



Analisi e progettazione orientata agli oggetti

Capitolo 1
marzo 2011

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**
 - terza edizione, 2005
 - Pearson Education Italia, ISBN 8871922700
- è un'introduzione all'analisi ed alla progettazione orientata agli oggetti (OOA/D) basata sull'applicazione (nel senso di azionare, mettere in pratica) di
 - UML – Unified Modeling Language
 - pattern
 - sviluppo iterativo – mediante un approccio agile

A P S Analisi e progettazione a oggetti

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

- è possibile dominare questa complessità – anziché farci sopraffare da essa?

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?

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

- analisi (OOA) – definire un modello (a oggetti) del problema
- progettazione (OOD) – scegliere gli elementi software (oggetti) e allocargli delle responsabilità (operazioni e dati)
- attenzione alle qualità del software

A P S Analisi e progettazione a oggetti

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

Alcuni problemi pragmatici nello sviluppo di software OO

- **La progettazione del software è in parte arte, in parte ingegneria, in parte tentativi ed esperimenti**

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

- **In ogni caso, anche se la progettazione del software è un'attività altamente creativa, le sue fondamenta possono essere studiate ed apprese**

- attenzione alle qualità del software

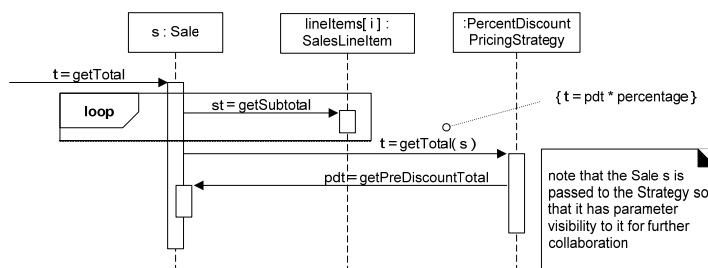
A P S Modelli e modellazione

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

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

A P S Modelli e 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 ...

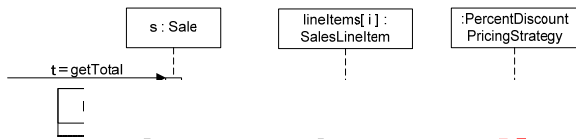
Scenari alternativi:

Se il cliente aveva pagato con la carta di credito, e la transazione di rimborso sul relativo conto di credito è stata respinta, allora il cliente viene informato e viene rimborsato in contanti.

Se ...

A P S Modelli e modellazione

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



Attenzione: l'importanza di un modello non è in un particolare diagramma, ma è nell'idea che il diagramma intende comunicare

Gestis

Scena

Un cliente

restituisce

registri.

Scena

Se il cliente

transazione di rimborso sul relativo conto di credito è stata respinta, allora il cliente viene informato e viene rimborsato in contanti.

Se ...

A P S Applicare UML

Unified Modeling Language (UML) è una notazione standard per descrivere software OO

- UML è una notazione visuale per scrivere (disegnare) modelli

Nella modellazione, una notazione è utile – addirittura 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

A P S Applicare UML

I segreti della modellazione secondo gli analisti ed i progettisti esperti

- lo scopo primario della modellazione è comprendere – 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

A P S Che cosa è importante?



A P S Che cosa è importante?

Conoscere come leggere e disegnare diagrammi UML male è – se esperti della progettazione e dei pattern non si è

La capacità di progettare gli oggetti e le architetture importante è – e non i diagrammi UML o gli strumenti CASE



Tuttavia, in pratica, anche un linguaggio per esprimere modelli e progetti necessario è

A P S Progettazione a oggetti: principi e pattern

Problemi nello sviluppo di software OO

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

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**

A P S Progettazione a oggetti: principi e pattern

Problemi nello sviluppo di software OO

- quali sono i problemi?
- che ruolo ha l'oggetto/classa?
- come si progettano gli oggetti?
- come si progettano le relazioni?
- come si progettano le interfacce?
- come si progettano le eccezioni?
- come si progettano le librerie?
- come si progettano i test?
- come si progettano i deployment?
- come si progettano i deployment?

soluzione a un problema progettuale, semplice ed elegante, che codifica principi di progettazione esemplari, già applicati e verificati in pratica

impegno a eseguire un compito o di conoscere un'informazione

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**

A P S 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


A P S Analisi dei requisiti e casi d'uso

L'enfasi del corso è sull'OOA/D

- viene presentata anche l'attività preliminare dell'**analisi dei requisiti**
- soprattutto con riferimento ai **casi d'uso** (una notazione per la modellazione dei requisiti)

