

---

---

# Unified Modeling Language

marrara@elet.polimi.it

Adattamento di  
materiale di Luciano  
Baresi



1

---

## Concetto di oggetto

---

- **Definizione:** un oggetto è un concetto, un'astrazione o una cosa ed ha un senso in un particolare contesto applicativo
- **Ha due scopi:**
  - rappresentare aspetti del mondo reale
  - fornire la base per un'implementazione al calcolatore
- **tutti gli oggetti hanno un'identità e sono distinguibili**

# Concetto di classe

---

- Una classe definisce un gruppo di oggetti con simili proprietà (attributi) e comportamenti comuni (operazioni o servizi)

```
class persona {  
  //attributi  
  stringa nome;  
  stringa cognome;  
public: //servizi  
  void presentati();  
}
```

variabili

servizi o metodi

→ "Buongiorno, sono:  
<nome> <cognome>"

# Oggetti e classi

---

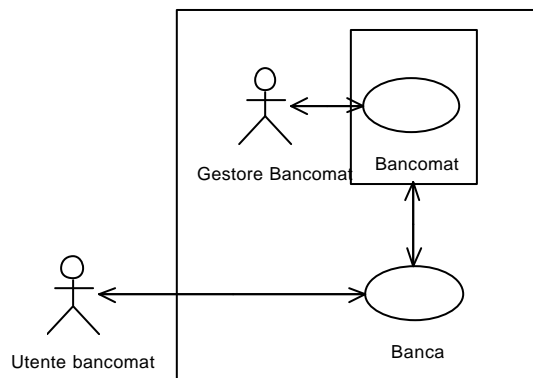
- In un linguaggio di programmazione orientato agli oggetti
  - La classe definisce un nuovo tipo
  - L'oggetto è un'istanza di una classe

```
class Persona {...};  
  
void main()  
{  
  Persona p;  
}
```

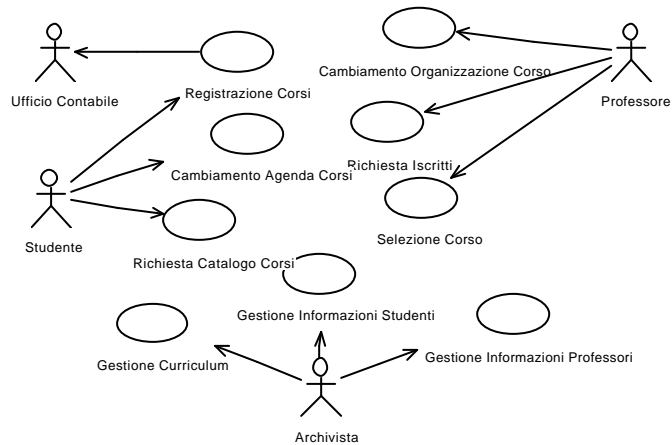
# UML: Comportamento del sistema

- Si definisce come il sistema agisce e reagisce
  - Ciò che è visibile all'esterno (scatola nera)
- Il comportamento del sistema è definito attraverso gli *use case*
  - Descrivono il sistema, l'ambiente e le relazioni fra sistema e ambiente
  - Un *attore* è *qualcuno* (utente) o *qualcosa* (sistemi esterni - hardware) che:
    - Scambia informazioni con il sistema
    - Fornisce input o riceve output dal sistema

## Attori



# Use case



# Documentazione

- *Titolo:* Computo tasse
- *Autori:* Pluto e Topolino
- *Descrizione:* Quando le iscrizioni di uno studente sono state accettate si inviano le informazioni all'ufficio contabile
- *Attori:* Ufficio contabile
- *Pre-condizioni:* Studente registrato
- *Post-condizioni:* Non identificate
- *Scenari:* Poi

# Scenari

---

- Uno *scenario* è un'istanza di uno *use case*
  - Definisce gli eventi in un caso particolare di esecuzione del programma
- Uno scenario è solitamente definito in linguaggio naturale. Si rispetta l'ordine temporale
  - Gli eventi devono essere semplici e autoesplicativi (non devono includere alternative complesse)

