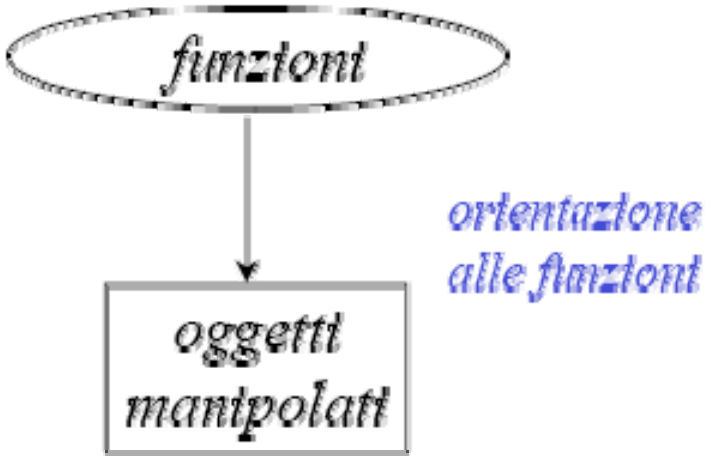
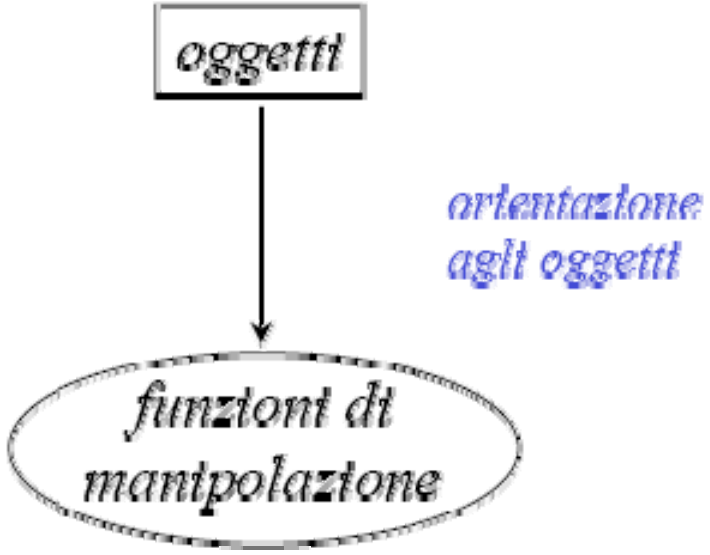
# Paradigma Object Oriented

# Funzioni od Oggetti ?

Molte delle tecniche tradizionali sono basate sull'idea di costruire un sistema concentrandosi sulle funzioni




Le tecniche Object-Oriented (OO) rovesciano questo rapporto: un sistema viene costruito partendo dalla classificazione degli oggetti da manipolare



# La programmazione orientata agli oggetti

- Nella **programmazione orientata agli oggetti** (**POO**) un programma è composto da un insieme di entità software, denominate **oggetti**, che interagiscono secondo determinati schemi. Esempi di linguaggi che supportano la programmazione orientata agli oggetti sono **Java** e **C++**.