A P S Sviluppo iterativo, modellazione agile e UP agile

E' utile descrivere l'analisi dei requisiti e l'OOA/D nel contesto di un **processo per lo sviluppo del software**



un approccio per lo sviluppo del software, che ne organizza ruoli, attività e prodotti, tempi e modi

A P S Sviluppo iterativo, modellazione agile e UP agile

E' utile descrivere l'analisi dei requisiti e l'OOA/D nel contesto di un **processo per lo sviluppo del software**

- **UP agile**
 - UP – Unified Process
 - agile = leggero e flessibile
- perché? è un processo di sviluppo **iterativo**
- i concetti presentati sono rilevanti anche per molti altri approcci e processi

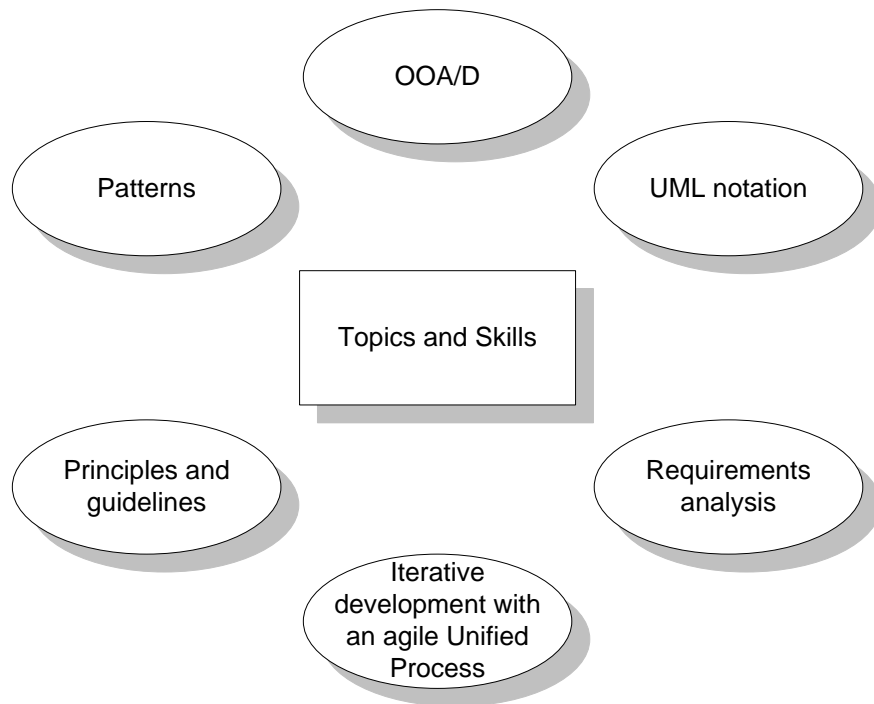
A P S Sviluppo iterativo, modellazione agile e UP agile

E' utile descrivere l'analisi dei requisiti e l'OOA/D nel contesto di un **processo per lo sviluppo del software**

- **UP agile**

E' importante capire non solo

- come fare analisi e
- progettazione, ma anche che occi e relazione c'è con le altre attività dello sviluppo del software – requisiti, programmazione, verifica e validazione, ...



Obiettivo del corso è sostenere lo sviluppo di software di qualità

- che cosa serve per sviluppare software di qualità?
 - un insieme di strumenti (modelli e metodi) coerenti per ciascuna delle attività da svolgere (requisiti, analisi, progettazione, ...)
 - casi d'uso, OOA, OOD
 - principi, linee guida e pattern
 - una notazione
 - un contesto in cui applicare al meglio questo insieme di strumenti
 - un processo iterativo per lo sviluppo del software

A P S 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**
- capacità importante sia se si progetta in modo visuale che se si scrive direttamente codice

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**

- l'analisi non è interessata direttamente alle soluzioni del problema

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

- la progettazione 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)

A P S Che cosa sono analisi e progettazione

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

- l'analisi non è interessata direttamente alle soluzioni del problema

La
sod

Attenzione: **analisi e**

- **progettazione hanno obiettivi** azione

diversi, perseguiti in modo

Do

diverso – tuttavia sono attività

- **fortemente sinergiche e** one)

inseparabili

A P S 1.4 Che cosa sono analisi e progett. OO

L'**analisi orientata agli oggetti (OOA)** è basata principalmente sull'identificazione dei concetti nel dominio del problema – e su una loro descrizione ad oggetti

