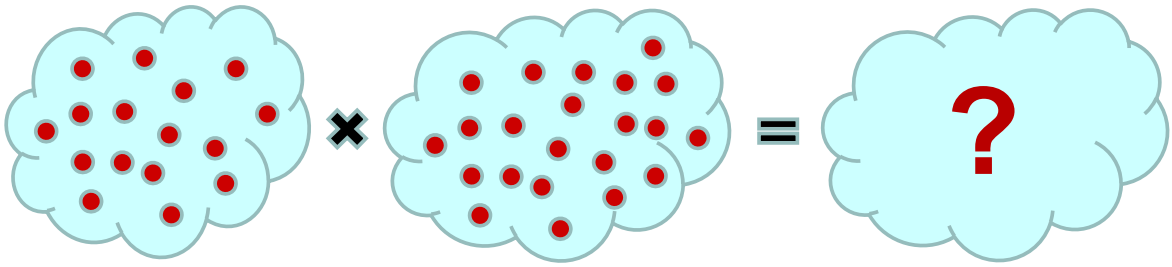
Nella modellazione, una notazione è utile – indispensabile – ma è molto più importante saper **pensare a oggetti**

- non ha alcuna utilità conoscere UML, senza però sapere come creare un buon progetto OO
- nella modellazione sono importanti le idee – ed è importante poter comunicare le idee

Sull'importanza di una buona notazione

Il matematico Whitehead, circa un secolo fa, osservava

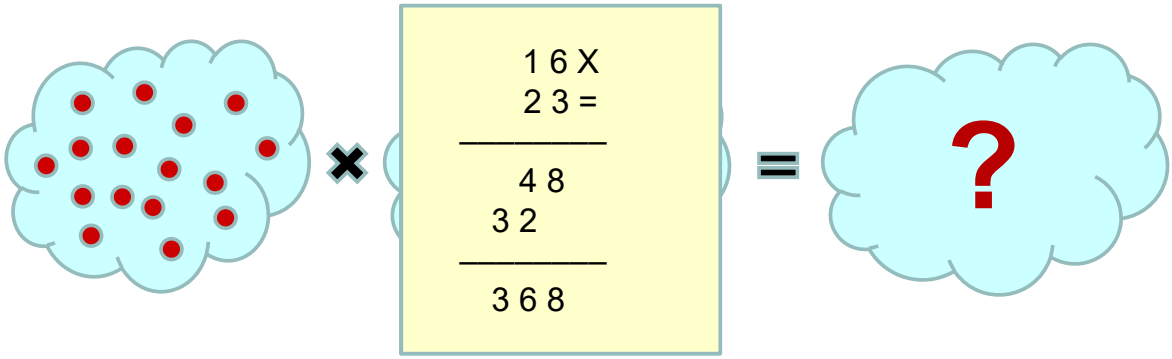
- liberando il cervello da tutto il lavoro non necessario, una buona notazione ci lascia liberi di concentrarci su problemi più avanzati – e in effetti incrementa il nostro "potere mentale"



Sull'importanza di una buona notazione

Il matematico Whitehead, circa un secolo fa, osservava

- liberando il cervello da tutto il lavoro non necessario, una buona notazione ci lascia liberi di concentrarci su problemi più avanzati – e in effetti incrementa il nostro "potere mentale"