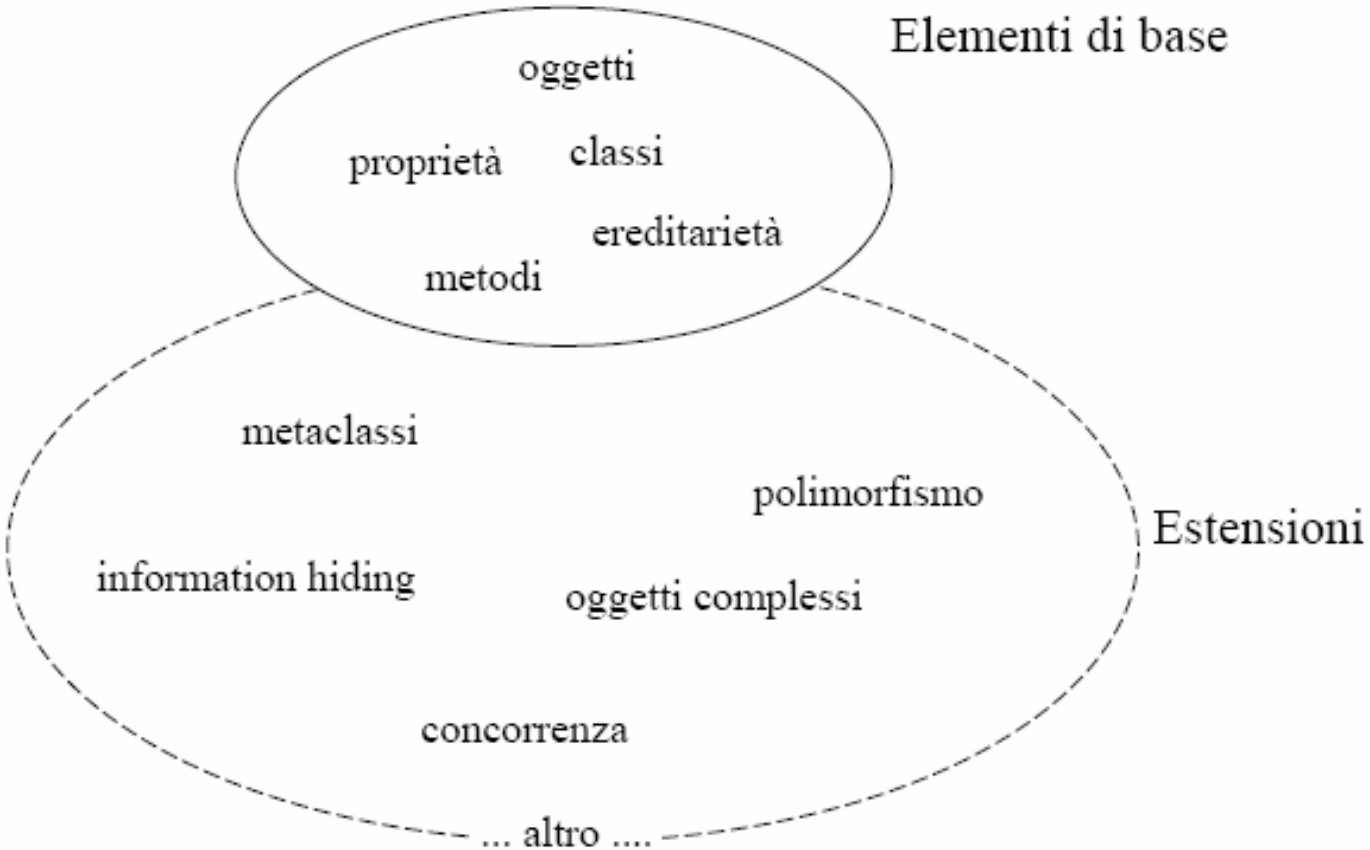
- La programmazione orientata agli oggetti differisce dalla cosiddetta **programmazione imperativa**, basata sul concetto di **istruzione**: un programma consiste in una sequenza di istruzioni che specificano in modo dettagliato le operazioni che devono essere eseguite dall'elaboratore per risolvere il problema dato. Esempi di linguaggi imperativi sono **Fortran**, **Pascal** e **C**.

- E' possibile utilizzare i linguaggi orientati agli oggetti per realizzare programmi prevalentemente imperativi (*come nel corso di Fondamenti di Informatica*), ma in tal modo non si sfruttano i vantaggi derivanti dall'uso dei meccanismi propri della programmazione orientata agli oggetti.

# Principi di base dell'approccio OO

- Modellare gli aspetti della realtà di interesse nel modo più diretto ed astratto possibile (strutturazione), mediante le nozioni di oggetto, classi e relazioni
- Costruire il programma in termini di moduli, sulla base del principio che ogni classe è un modulo (modularità)
- Legare gli aspetti comportamentali a quelli strutturali
- Proteggere le parti delicate del SW permettendo solo un accesso controllato a dati e funzioni (concetto di interfaccia)

# Fondamenti dell'approccio OO



# Oggetti

I linguaggi orientati agli oggetti (*object oriented*) sono basati sul concetto di ***oggetto***.

Un *oggetto* è una entità software che incapsula un insieme di ***attributi*** (o *dati*) e comprende un insieme di ***operazioni*** (o *metodi* o *funzioni*) che manipolano i dati.