# Un esempio

---

## Iscrizione ai corsi:

Paperino Paolino sceglie i corsi:informatica, fisica, analisi e disegno

Come corsi alternativi sceglie: fotografia e giornalismo

I corsi scelti sono immessi nel sistema di registrazione

Lo studente viene aggiunto ai primi 3 corsi principali

Il quarto corso non è disponibile

**SE** la prima alternativa è disponibile, lo studente viene aggiunto al corso

**ALTRIMENTI SE** la seconda alternativa è disponibile, lo studente viene aggiunto al corso

**ALTRIMENTI** si notifica allo studente che non è iscritto ad un numero sufficiente di corsi

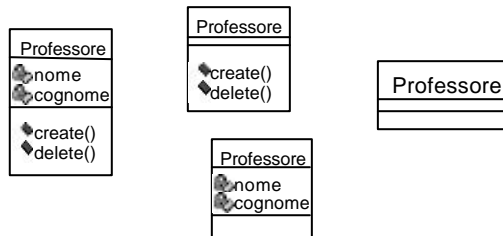
Si crea la cartella corsi dello studente

Si mandano le informazioni all'ufficio contabile

# Classi

---

- In UML una classe è composta da tre parti
  - Il nome
  - Gli attributi (lo stato)
  - I metodi (il comportamento)



# Classi: come trovarle

---

- Evidenziare i nomi (oggetti) nella specifica del problema
- Considerare gli *use case* e gli *scenari*
  - Si definiscono gli use case
  - Si specificano alcuni use case in scenari
  - Si identificano gli oggetti propri di ogni scenario
- Si raggruppano gli oggetti in classi

# Identificare gli oggetti

---

- Paperino Paolino
- Corso
- Informatica
- Fisica
- Analisi
- Disegno
- Corso alternativo
- Fotografia
- Giornalismo
- Sistema di registrazione
- Studente
- Corso principale
- Alternativa
- Notifica
- Cartella studente
- Informazioni
- Selezione
- Ufficio contabile

Come filtrare ?

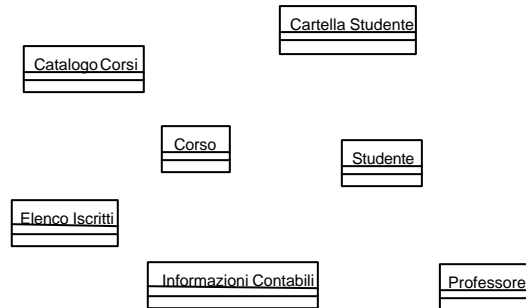
# Come raggruppare

---

- Esaminando gli oggetti identificati negli scenari, alcuni oggetti possono essere raggruppati
  - Tutti i corsi specifici appartengono alla classe **Corso**
- La lista cresce considerando diversi scenari
- Le classi identificate sono:
  - **Corso, Studente e Selezione**
  - Si devono raffinare:
    - **Iscritti, CartellaStudente e Informazioni...**

# Modello iniziale delle classi

---

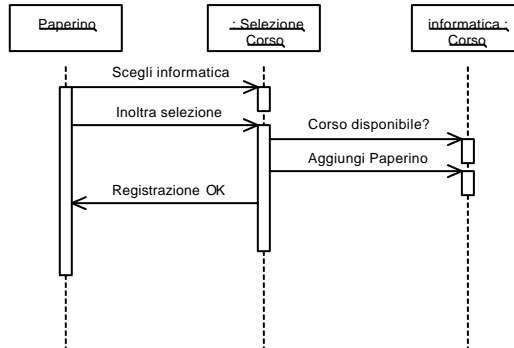


# Relazioni fra oggetti

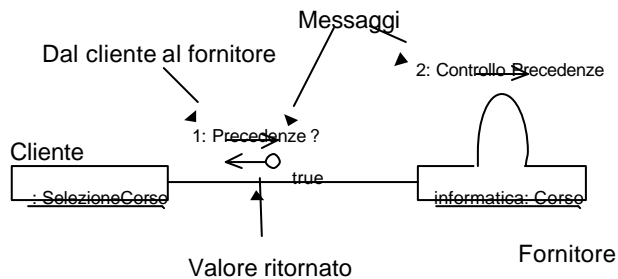
---

- In UML le interazioni (scambi messaggi) fra gli oggetti sono definiti tramite *interaction diagram*:
  - Oggetti
  - Scambio messaggi
  - La sequenza di invocazione dei messaggi
- UML offre due tipi di *interaction diagram*:
  - *sequence diagram*
  - *collaboration diagram*