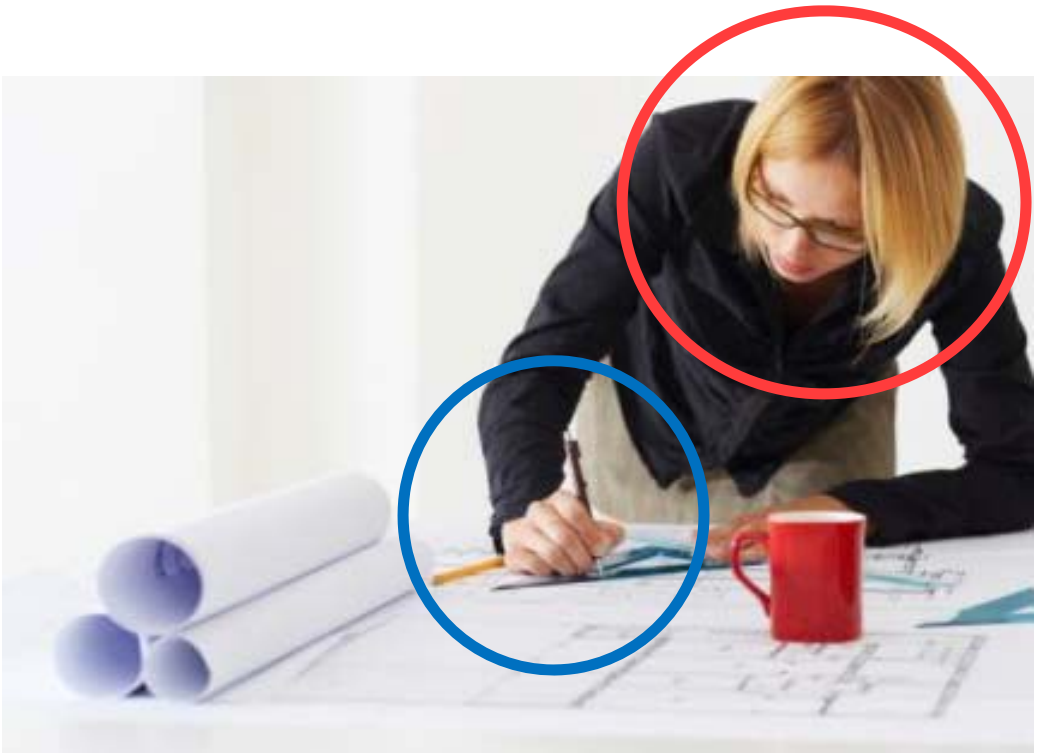
- dunque, una buona notazione è indispensabile
- molto più importante, però, è sapere cosa farci e come utilizzarla

Applicare UML

I segreti della modellazione

- lo *scopo primario della modellazione* è **comprendere** – e non documentare
- il *valore primario della modellazione* è nella **discussione durante la modellazione** – noi modelliamo per poter conversare
- **UML può essere applicato per descrivere idee, ragionamenti e scelte, di analisi e progettazione**

Che cosa è importante?



Programmazione a oggetti: il problema

Sviluppo di software OO

- oggetti e classi
- ciascun oggetto conosce delle informazioni
- ciascun oggetto sa eseguire delle operazioni

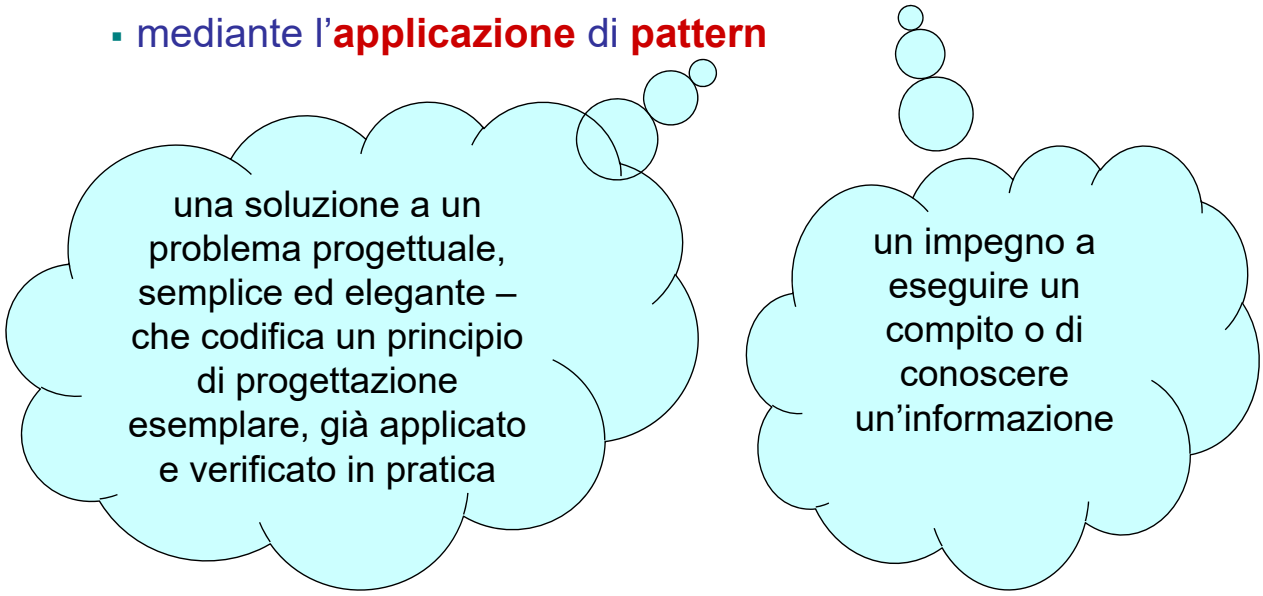
Problemi nello sviluppo di software OO

- quali sono gli oggetti e le classi?
- che cosa deve conoscere ciascun oggetto/classe?
- che cosa deve fare ciascun oggetto/classe?
- come **collaborano** gli oggetti?
- in generale, come vanno allocate le **responsabilità** (operazioni e dati) agli oggetti?

