# Programmazione Orientata agli Oggetti in Linguaggio Java

## Ruoli e Responsabilità: Il Processo di Sviluppo

versione 1.5



**G. Mecca – Università della Basilicata – mecca@unibas.it**



## Sommario

- I Passi Principali
- Analizzare le Specifiche
- Modello Concettuale
- Scegliere i Componenti
- Scrivere i Metodi

# I Passi Principali


- A questo punto
  - ⇨possiamo schematizzare i passi fondamentali del processo di sviluppo
- Attenzione
  - ⇨il processo di sviluppo è completamente diverso rispetto a quello della programmazione procedurale
  - ⇨e richiede tecniche e linee guida nuove





# I Passi Principali

- Nella programmazione procedurale
  - ⇨il metodo era centrato attorno all'obiettivo di scrivere i sottoprogrammi
- Nel caso della programmazione a oggetti
  - ⇨vari problemi diversi
  - ⇨scegliere i componenti
  - ⇨attribuire le responsabilità
  - ⇨scrivere il codice dei metodi (problema simile a quello classico)

# I Passi Principali

- Per ora
  - ⇨una presentazione semplificata
- I Passi principali
  - ⇨analizzare le specifiche
  - ⇨costruire il modello concettuale
  - ⇨scegliere i componenti
  - ⇨attribuire le responsabilità ai componenti
  - ⇨scrivere il codice dei metodi





# I Passi Principali

- Esecuzione iterativa
  - ⇨al solito si suggerisce di eseguire i passi in modo incrementale
  - ⇨ritornando sulle proprie scelte ripetutamente
  - ⇨compilando ed eseguendo progressivamente il codice
- Nel seguito
  - ⇨alcune linee guida e una ridiscussione degli esempi presentati

# Analizzare le Specifiche

- Al solito
  - ⇨il passo iniziale è l'analisi delle specifiche
- Lo strumento
  - ⇨i casi d'uso
  - ⇨ciascun caso d'uso è un insieme di "storie" che descrivono scenari di utilizzo dell'applicazione
  - ⇨l'analisi delle specifiche consiste per cominciare nella descrizione dei casi d'uso





# Modello Concettuale

- Il passo successivo
  - ⇨la specifica descrive la realtà di interesse, ovvero il "dominio applicativo"
  - ⇨è necessario sviluppare l'insieme di componenti dell'applicazione
- Passaggio non banale
  - ⇨da un insieme di documenti (i casi d'uso)
  - ⇨è necessario sviluppare un insieme di componenti software

# Modello Concettuale

- Idea
  - ⇨la realtà di interesse descrive una serie di concetti
  - ⇨gli oggetti software devono rappresentare nell'applicazione questi concetti della realtà
  - ⇨per potere sviluppare l'applicazione, per cominciare è necessario comprendere la natura di questi concetti





# Modello Concettuale

- Modello concettuale
  - ⇨diagramma che descrive i concetti di un dominio applicativo e le loro relazioni
- Rispetto ai diagrammi visti finora
  - ⇨si tratta di uno strumento di analisi
  - ⇨serve a capire le specifiche
  - ⇨prima ancora di cominciare a pensare allo sviluppo ed ai dettagli implementativi

# Modello Concettuale

- Costruire il modello concettuale
  - ⇨vuol dire identificare una serie di concetti (es: segmento, punto)
  - ⇨identificare le caratteristiche di questi concetti (es: ascissa, ordinata)
  - ⇨identificare le relazioni tra questi concetti (es: un segmento ha due vertici)
  - ⇨e le eventuali tassonomie





# Modello Concettuale

- Sintassi grafica per costruire il modello
  - ⇨ne esistono varie
  - ⇨esempio: diagrammi Entità-Relazione
- Più recentemente
  - ⇨per questo scopo viene utilizzato UML
- In particolare
  - ⇨un diagramma concettuale di UML

# Modello Concettuale

- Diagramma concettuale di UML
  - ⇨diagramma delle classi mantenuto ad un livello di astrazione molto alto
  - ⇨un concetto è rappresentato come una classe con nome e proprietà
  - ⇨una relazione tra concetti è rappresentato come una associazione con le relativa cardinalità
  - ⇨i concetti possono essere in gerarchie