L'insieme dei valori assunti dai suoi attributi costituisce lo ***stato*** dell'oggetto.

L'insieme delle sue operazioni definiscono il ***comportamento*** dell'oggetto.

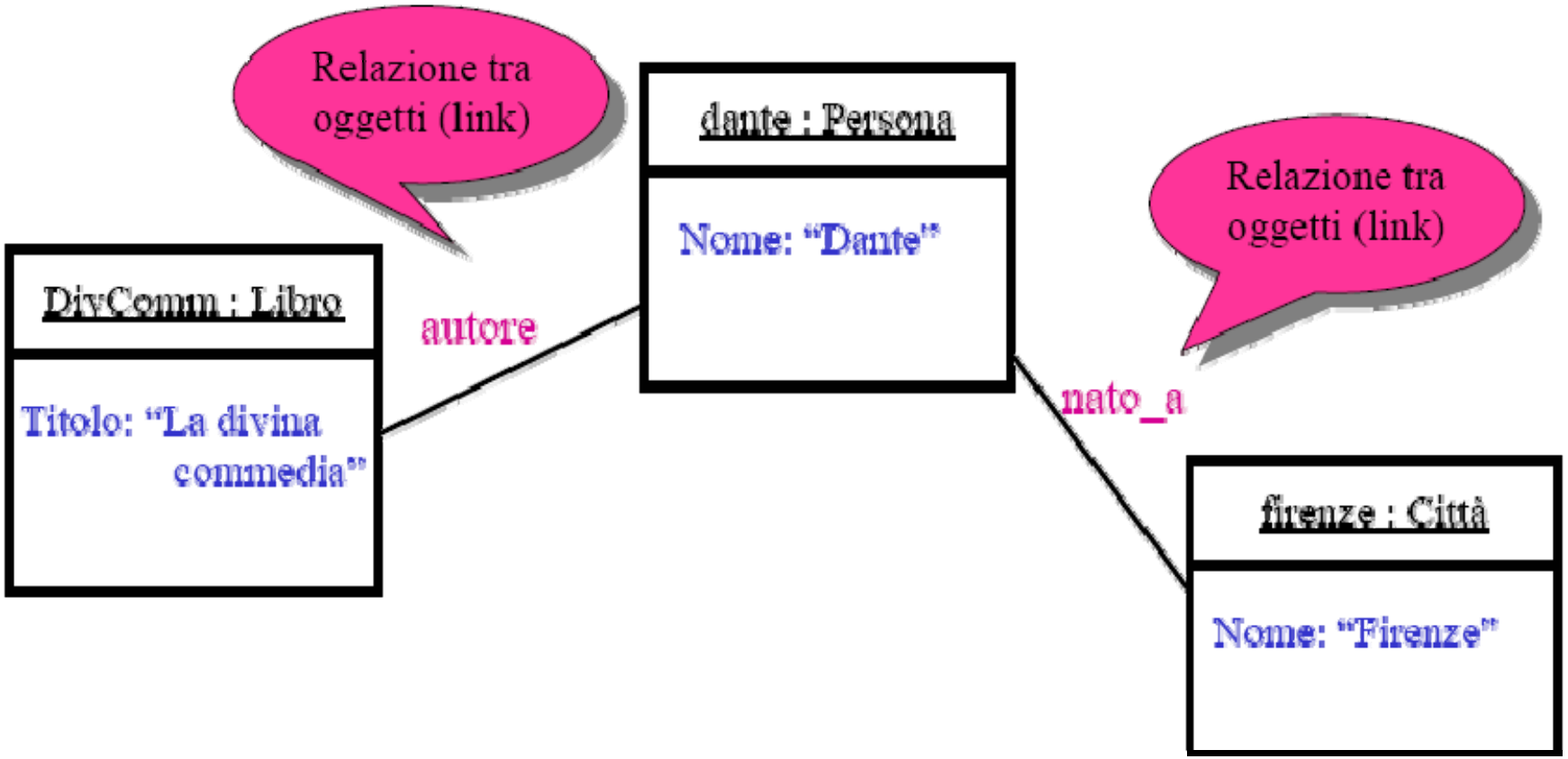
# Oggetti

- Elemento che ha vita propria e che è caratterizzato da un identificatore e da un insieme di proprietà (sia locali sia in relazione con altri oggetti)

- DivComm è l'identificatore
- Titolo è una proprietà locale ed il valore associato a tale proprietà è una stringa

# Oggetti

- Elemento che ha vita propria e che è caratterizzato da un identificatore e da un insieme di proprietà, sia locali (attributi) sia in relazione con altri oggetti



# Oggetti

Gli oggetti sono generalmente ***entità passive***: in un dato istante nell'esecuzione di un programma un solo oggetto è ***attivo*** (ovvero *in esecuzione*).