Progettazione a oggetti: principi e pattern

Lo sviluppo di software OO può essere basato sull'applicazione di opportuni principi e pattern

- in questo corso
  - **progettazione guidata dalle responsabilità**
  - mediante l'**applicazione** di **pattern**



Studi di caso

Il corso illustra l'applicazione di tutti gli strumenti insegnati con riferimento ad alcuni studi di caso

- gli studi di caso del libro
- altri studi di caso

Analisi dei requisiti e casi d'uso

L'enfasi del corso è sull'OOA/D

- ma presenta anche l'attività preliminare dell'**analisi dei requisiti**
  - con riferimento ai **casi d'uso** e alle **storie utente**

Sviluppo iterativo e modellazione agile

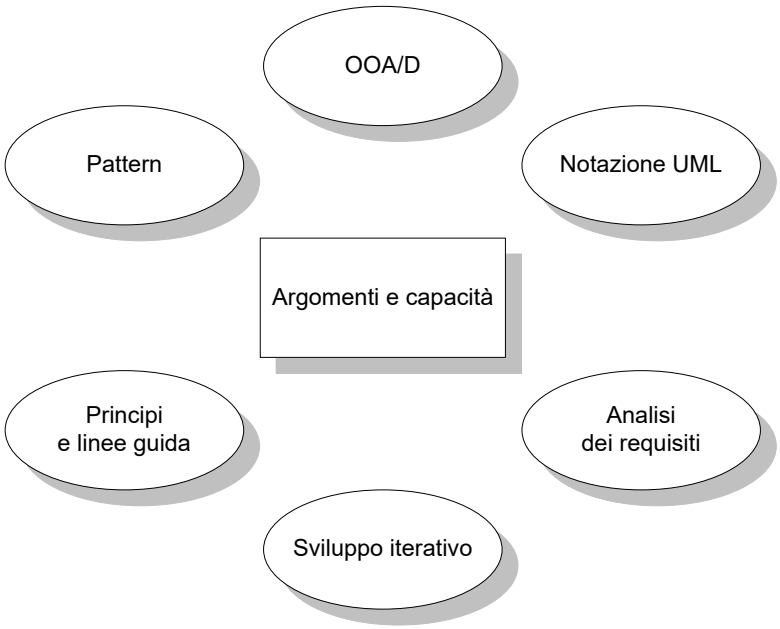
È utile descrivere le attività di analisi progettazione nel contesto di un **processo per lo sviluppo del software**

- un **processo** è un approccio sistematico per lo sviluppo del software – che ne organizza ruoli, attività e prodotti, tempi e modi
- perché è utile farlo?
  - per capire la relazione tra le diverse attività dello sviluppo del software
- faremo riferimento a processi di sviluppo **iterativi** – che possono essere applicate in modo **agile**
  - in particolare, **Unified Process** (**UP**) e a **Scrum**
- i concetti presentati sono rilevanti anche per altri processi ed approcci

In sintesi

Obiettivo del corso è sostenere lo sviluppo di software di qualità

- che cosa serve per sviluppare software di qualità?



1.2 L'obiettivo di apprendimento principale

Qual è la singola capacità più importante nell'OOA/D?