- ad es., nel caso di un sistema informativo per voli, **Plane**, **Flight** e **Pilot**

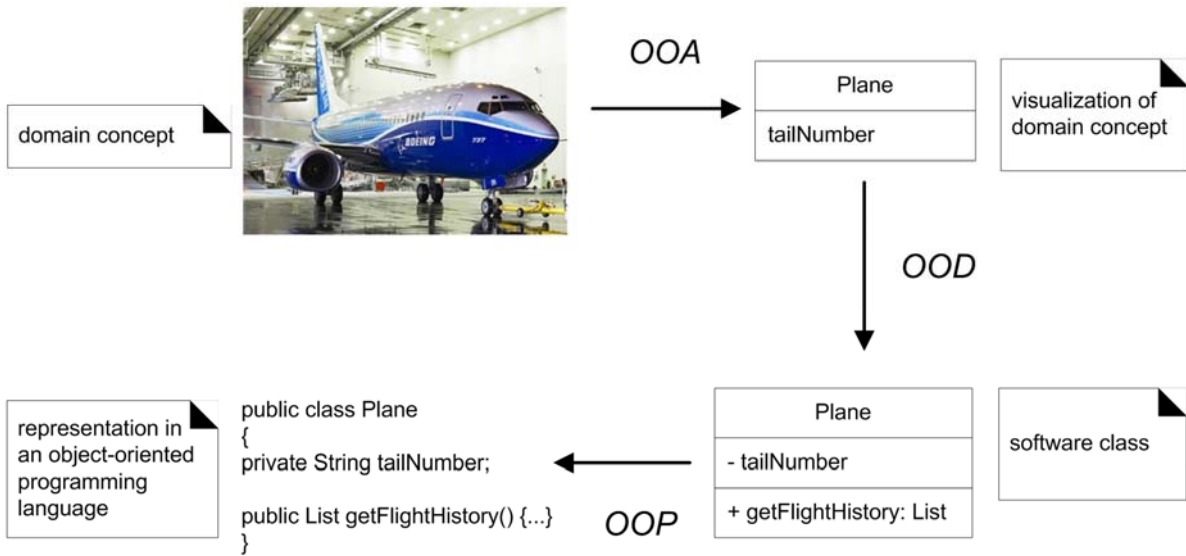
La **progettazione orientata agli oggetti (OOD)** è basata principalmente sulla definizione e la caratterizzazione di una comunità di oggetti software – e del modo in cui questi collaborano per soddisfare i requisiti

- un oggetto software **Plane** ha un attributo **tailNumber** e un metodo **getFlightHistory**

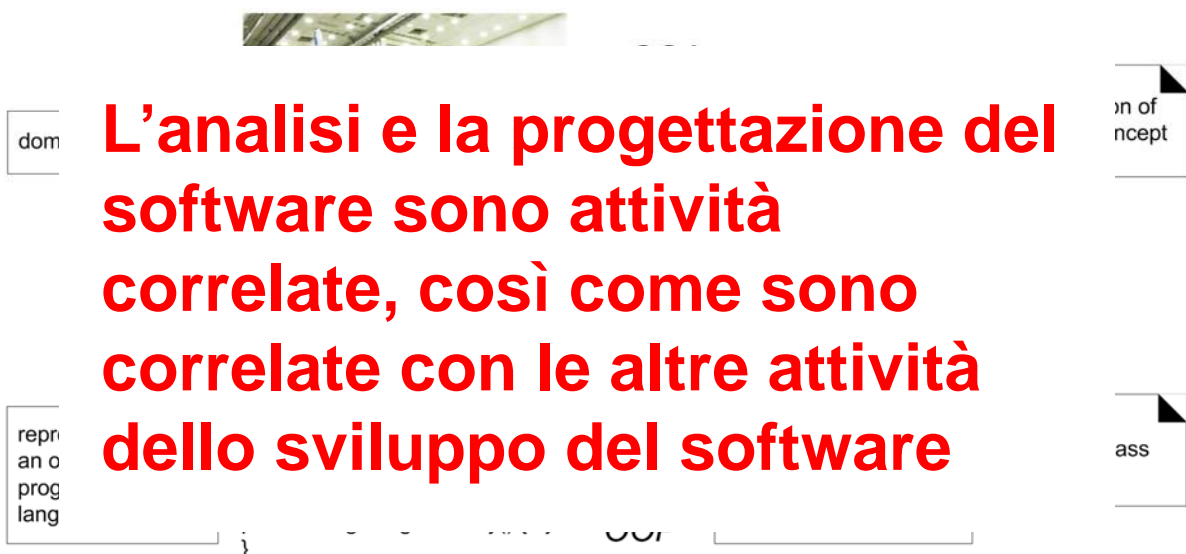
Le classi scelte durante l'OOD vengono implementate durante la **programmazione orientata agli oggetti (OOP)**