Un oggetto diventa attivo quando riceve l'invocazione di un metodo da parte di un altro oggetto. Mentre l'oggetto ricevente è attivo, il richiedente è passivo (non in esecuzione) in attesa dei risultati del metodo invocato.

Una volta che i risultati sono stati inviati al richiedente, l'oggetto ricevente torna in uno stato passivo ed il richiedente prosegue la sua esecuzione.

# Classi

Ogni oggetto è istanza di una determinata ***classe***.

Una *classe* definisce una tipologia di oggetti che possono essere descritti dallo stesso insieme di attributi e dallo stesso insieme di metodi.

Due istanze della stessa classe condividono le stesse ***proprietà strutturali*** (attributi) e le medesime ***proprietà comportamentali*** (metodi), ma sono comunque oggetti distinti.

# Classi

Il concetto di classe fornisce al programmatore uno strumento per creare *nuovi tipi* che possono essere usati facilmente come i *tipi predefiniti* (come *int*, *float*, *double*, etc.).

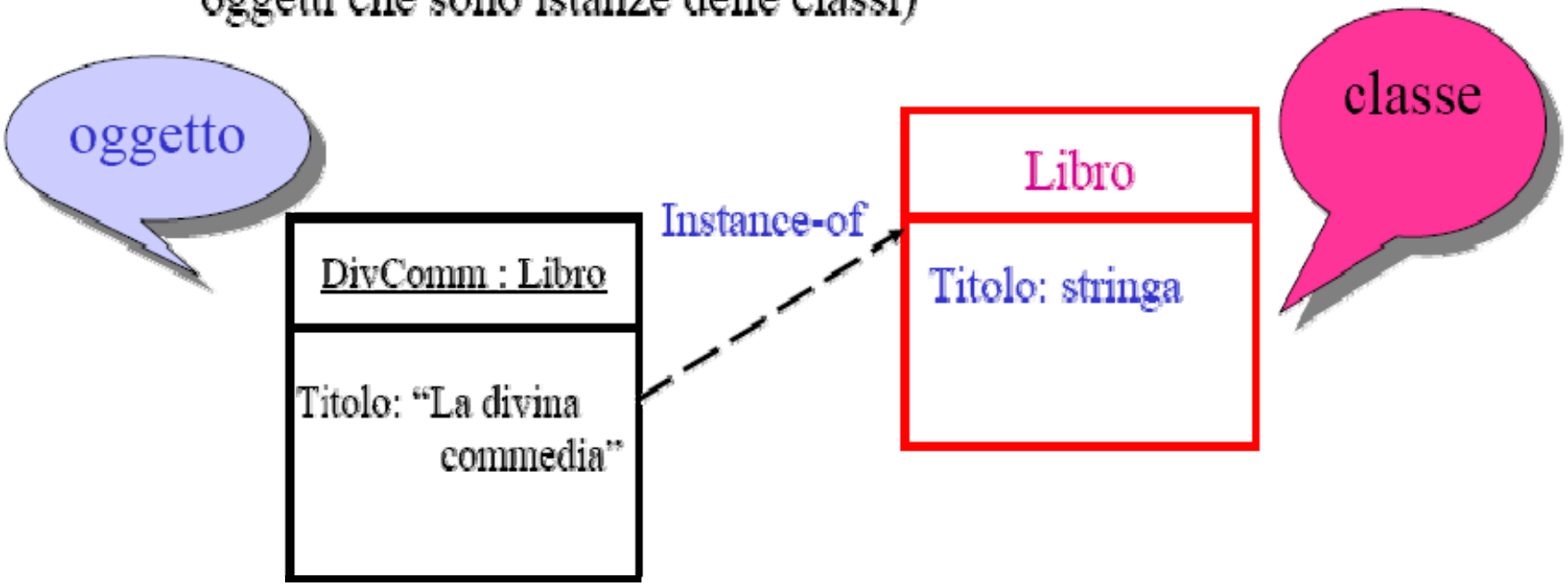
Un tipo è la rappresentazione concreta di un *concetto*. Ad es., il tipo predefinito *double*, insieme alle sue operazioni +, -, * e /, fornisce una rappresentazione del concetto di numero reale.