- **una capacità critica nello sviluppo OO è quella di assegnare in modo abile responsabilità agli oggetti software**

In particolare, in questo corso l'OOD è fortemente basata su principi per l'assegnazione di **responsabilità**, codificati come pattern

- pattern GRASP – nove principi fondamentali della progettazione a oggetti

A P S 1.3 Che cosa sono analisi e progettazione

L'**analisi** enfatizza un'**investigazione** di un problema e dei suoi requisiti – ***che cosa***

- ma non è interessata direttamente alle soluzioni del problema

La **progettazione** enfatizza una **soluzione concettuale** che soddisfa i requisiti del problema – ***come***

- ma non è interessata direttamente alla realizzazione (implementazione) della soluzione

Do the right thing, and do the thing right

- fare la cosa giusta (analisi), e fare la cosa bene (progettazione)

Analisi e progettazione hanno obiettivi diversi, perseguiti in modo diverso – tuttavia sono attività fortemente sinergiche e inseparabili

A P S 1.4 Che cosa sono OOA e OOD

L'**analisi orientata agli oggetti** (**OOA**, **Object-Oriented Analysis**) è basata principalmente sull'identificazione dei *concetti nel dominio del problema* – e su una loro descrizione a oggetti

- ad es., in un sistema per la gestione di voli, **Aereo**, **Volo** e **Pilota**

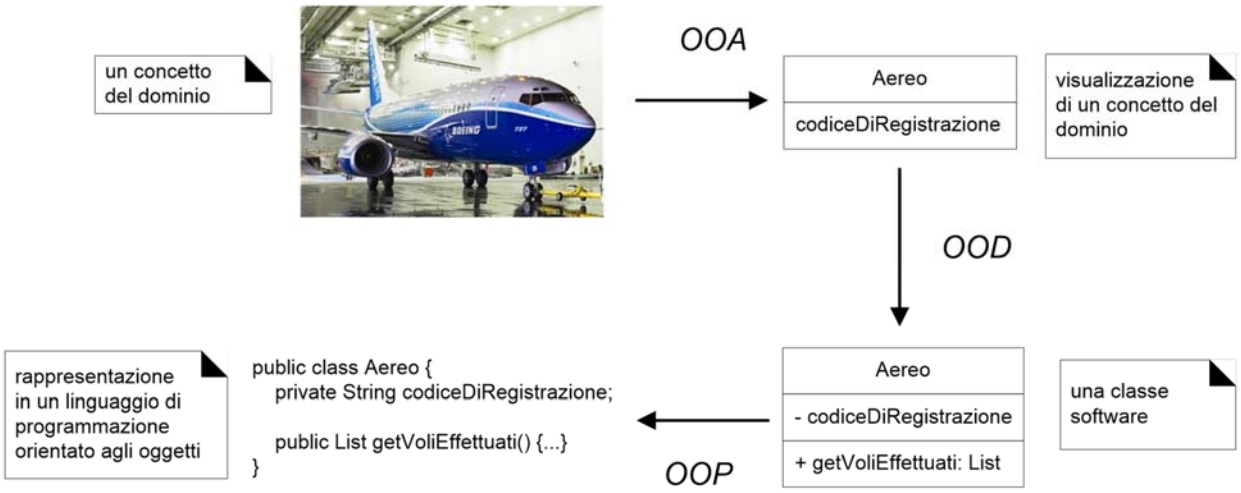
La **progettazione orientata agli oggetti** (**OOD**, **Object-Oriented Design**) è basata principalmente sulla definizione e la caratterizzazione di una comunità di *oggetti software che collaborano* per soddisfare i requisiti

- un oggetto software **Aereo** ha un attributo **codiceDiRegistrazione** e un metodo **getVoliEffettuati**

Le classi scelte durante l'OOD vengono implementate durante la **programmazione orientata agli oggetti** (**OOP** , **Object-Oriented Programming**)

- una classe **Aereo** in Java

OOA ⇒ OOD ⇒ OOP

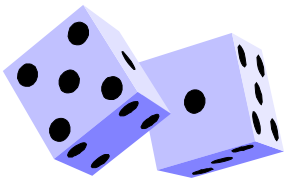


L'analisi e la progettazione del software sono attività correlate, così come sono correlate con le altre attività dello sviluppo del software

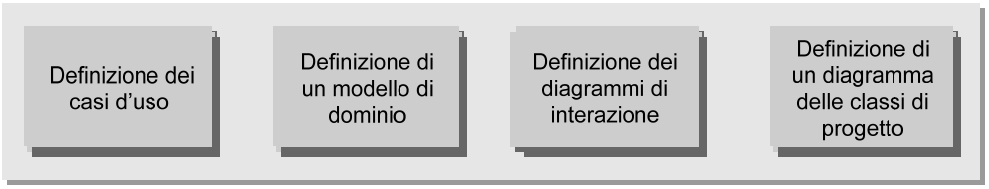
1.5 Un breve esempio

Gioca una Partita a Dadi

- un giocatore tira due dadi
- se il totale è sette, ha vinto
- altrimenti, ha perso