- una classe **Plane** in Java

A P S Analisi e progettazione OO



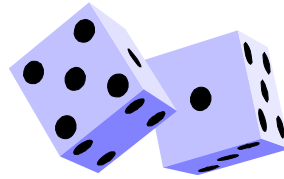
A P S Analisi e progettazione OO



A P S 1.5 Un breve esempio

Gioca una partita a dadi (Play a Dice Game)

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



Attività

- requisiti
 - definizione dei casi d'uso
- analisi
 - definizione di un modello di dominio
- progettazione
 - definizione dei diagrammi di interazione
 - definizione dei diagrammi delle classi di progetto

A P S Definizione dei casi d'uso

L'analisi dei requisiti comprende una descrizione delle funzionalità del sistema da sviluppare

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

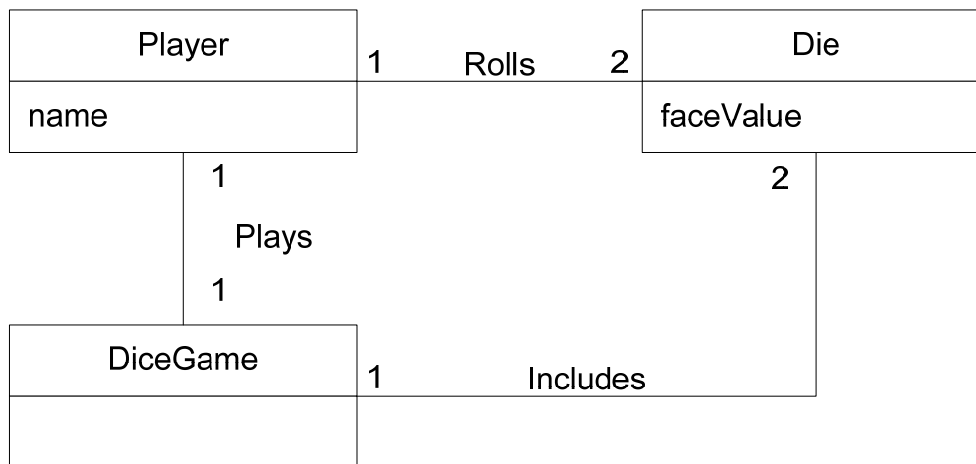
Caso d'uso per *Gioca una partita a dadi*

- 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

A P S Definizione di un modello di dominio

L'OOA è interessata a descrivere il dominio di interesse sulla base di una classificazione (concettuale) a oggetti

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



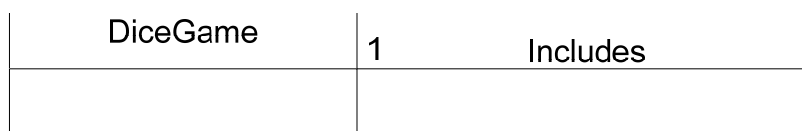
A P S Definizione di un modello di dominio

L'OOA è interessata a descrivere il dominio di interesse sulla base di una classificazione (concettuale) a oggetti