# Modello Concettuale

**ATTENZIONE**
alla differenza tra modello concettuale e diagramma delle classi

- Attenzione a non fare confusione
  - ⇨un diagramma concettuale NON descrive la struttura di un'applicazione software
  - ⇨i concetti NON sono classi
  - ⇨i concetti NON hanno metodi
  - ⇨i concetti NON hanno modificatori di visibilità
  - ⇨i concetti NON hanno relazioni di dipendenza

# Modello Concettuale

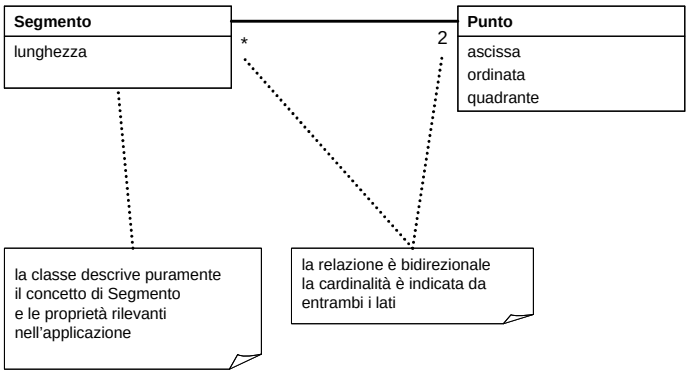
- Un esempio: segmenti
  - ⇨un caso d'uso: "Utente analizza segmento"
- "Utente analizza segmento"
  - ⇨l'utente fornisce le coordinate degli estremi del segmento
  - ⇨il sistema riassume i dati del segmento fornito
  - ⇨il sistema calcola e stampa la lunghezza del segmento e il quadrante degli estremi