Attività



Definizione dei casi d'uso

L'analisi dei requisiti è interessata a descrivere le funzionalità del sistema da sviluppare

- un **caso d'uso** descrive una modalità di uso del sistema

Caso d'uso per ***Gioca una Partita a Dadi***

- il Giocatore chiede di lanciare i dadi
- il Sistema presenta il risultato: se il valore totale delle facce è sette, il giocatore ha vinto, altrimenti ha perso

Definizione di un modello di dominio

L'OOA è interessata a descrivere il dominio di interesse sulla base di un modello concettuale, a oggetti

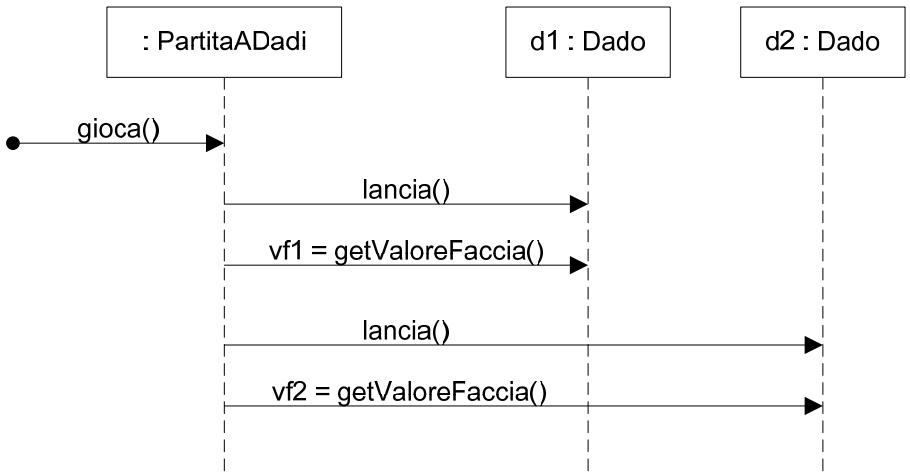
- un **modello di dominio** rappresenta graficamente i concetti, le associazioni e gli attributi significativi del dominio di interesse

- attenzione, qui si parla di **oggetti del mondo reale**

Definizione dei diagrammi di interazione

L'OOD è interessata a definire gli oggetti software e le loro collaborazioni

- un **diagramma di interazione** mostra alcuni oggetti software e le loro collaborazioni, per ottenere un certo comportamento
- descrive scelte progettuali circa l'assegnazione di responsabilità

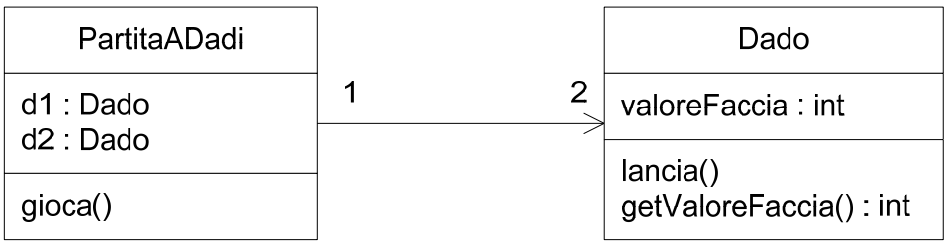


- attenzione: le classi hanno gli stessi nomi – ma adesso si parla di **oggetti software**

Diagrammi delle classi di progetto

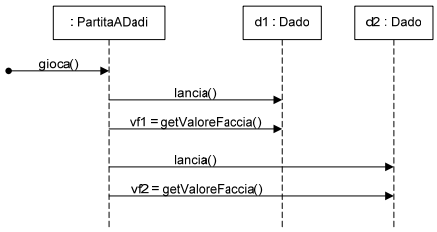
I diagrammi di interazione sono una vista dinamica sugli oggetti software

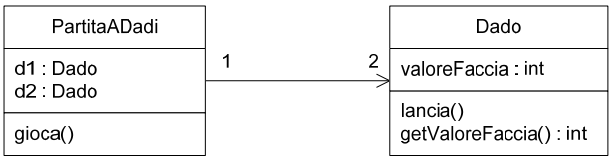
- un **diagramma delle classi di progetto** è una descrizione statica della struttura delle classi software, con i loro attributi e metodi