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



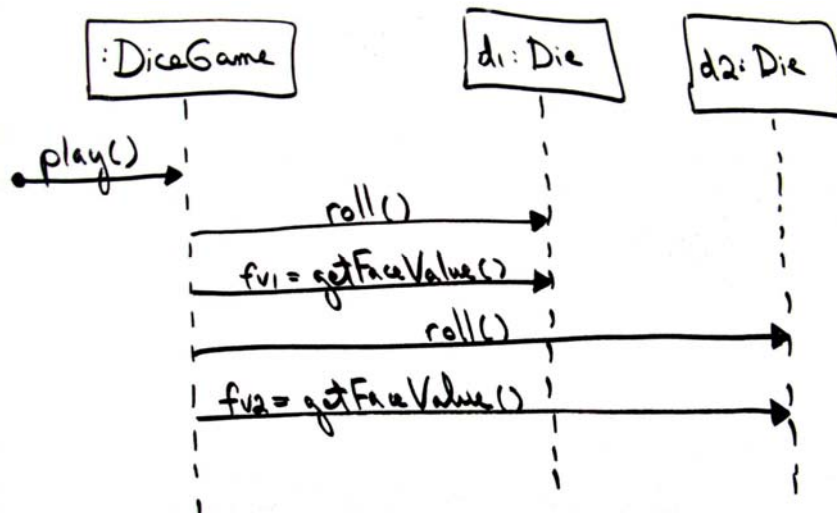
Attenzione: **qui si parla di oggetti del mondo reale**



A P S 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



A P S 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



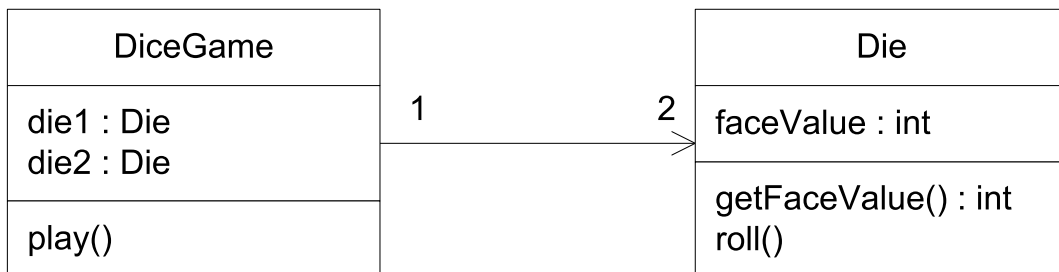
Attenzione: **le classi hanno gli stessi nomi – ma adesso si parla di oggetti software**



A P S Diagrammi delle classi di progetto

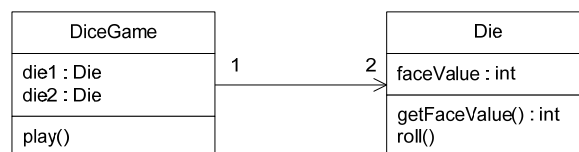
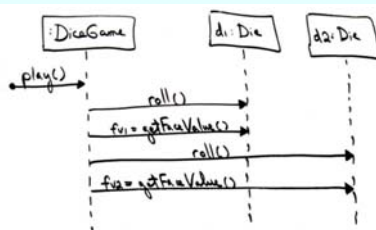
I diagrammi di interazione sono una vista dinamica sugli oggetti software

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



Attenzione: **anche qui si parla di oggetti software**

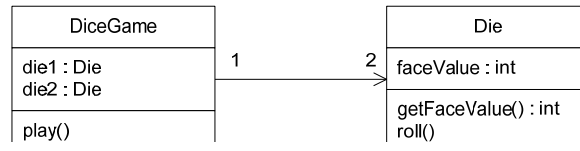
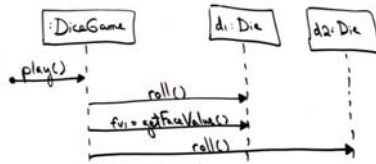
A P S Dal progetto al codice



```
class DiceGame {
    private Die die1;
    private Die die2;
    public DiceGame() {
        die1 = new Die();
        die2 = new Die();
    }
    public void play() {
        int fv1, fv2;
        die1.roll();
        fv1 = die1.getFaceValue();
        die2.roll();
        fv2 = die2.getFaceValue();
        ...
    }
}
```

```
class Die {
    private int faceValue;
    public Die() {
        ...
    }
    public void roll() {
        ...
    }
    public int getFaceValue() {
        return faceValue;
    }
}
```

A P S Dal progetto al codice



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

```
class
priv
priv
pub
d
d
}
pub
ir
d
fv1 = die1.getFaceValue();
die2.roll();
fv2 = die2.getFaceValue();
...
}
```

35

Analisi e progettazione orientata agli oggetti

Luca Cabibbo - A.P.S

A P S 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 (nella versione UML 2) – ma solo in modo parziale