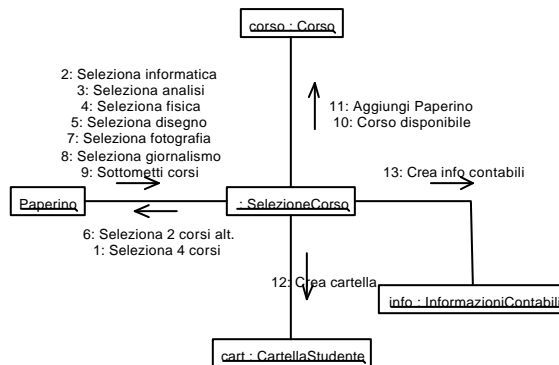
# Sequence diagram



# Collaboration diagram



# Collaboration diagram



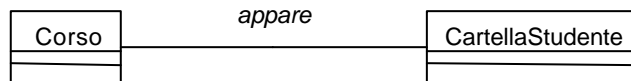
## Riassumendo ...

- Uno scenario può essere rappresentato per mezzo di un *interaction diagram*
- *Sequence diagram* e *collaboration diagram* rappresentano le stesse informazioni
  - Cambia la forma concreta
  - Si pone l'accento su aspetti leggermente diversi
- Commenti testuali possono fornire ulteriori dettagli al modello definito

# Associazioni

---

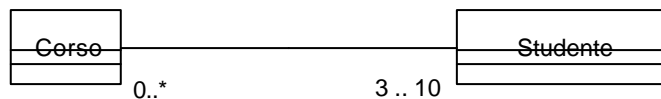
- Un' *associazione* deve avere un nome
- Il nome è solitamente un verbo (un' etichetta al centro della linea che definisce l'associazione)



# Associazioni: molteplicità

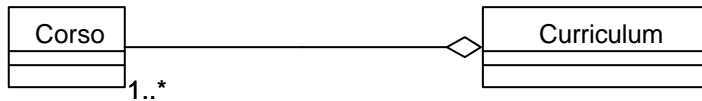
---

- La molteplicità dice:
  - Se l'associazione è obbligatoria oppure no?
  - Il numero minimo e massimo di oggetti che possono essere relazionati ad un altro oggetto



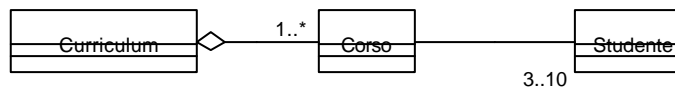
# Aggregazioni

- Le *aggregazioni* sono una forma particolare di associazione. Una parte è in relazione con un oggetto (**part-of**)



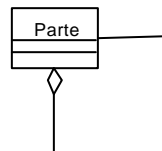
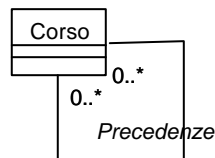
# Associazione o aggregazione

- Aggregazione se due oggetti sono “uniti” da una relazione **part-of**
- Associazione se due oggetti sono solitamente considerati indipendenti, anche se spesso “collegati”



# Associazioni riflesse

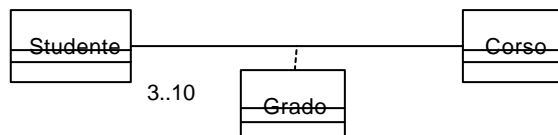
- Un'associazione è *riflessiva* se coinvolge oggetti della stessa classe
  - Indicano oggetti multipli della stessa classe collaborano in qualche modo



Aggregazione

# Attributi delle associazioni

- L'attributo si mette all'interno dell'icona di una classe e si collega l'icona all'associazione
- La classe attributo può avere proprietà specifiche
- È permesso un solo attributo per associazione