- anche qui si parla di **oggetti e classi software**

Dal progetto al codice






```
public class PartitaADadi {
    private Dado d1;
    private Dado d2;
    public PartitaADadi() {
        d1 = new Dado();
        d2 = new Dado();
    }
    public void gioca() {
        int vf1, vf2;
        d1.lancia();
        vf1 = d1.getValoreFaccia();
        d2.lancia();
        vf2 = d2.getValoreFaccia();
        ...
    }
}
```

```
public class Dado {
    private int valoreFaccia;
    public Dado() {
        ...
    }
    public void lancia() {
        ...
    }
    public int getValoreFaccia() {
        return valoreFaccia;
    }
}
```

Avete notato la relazione tra requisiti, analisi, progettazione, e programmazione?

1.6 Che cosa è UML

UML è una notazione grafica standard per la modellazione OO

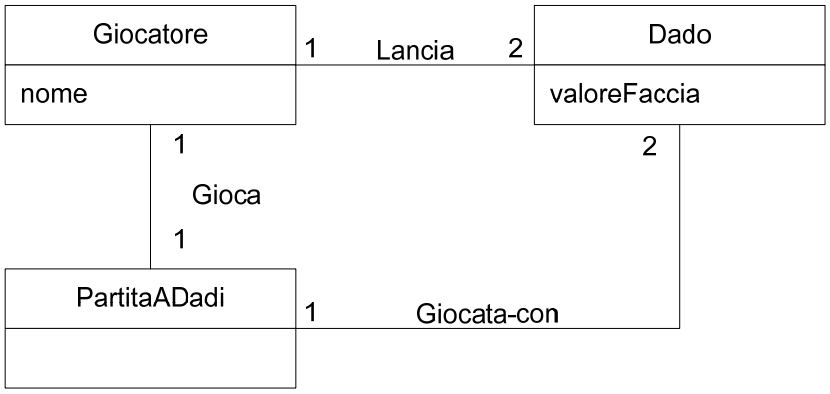
- UML (Unified Modeling Language) è un linguaggio visuale per la specifica, la costruzione e la documentazione degli elaborati di un sistema (software e non) [OMG]
- **www.uml.org**

Questo corso presenta UML (UML 2) – ma in modo parziale

- l'enfasi del corso non è su UML, ma sull'**applicazione** di UML nell'analisi e progettazione del software

Tipi di diagrammi di UML

UML definisce alcuni tipi "grezzi" di diagrammi – ad esempio i diagrammi delle classi



- per ciascun tipo di diagramma, UML definisce una sintassi e una semantica – che talvolta è volutamente generica
  - tuttavia, UML non impone né un metodo né un modo preferito di usare tali diagrammi

Punti di vista per applicare UML

È utile conoscere due possibili *punti di vista* relativamente all'applicazione di UML [Fowler, UML Distilled]

- punto di vista **concettuale**
  - i diagrammi sono utilizzati (scritti e interpretati) per descrivere oggetti del mondo reale o in un dominio di interesse
- punto di vista **software**
  - i diagrammi descrivono astrazioni o componenti software
  - in particolare, possono descrivere: (i) implementazioni in una particolare tecnologia (ad es., una classe Java o C#) oppure (ii) specifiche e interfacce di componenti software, ma indipendenti da ogni possibile implementazione (ad es., l'intenzione di definire una classe)
- attenzione: la notazione è sempre la stessa!

