Richiami su oggetti e OOP

- Un **oggetto** (*object*) è una entità caratterizzata da una struttura dati alla quale si associa l'insieme delle operazioni che è possibile compiere su di essa.
- Un oggetto può essere *concreto* (es: un file) o *concettuale* (es: una politica di scheduling)
- Una attività è dunque orientata agli oggetti se opera su di essi, come ad esempio:
  - object-oriented analysis (OOA)
  - object-oriented design (OOD)
  - object-oriented programming (OOP)

Oggetti: caratteristiche

- Classificazione (Classification)
- Identità (Identity)
- Ereditarietà (Inheritance)
- Polimorfismo (Polymorphism)

Classificazione

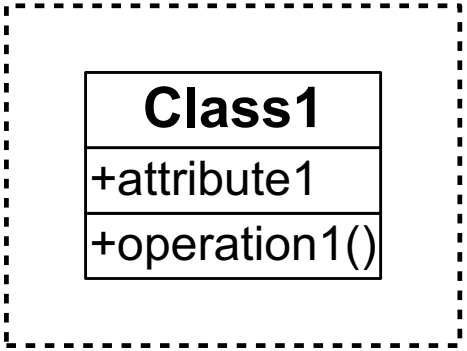
- Oggetti con identica struttura dati (attributes) ed operazioni (operations) sono raggruppati in **classi** (classes)
- Ogni oggetto è una istanza di una classe
- Ogni oggetto contiene un riferimento implicito alla classe di appartenenza
- Ogni oggetto condivide struttura dati ed operazioni con le altre istanze della sua classe di appartenenza
- L'istanziazione di un oggetto implica l'assegnamento di valori agli attributi

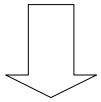
Identità

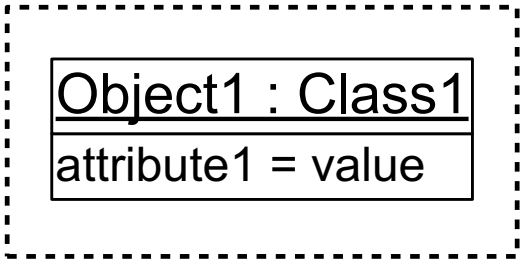
- Ogni oggetto ha una sua propria identità
- Due oggetti sono entità distinte anche se tutti i valori istanziati per gli attributi sono identici
- Gli oggetti sono identificabili attraverso un riferimento esplicito (handle)

Notazione grafica

Notazione per classe

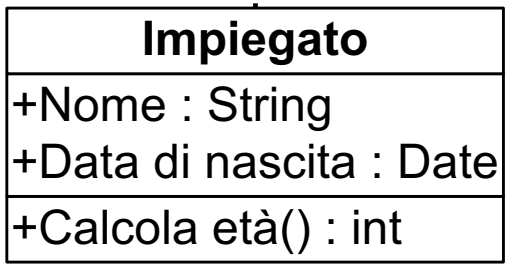


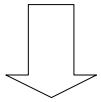


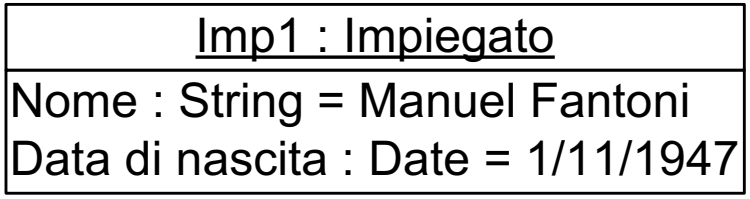


Notazione per oggetto (istanza)