Una classe è quindi un nuovo tipo definito dal programmatore. Si progetta un nuovo tipo per fornire la definizione di un concetto che non è definito direttamente tra i tipi predefiniti.

Ad esempio, può essere utile definire le classi *Resistore*, *Condensatore*, o *Diodo* in un programma per l'analisi o la progettazione di circuiti elettronici.

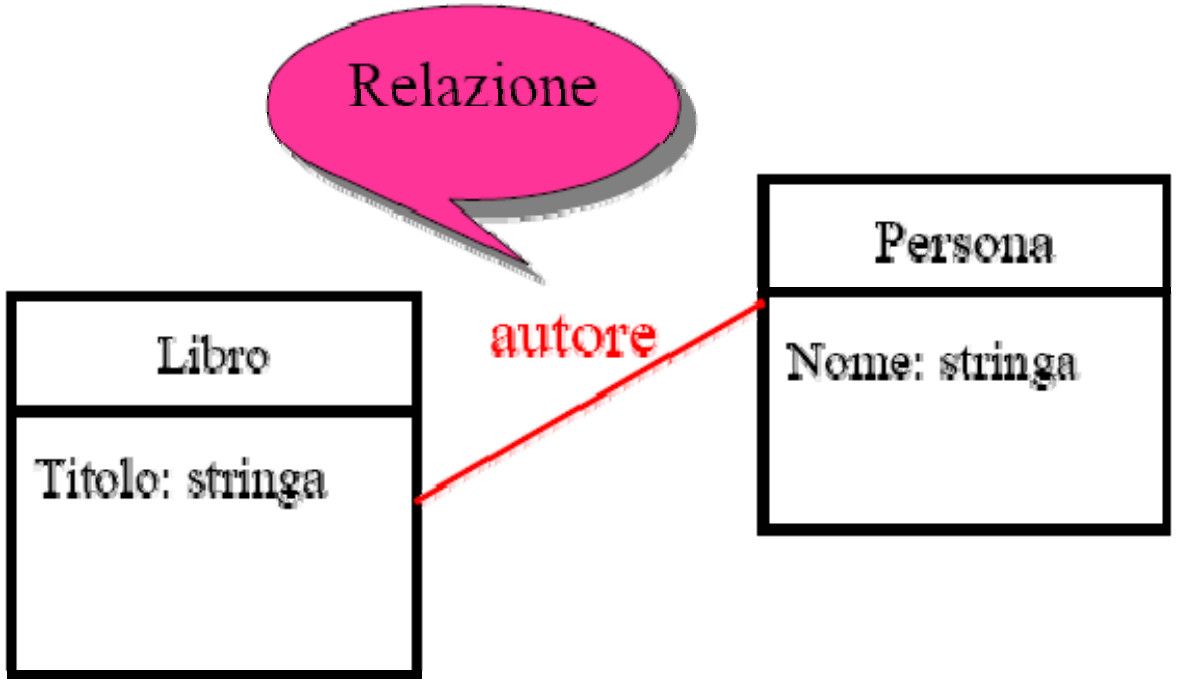
# Classi

- Rappresentano un naturale meccanismo umano di astrazione che spinge a concentrarsi sulle proprietà comuni di oggetti (istanze della classe) differenti. Una classe e' descritta da:
  - un nome
  - un insieme di proprietà (astrazioni delle proprietà comuni degli oggetti che sono istanze delle classi)