Punti di vista per applicare UML

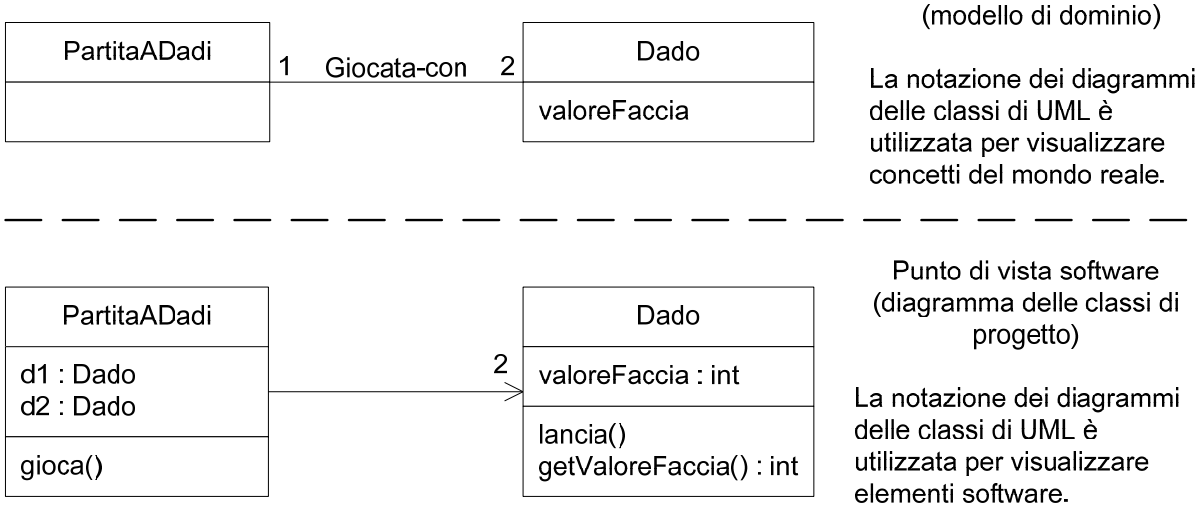
È utile conoscere due possibili *punti di vis* all'applicazione di UML [Fowler, UML Disti

- punto di vista **concettuale**
  - i diagrammi sono utilizzati (scritti e interpretati) per descrivere oggetti del mondo reale o in un dominio di interesse
- punto di vista **software**
  - i diagrammi descrivono astrazioni o componenti software
  - in particolare, possono descrivere: (i) implementazioni in una particolare tecnologia (ad es., una classe Java o C#) oppure (ii) specifiche e interfacce di componenti software, ma indipendenti da ogni possibile implement l'intenzione di definire una classe
- attenzione: la notazione è sempre

è il punto di vista prevalente nell'analisi

è il punto di vista prevalente nella progettazione

Punti di vista differenti



Punto di vista concettuale (modello di dominio)

La notazione dei diagrammi delle classi di UML è utilizzata per visualizzare concetti del mondo reale.

Punto di vista software (diagramma delle classi di progetto)

La notazione dei diagrammi delle classi di UML è utilizzata per visualizzare elementi software.

Il significato di "classe" nei diversi punti di vista

Un rettangolo, in un diagramma delle classi di UML, denota una **classe** – e, più precisamente

- **classe concettuale**
  - cosa o concetto del mondo reale – nel modello di dominio
- **classe software**
  - una classe che rappresenta un componente software
  - in particolare, può rappresentare la *specifica* di una classe oppure l'*implementazione* di una classe
- **classe**
  - classe concettuale o classe software

Tre modi per applicare UML

Inoltre, ci sono tre *modi* per applicare UML [Fowler, UML Distilled]

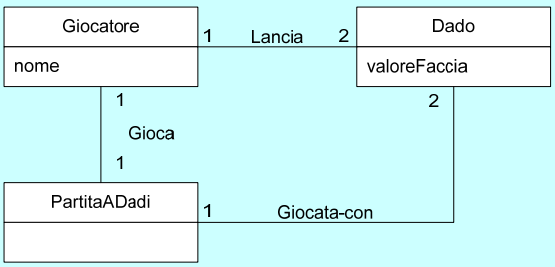
- **UML come abbozzo** (sketch)
  - diagrammi informali e incompleti – creati per esplorare parti complesse dello spazio del problema o della soluzione
- **UML come progetto** (blueprint)
  - diagrammi relativamente dettagliati usati (i) dal reverse engineering di codice esistente, o (ii) per guidare la generazione, automatica o manuale, del codice
- **UML come linguaggio di programmazione**

- la modellazione agile enfatizza l'applicazione di **UML come abbozzo**
- tuttavia, si può imparare ad applicare bene UML come abbozzo solo se si è imparato ad applicare **UML come progetto**

1.7 Vantaggi della modellazione visuale

L'uso di UML implica che si sta lavorando *in modo visuale*

- per sfruttare la capacità del nostro cervello di comprendere più rapidamente concetti e relazioni mostrati con una notazione grafica (prevalentemente bidimensionale)



Classi concettuali e attributi
Giocatore – nome
Dado – valoreFaccia
PartitaADadi

Associazioni
una **PartitaADadi** è giocato da un **Giocatore**
la **PartitaADadi** viene giocata con due **Dadi**
il **Giocatore** lancia i due **Dadi**

- quale di questi due modelli è più facilmente comprensibile?