# Relazioni

---

- Tutti i sistemi sono composti da oggetti che collaborano per realizzare le funzionalità richieste
- Le relazioni sono: associazioni e aggregazioni
- Un'associazione definisce un canale di comunicazione bidirezionale fra le due classi
- La molteplicità definisce il numero di istanze che prendono parte alla relazione

# Operazioni e dominio

---

- **Tutte e sole** le operazioni rilevanti per il dominio applicativo
  - Una persona ha un comportamento diverso in funzione del punto di vista
    - *Bancario*
      - ricevePrestito, riceveCredito, apreConto
    - *Medico*
      - faEsame, prendeMedicina, vaOspedale

# Attributi

---

- Un attributo è una caratteristica della classe
- Gli attributi non hanno identità
- Ogni attributo deve essere definito in modo preciso
- *Attributi buoni* per Studente
  - Nome, Cognome, ...
- *Attributi cattivi*
  - CorsiScelti

# Attributi

---

- Gli attributi dipendono dal dominio
- Persona (ambito bancario)
  - nome, cognome, codiceFiscale, numeroConto
- Persona (ambito medico)
  - nome, cognome, allergie, peso, altezza

# Attributi derivati

---

- Quando si vuole avere un attributo calcolato e non memorizzato
  - Si usa quando non si vuole ricalcolare tutti i valori per quell'attributo nei vari oggetti
- Un attributo derivato è un attributo il cui valore viene calcolato in base ai valori di altri attributi
  - età
  - area, perimetro

# Attributi: come individuarli

---

- Direttamente nel testo della specifica
  - nomi che non sono buoni candidati per diventare classi
- Durante la definizione delle classi stesse
- Conoscenza del dominio applicativo

# Ereditarietà

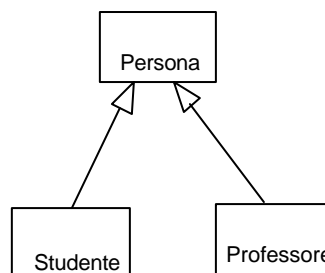
---

- Bisogna considerare tutte le classi
  - Si devono ricercare eventuali comportamenti comuni
- Aggiunta
  - Si aggiungono nuove proprietà alle classi
- Modifica
  - Si ridefiniscono/modificano operazioni esistenti

# Ereditarietà

---

- Esplicita eventuali comportamenti comuni
- È possibile dover:
  - Aggiungere nuove proprietà alle classi
  - Ridefinire/modificare e operazioni esistenti



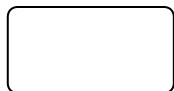
# Comportamento

---

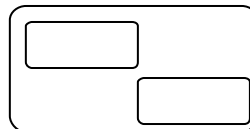
- Si definisce per mezzo di *statecharts diagram*
- Per ogni oggetto significativo
  - stati e transizioni
  - eventi e azioni
  - guardie
- Attenzione agli stati innestati