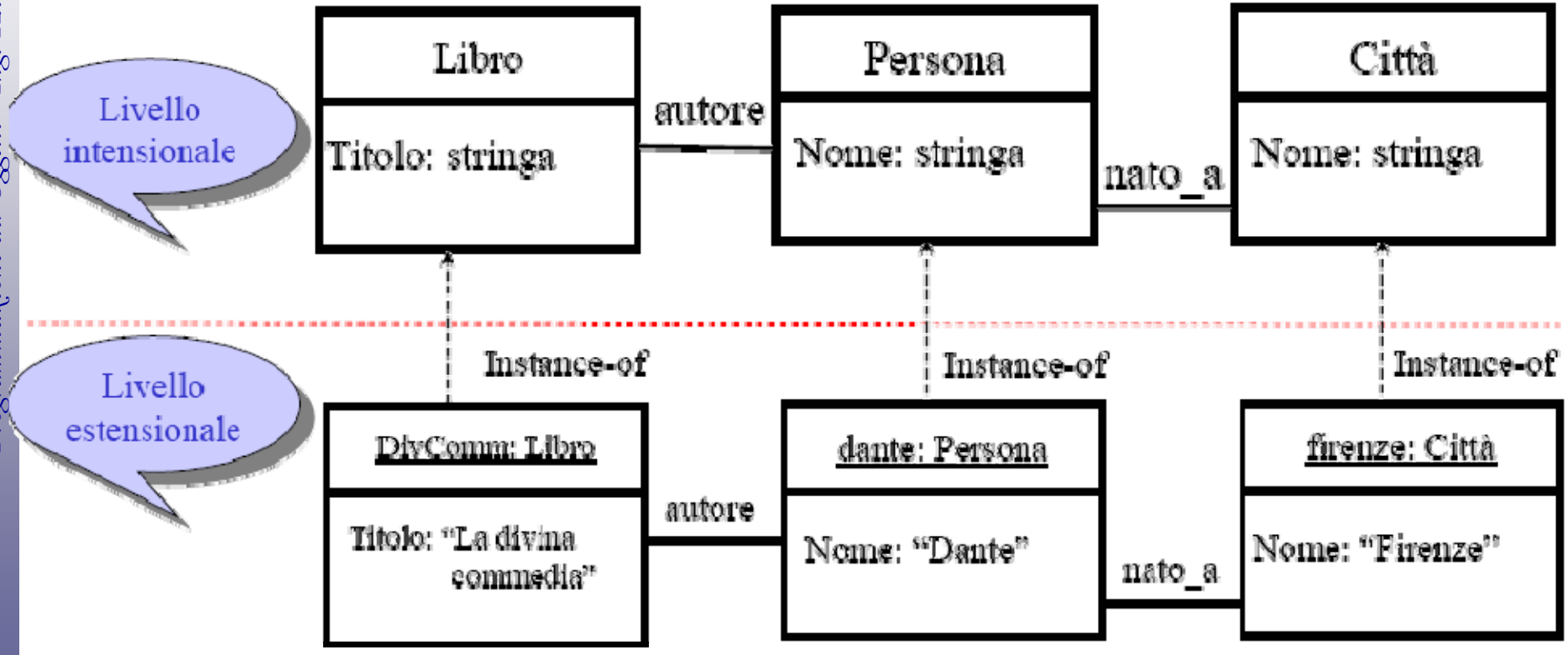
# Relazioni

- Rappresentano il meccanismo per definire legami tra classi (a cui corrispondono legami tra gli oggetti che sono istanze delle classi)



# Livello degli oggetti e livello delle classi

- Oggetti e classi appartengono a livelli diversi:
  - Livello estensionale (degli oggetti, o delle istanze)
  - Livello intensionale (livello delle classi)



# Proprietà fondamentali della POO

**Classificazione**. Ogni oggetto è istanza di una classe, che rappresenta un insieme di oggetti aventi le stesse proprietà strutturali e comportamentali.

**Incapsulamento**. Meccanismo che rende possibile il principio dell'*information hiding*, ovvero la capacità degli oggetti di nascondere al mondo esterno la propria organizzazione in termini di struttura e logica interna.

**Ereditarietà**. Meccanismo attraverso il quale una classe incorpora struttura e comportamento definiti da una classe più generale. La classe da cui si eredita è detta *superclasse*, mentre la classe discendente è detta *sottoclasse*.