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

36

Analisi e progettazione orientata agli oggetti

Luca Cabibbo - A.P.S

A P S Tre punti di vista per applicare UML

UML definisce alcuni tipi “grezzi” di diagrammi – ma non impone nessun “punto di vista” predefinito relativamente al loro uso

Tre *punti di vista* sull'applicazione di UML [Fowler, UML Distilled]

- punto di vista **concettuale**
 - i diagrammi sono interpretati come descrizioni di oggetti del mondo reale o nel dominio di interesse
- punto di vista della **specifica (software)**
 - i diagrammi descrivono componenti software, a livello della loro interfaccia e in modo indipendente dalle possibili implementazioni (e.g., Java o C#)
- punto di vista dell'**implementazione (software)**
 - i diagrammi schematizzano l'implementazione del software, con riferimento a una particolare tecnologia o linguaggio

A P S Tre punti di vista per applicare UML

UML definisce alcuni tipi “grezzi” di diagrammi – ma non impone nessun “punto di vista” predefinito relativamente al loro uso

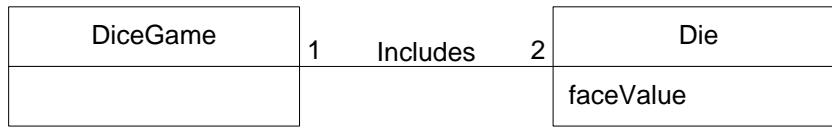
Tre *punti di vista* sull'applicazione di UML

- punto di vista **concettuale**
 - i diagrammi sono interpretati come descrizioni di oggetti del mondo reale o nel dominio di interesse
- punto di vista della **specifica (software)**
 - i diagrammi descrivono componenti software, a livello della loro interfaccia e in modo indipendente dalle possibili implementazioni (e.g., Java o C#)
- punto di vista dell'**implementazione (software)**
 - i diagrammi schematizzano l'implementazione del software, con riferimento a una particolare tecnologia o linguaggio

è il punto di vista prevalente nell'analisi

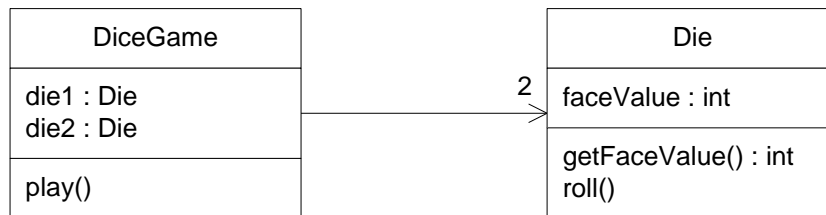
è il punto di vista prevalente nella progettazione

A P S Punti di vista differenti



Conceptual Perspective
(domain model)

Raw UML class diagram notation used to visualize real-world concepts.



Specification or Implementation Perspective
(design class diagram)

Raw UML class diagram notation used to visualize software elements.

A P S 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 di progetto**
 - specifica di un componente software – nel modello di progetto
- **classe di implementazione**
 - l'implementazione di un componente software – ad es., una classe Java
- **classe software**
 - classe di progetto o classe di implementazione
- **classe**
 - classe concettuale o classe software

A P S Tre modi per applicare UML

[Fowler, UML Distilled] descrive tre *modi* per applicare UML

- **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**

A P S Tre modi per applicare UML

[Fowler, UML Distilled] descrive tre *modi* per applicare UML

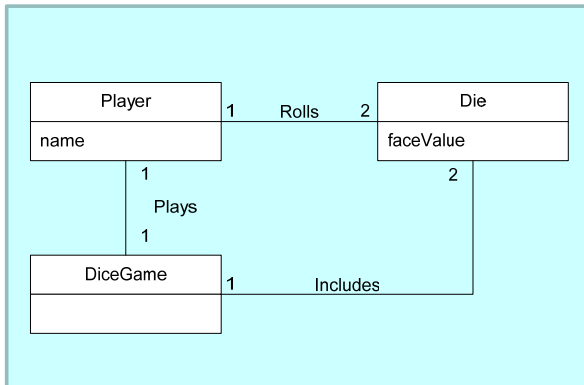
- **UML come abbozzo** (sketch)

- 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**

A P S 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

Player – name
Die – faceValue
DiceGame

Relazioni

il **DiceGame** è giocato da un **Player**
il **DiceGame** comprende due **Die**
il **Player** tira i due **Die**

Quale dei due modelli è più facilmente comprensibile?