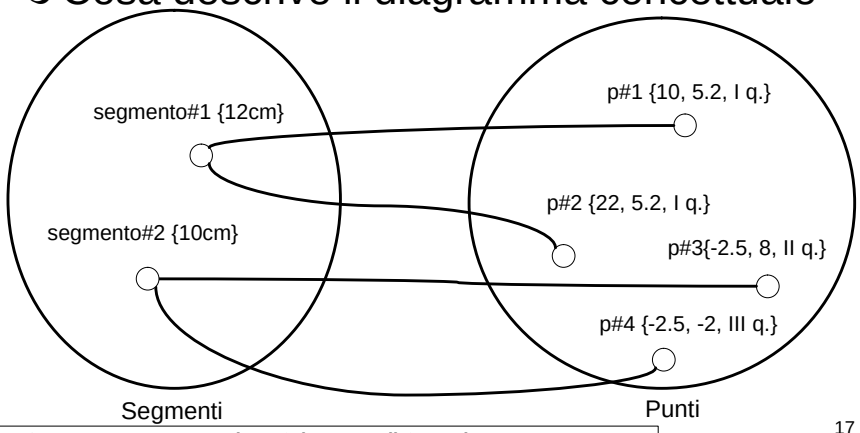
# Modello Concettuale

- Il diagramma concettuale



# Modello Concettuale

- Cosa descrive il diagramma concettuale

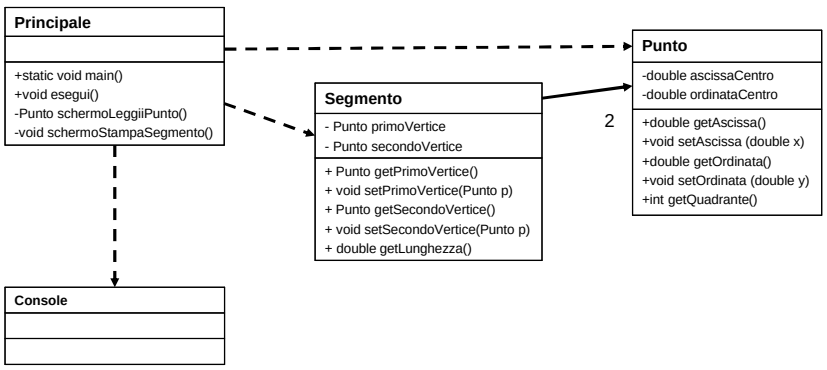






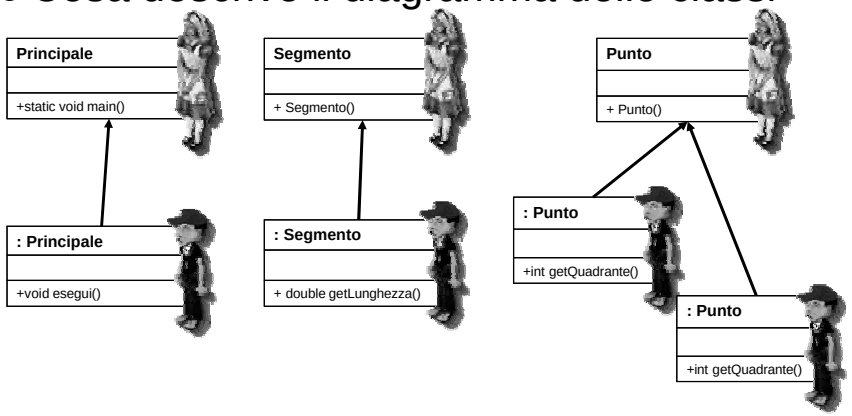

# Modello Concettuale

- Il diagramma delle classi finale (vers. a)



# Modello Concettuale

- Cosa descrive il diagramma delle classi








# Modello Concettuale

- Quindi: il diagramma delle classi di UML
  - ⇨si può usare sia come diagramma di documentazione del codice
  - ⇨sia come modello concettuale
- Perchè questa ambiguità ?
  - ⇨perchè UML è costruito per essere usato a diversi livelli, secondo diversi punti di vista
  - ⇨e per descrivere aspetti diversi dell'applicazione

## Modello Concettuale

**ATTENZIONE**
alla varie accezioni
del termine classe

- Il termine classe in UML: varie accezioni
  - ⇨nel codice: classe come componente software dell'applicazione
  - ⇨nel diagramma delle classi: classe come descrizione/specifica grafica del codice di un componente software
  - ⇨nel diagramma concettuale: classe come concetto, ovvero insieme di oggetti della realtà di interesse





## Modello Concettuale

- Costruire il modello concettuale
  - ⇨il punto di partenza sono i casi d'uso
  - ⇨è necessario leggerli con attenzione
- I passi principali
  - ⇨I passo: individuare i concetti/classi e le relative proprietà
  - ⇨II passo: individuare le associazioni e le relative cardinalità

# Modello Concettuale

- I passo: individuare i concetti
  - ⇨individuare i termini ricorrenti nella specifica
  - ⇨individuare le proprietà di ciascun concetto individuato rilevanti ai fini dell'applicazione
- II passo: individuare le associazioni
  - ⇨individuare le coppie di termini che sono in relazione nella specifica
  - ⇨cercare di risalire alle associazioni tra i concetti corrispondenti ed alle cardinalità



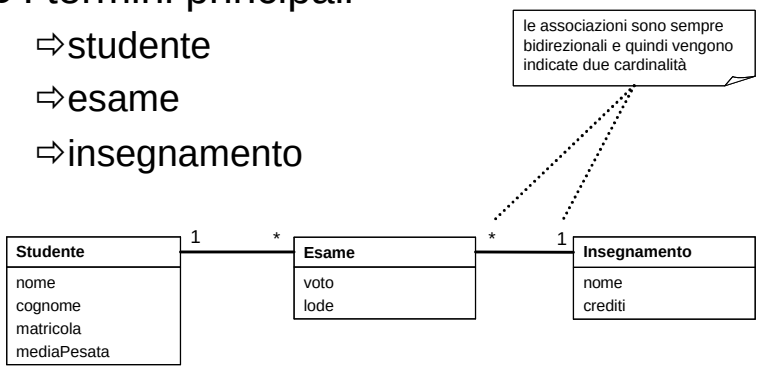


# Modello Concettuale

- Esempio: la media pesata
- Caso d'uso: "Studente calcola media"
  - ⇨lo studente fornisce i suoi dati personali
  - ⇨lo studente fornisce i dati di tutti i suoi esami universitari (insegnamento, voto, lode)
  - ⇨se c'è almeno un esame il sistema calcola la media pesata rispetto ai crediti e la visualizza altrimenti termina