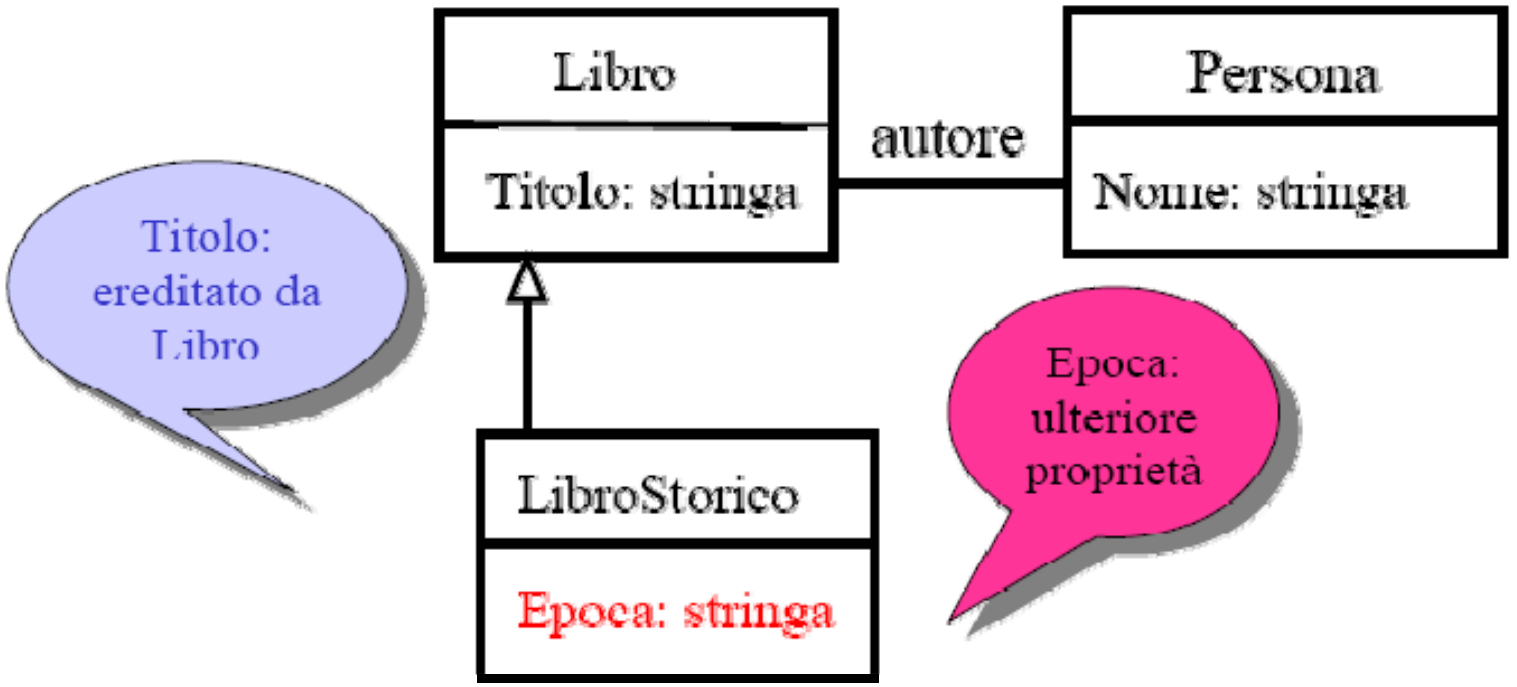
**Polimorfismo**. Capacità di supportare operazioni con la medesima firma e comportamenti diversi, situate in classi diverse ma derivanti da una stessa superclasse.

# Incapsulamento

- Con tale termine si intende l'assemblaggio di dati ed operazioni in una unica entità.
- La classe è il costrutto primario per tale proprietà: tramite il costrutto di classe è infatti possibile specificare la struttura di un insieme di oggetti unitamente al loro comportamento dinamico.