# StateCharts

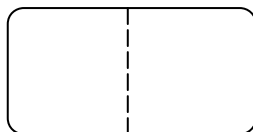
---



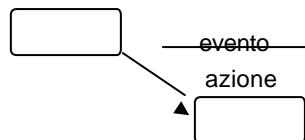
Stato



Decomposizione OR

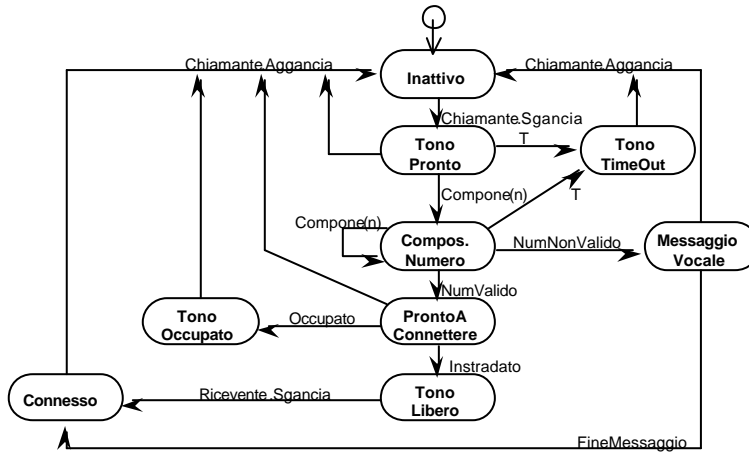


Decomposizione AND



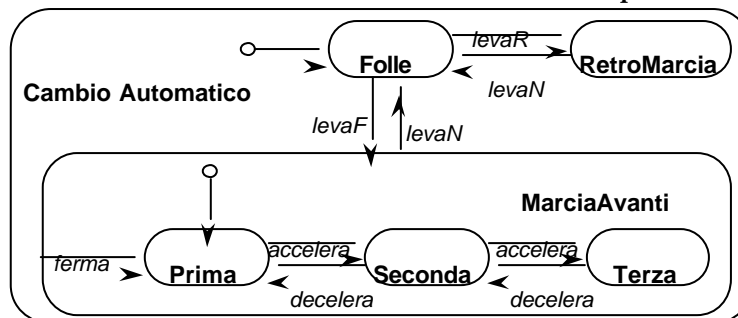
Transizione

# Esempio

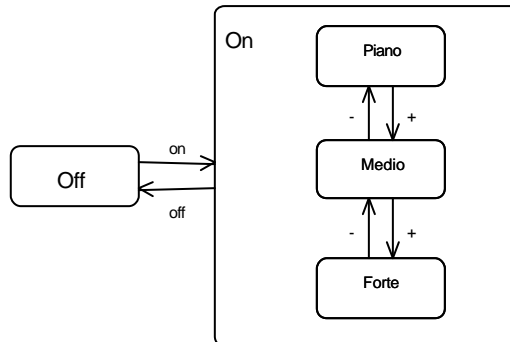


# Diagrammi a stati strutturati

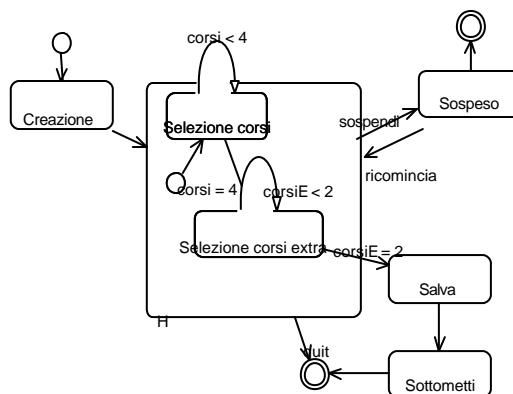
- Un macro stato equivale ad una scomposizione Or degli stati
- I sottostati ereditano le transizioni dei loro superstati



# Esempio: il ventilatore



# Esempio: selezione corso



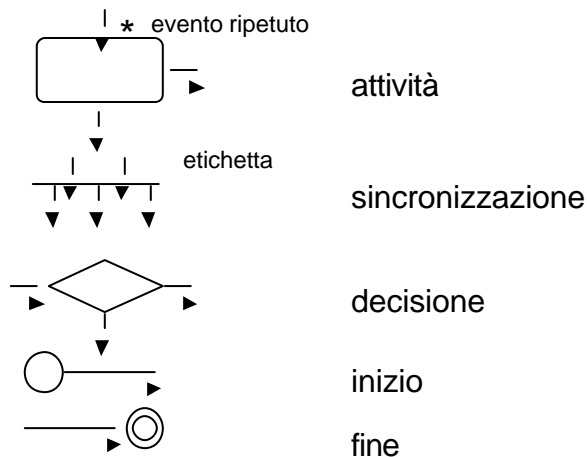
# Activity Diagram

---

- Forniscono la sequenza di operazioni che definiscono un'attività più complessa
- Un Activity Diagram può essere associato
  - A una classe
  - All'implementazione di un'operazione
  - Ad uno Use Case
- Retaggio della scomposizione “funzionale”

# Elementi principali

---



# Esempio