# Modello Concettuale

- I termini principali
  - studente
  - esame
  - insegnamento








# Modello Concettuale

**ATTENZIONE**
ad evitare questo errore

- Un errore frequente
  - costruire il diagramma concettuale pensando già alle classi dell'applicazione
  - non è questa la finalità del modello
- Esempio
  - le classi di controllo non rappresentano concetti
  - i tipi delle proprietà e tutti i dettagli implementativi sono irrilevanti

# Modello Concettuale

**ATTENZIONE**
alla differenza di uso delle associazioni

- Attenzione alle associazioni
  - ⇨nel diagramma concettuale le associazioni NON corrispondono alla presenza di proprietà nelle classi
  - ⇨di conseguenza nella classe/concetto Esame non c'è una proprietà che corrisponde ad un riferimento all'Insegnamento
  - ⇨analogamente nella classe/concetto Studente non c'è una proprietà che corrisponde alla collezione di rif. agli esami





# Modello Concettuale

- Fino a questo punto
  - ⇨il lavoro è stato fatto sulla carta
  - ⇨il risultato dovrebbe essere una maggiore comprensione del dominio e della specifica
- Il passo successivo
  - ⇨lo sviluppo del codice
  - ⇨a partire dalla specifica e dal modello concettuale è necessario sviluppare i componenti

# Modello Concettuale

- Nota
  - ⇨il modello concettuale non serve solo a sviluppare le classi
  - ⇨ma anche ad altri scopi
- Gli altri scopi
  - ⇨funge da strumento di documentazione delle specifiche per la comunicazione con il cliente
  - ⇨è il punto di partenza per lo sviluppo delle basi di dati (>>)





# Scegliere i Componenti

- Il primo passo
  - ⇨scegliere le classi
- Il principio guida
  - ⇨rispettare gli strati applicativi
- Per lo strato di modello e persistenza
  - ⇨il punto di partenza è il modello concettuale
- Per lo strato di interfaccia e controllo
  - ⇨il punto di partenza sono i casi d'uso

# Scegliere i Componenti

- Scegliere i componenti del modello
  - ⇨il punto di partenza è il modello concettuale
  - ⇨ogni concetto è candidato a diventare una classe dello strato di modello&persistenza
- Nel nostro esempio
  - ⇨tre classi candidate
  - ⇨Studente, Esame, Insegnamento
  - ⇨non necessariamente tutte e tre sopravvivranno fino alla fine





# Scegliere i Componenti

- Il passo successivo
  - ⇨attribuire le proprietà alle classi selezionate
- Proprietà dei componenti del modello
  - ⇨discendono dalle proprietà individuate nel modello concettuale
  - ⇨e dalle associazioni individuate tra i concetti

# Scegliere i Componenti

- Implementazione delle proprietà
  - ciascuna proprietà di un concetto è candidata a diventare una proprietà della classe corrispondente
  - bisogna individuare il tipo di dato ed il livello di visibilità; es: private String nome
  - alcune proprietà del concetto possono essere dati da calcolare e quindi dare luogo a metodi; es: mediaPesata





# Scegliere i Componenti

- Implementazione delle associazioni
  - le associazioni individuate nel modello concettuale rappresentano esclusivamente legami tra i concetti
  - è necessario tradurle in proprietà delle classi
  - per farlo, preliminarmente, è necessario individuare i versi significativi