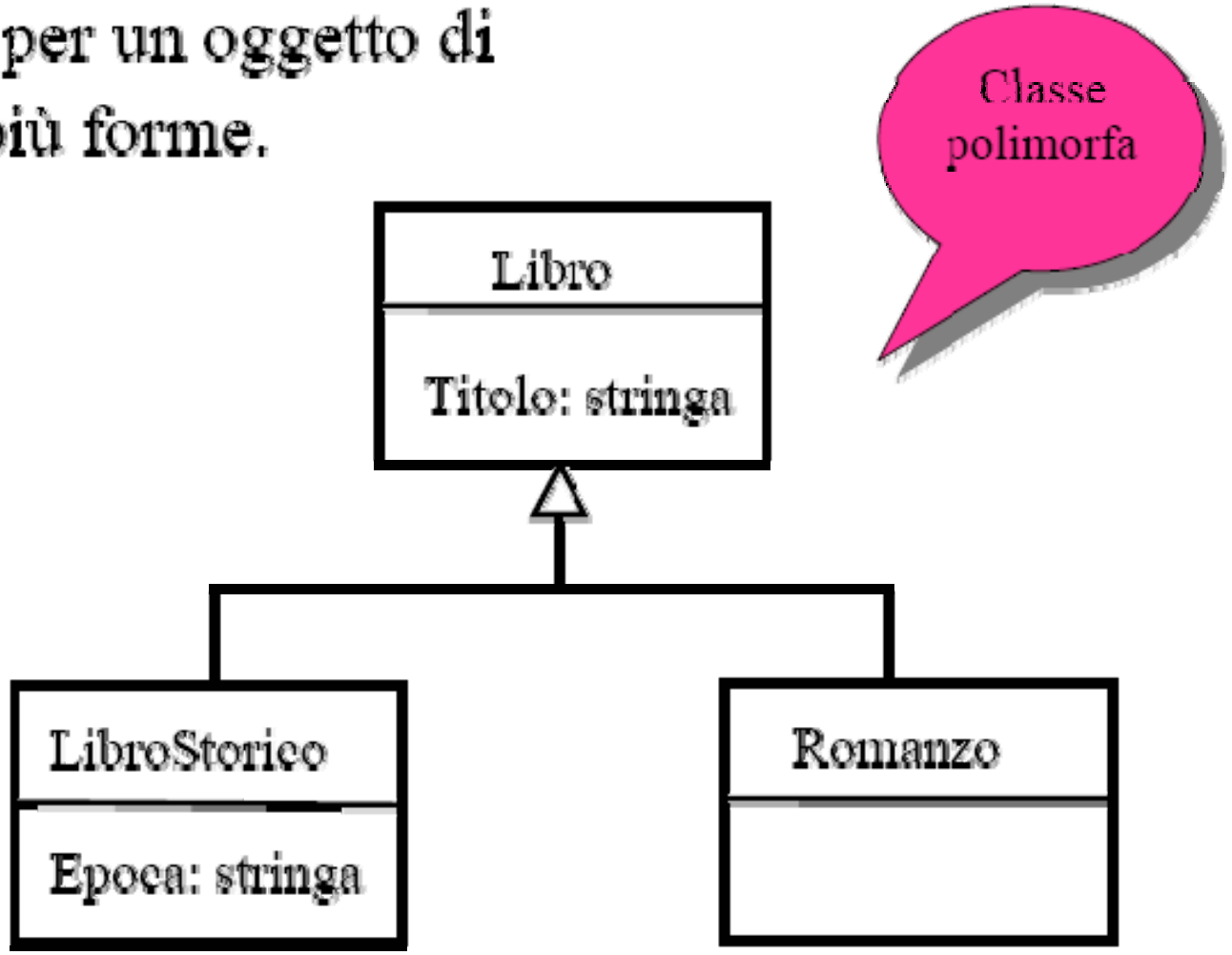
# Derivazione ed Ereditarietà

La classe sottoinsieme eredita tutte le proprietà della classe soprainsieme, e può avere ulteriori proprietà.



# Polimorfismo

Possibilità per un oggetto di assumere più forme.




Nota: deriva direttamente dalla ereditarietà

# Metodi

- Proprietà di una classe:
  - Statiche: attributi, relazioni
  - Dinamiche: metodi (un metodo può essere utilizzato per calcolare in modo dinamico una proprietà e anche per catturare degli aspetti comportamentali degli istanze della classe – ad esempio il metodo ISCRIZIONE per la classe STUDENTE).

```
public class Persona {
    public String nome;
    private Data dataNascita;

    public int Eta(Data oggi)  {
        return oggi – dataNascita;
    }
}
```