# Scegliere i Componenti

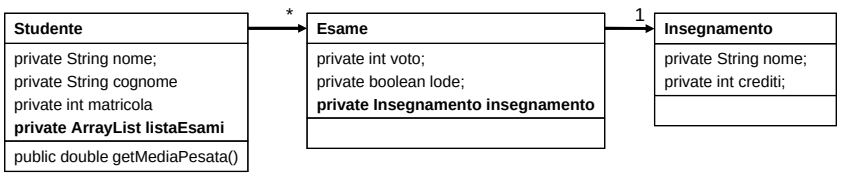
- Individuare i versi
  - ⇨non necessariamente tutte le associazioni devono restare bidirezionali
- Esempio
  - ⇨nel nostro caso è indispensabile risalire dallo studente ai suoi esami, ma non il viceversa
  - ⇨analogamente, è necessario risalire dall'esame al suo insegnamento e non viceversa





# Scegliere i Componenti

- Implementare le associazioni
  - ⇨le associazioni 1 ad 1 diventano riferimenti
  - ⇨le associazioni 1 a molti diventano liste
- Il punto della situazione



# Scegliere i Componenti

- In concreto
  - ⇨questo vuol dire che posso cominciare a sviluppare il codice

```
public class Studente {
  private String nome;
  private String cognome;
  private int matricola;
  private ArrayList listaEsami =
        new ArrayList();
  ...
}
```

```
public class Esame {
  private int voto;
  private boolean lode;
  private Insegnamento insegnam;
  ...
}
```

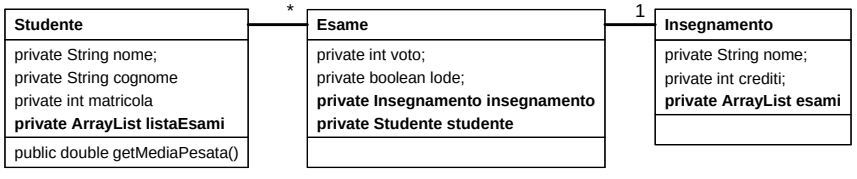
```
public class Insegnamento {
  ...
}
```





# Scegliere i Componenti

- Nota
  - ⇨se avessi deciso di mantenere le associazioni bidirezionali avrei avuto delle proprietà aggiuntive



# Scegliere i Componenti

- Per lo strato di interfaccia&controllo
  - ⇨è necessario individuare gli schermi ed il flusso di controllo dei vari casi d'uso
  - ⇨è sempre necessaria una classe di controllo Principale
  - ⇨se ci sono casi d'uso complessi è possibile prevedere ulteriori classi di controllo secondarie



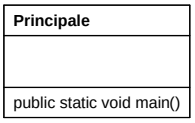


# Scegliere i Componenti

- Differenza tra le classi dei due strati
  - ⇨mentre le classi del modello discendono dai concetti della realtà di interesse
  - ⇨le classi di interfaccia e controllo hanno una funzione puramente implementativa
- Terminologicamente
  - ⇨queste ultime si definiscono "artefatti implementativi" ("pure fabrications")

# Scegliere i Componenti

- Proprietà delle classi di controllo
  - ⇨normalmente le classi di interfaccia e controllo non mantengono proprietà
  - ⇨è opportuno privilegiare l'uso di variabili locali
- Nel nostro caso
  - ⇨sembra sufficiente un'unica classe di controllo
  - ⇨per ora senza proprietà








# Scrivere i Metodi

- Scegliere i metodi
  - ⇨è il passo fondamentale nell'attribuzione delle responsabilità ai componenti
- I principi
  - ⇨sono quelli che abbiamo visto
  - ⇨corretta organizzazione degli strati applicativi
  - ⇨massimizzare la coesione
  - ⇨minimizzare l'accoppiamento

# Scrivere i Metodi

- La parte semplice
  - costruttori
  - metodi set/get per le proprietà
- Nota
  - non necessariamente tutte le proprietà devono avere sia un metodo set che get
  - vale un principio di minimalità: aggiungere un metodo solo se necessario





# Scrivere i Metodi

- I metodi realmente interessanti
  - sono quelli che rappresentano la reale "logica applicativa"
  - spesso corrispondono ai "verbi" utilizzati in congiunzione con i nomi nella specifica
- Metodo
  - ripercorrere i casi d'uso ed individuare i metodi necessari

# Scrivere i Metodi

- Simulazione di scenari
  - ⇨vengono analizzati vari scenari dei casi d'uso dell'applicazione
  - ⇨per ciascuno scenario, si "simula" a voce la sequenza delle chiamate tra i componenti individuati
  - ⇨ogni volta che viene individuata una nuova operazione significativa, questa viene attribuita ad una delle classi





# Scrivere i Metodi

- Per le classi di controllo
  - ⇨i metodi corrispondono agli schermi dei casi d'uso ed alle operazioni di controllo del flusso
- Per le classi del modello
  - ⇨i metodi vengono attribuiti seguendo il principio di "rivolgersi all'esperto"
  - ⇨un metodo dovrebbe essere eseguito dal componente che ha le informazioni per farlo

# Scrivere i Metodi

- Attenzione
  - ⇨come al solito il processo è basato sulla scrittura del codice
  - ⇨ci sono momenti in cui si pensa
  - ⇨ci sono momenti in cui si scrive
  - ⇨ci sono momenti in cui si compila
  - ⇨ci sono momenti in cui si verifica
  - ⇨i cicli vanno ripetuti frequentemente



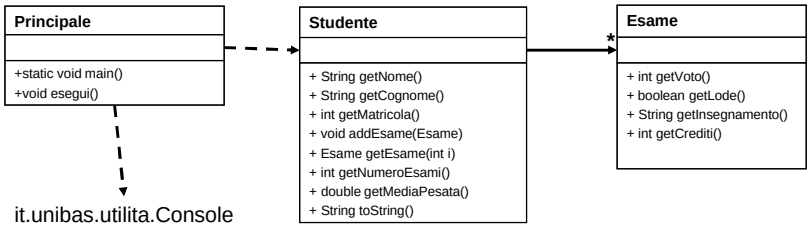


# Scrivere i Metodi

- Verifiche
  - ⇨è necessario sottoporre a verifica periodica i risultati ottenuti
  - ⇨non sempre le classi individuate inizialmente sono quelle corrette
  - ⇨diffidare di classi con troppe responsabilità (probabilmente sdoppiabili)
  - ⇨diffidare di classi troppo semplici o senza responsabilità (probabilmente eliminabili)

# Scrivere i Metodi

- Nel nostro esempio
  - ⇨la classe Insegnamento sembra evitabile
  - ⇨non avrebbe reali responsabilità di logica applicativa








# Scrivere i Metodi

- Di conseguenza
  - ⇨il processo è significativamente più articolato del processo di sviluppo per la programmazione procedurale
  - ⇨detto metodo di sviluppo “in piccolo”
- Il passo fondamentale
  - ⇨selezionare i componenti
  - ⇨attribuire le responsabilità

# Scrivere i Metodi

- Il metodo di sviluppo in piccolo
  - ⇨serve comunque per la scrittura del codice dei metodi complessi
  - ⇨un problema riconducibile a quello classico
  - ⇨ogni metodo può essere interpretato come un problema da risolvere
  - ⇨scomponibile in vari sottoproblemi (ulteriori metodi)
  - ⇨riutilizzare tecniche algoritmiche note





# Scrivere i Metodi

- Valgono le linee guida tradizionali
  - ⇨scrivere codice in modo semplice e leggibile
  - ⇨scrivere metodi brevi
  - ⇨scegliere identificatori chiari
  - ⇨adottare convenzioni di stile
- Idea
  - ⇨il codice deve essere “autodocumentante”
  - ⇨leggibile chiaramente a mesi di distanza
  - ⇨tanto da rendere inutili i commenti

# Scrivere i Metodi

- Ma...
  - ⇨in generale nella programmazione a oggetti c'è maggiore enfasi sulla qualità
  - ⇨gestione delle eccezioni (>>)
  - ⇨test di regressione (>>)
  - ⇨correzione ("debugging") (>>)





# Riassumendo

- I Passi Principali
- Analizzare le Specifiche
- Modello Concettuale
- Scegliere i Componenti
- Scrivere i Metodi

# Termini della Licenza