Esempio di classe





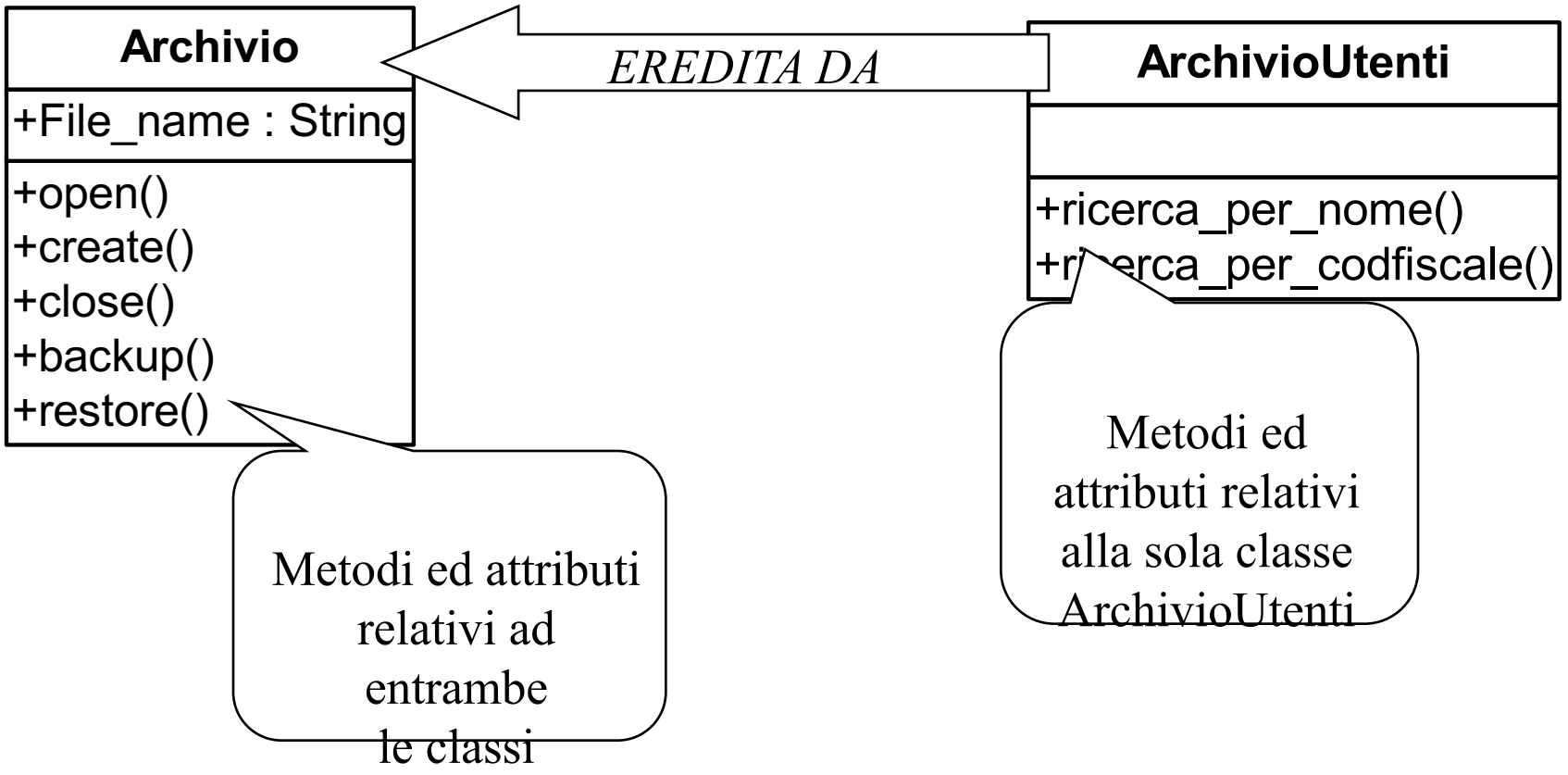


Esempio di oggetto

Ereditarietà

- L'ereditarietà è la condivisione di attributi ed operazioni tra classi in una qualche relazione gerarchica
- Una classe può essere raffinata attraverso un insieme di sottoclassi (subclasses), costituendo una superclasse (superclass) delle suddette sottoclassi
- Ogni sottoclasse eredita tutte le proprietà della relativa superclasse

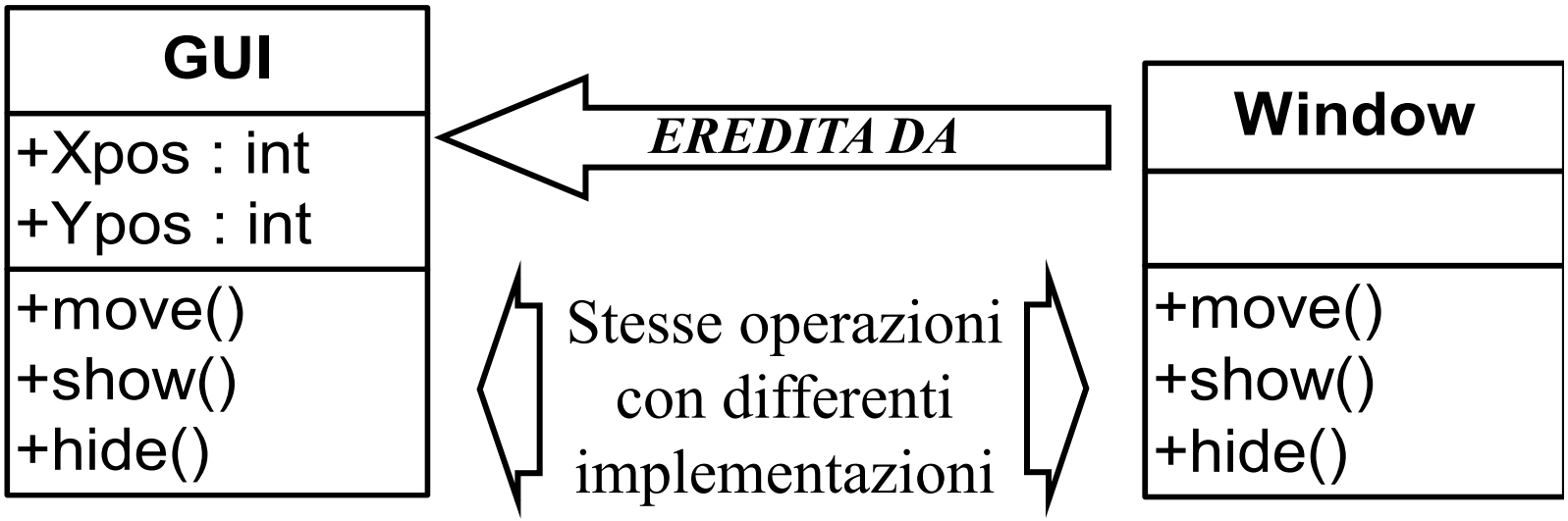
Ereditarietà: esempio



Polimorfismo

- Oggetti appartenenti a classi in una qualche relazione gerarchica possono condividere operazioni senza condividere forzatamente l'implementazione relativa ad ognuna di esse
- Una operazione rappresenta una particolare manipolazione dei dati istanziati per un oggetto
- Ci si riferisce all'*implementazione di una operazione* specifica per una particolare classe come ad un metodo (method) di tale classe
- Ad una operazione possono essere associati più metodi che la implementano

Polimorfismo: esempio



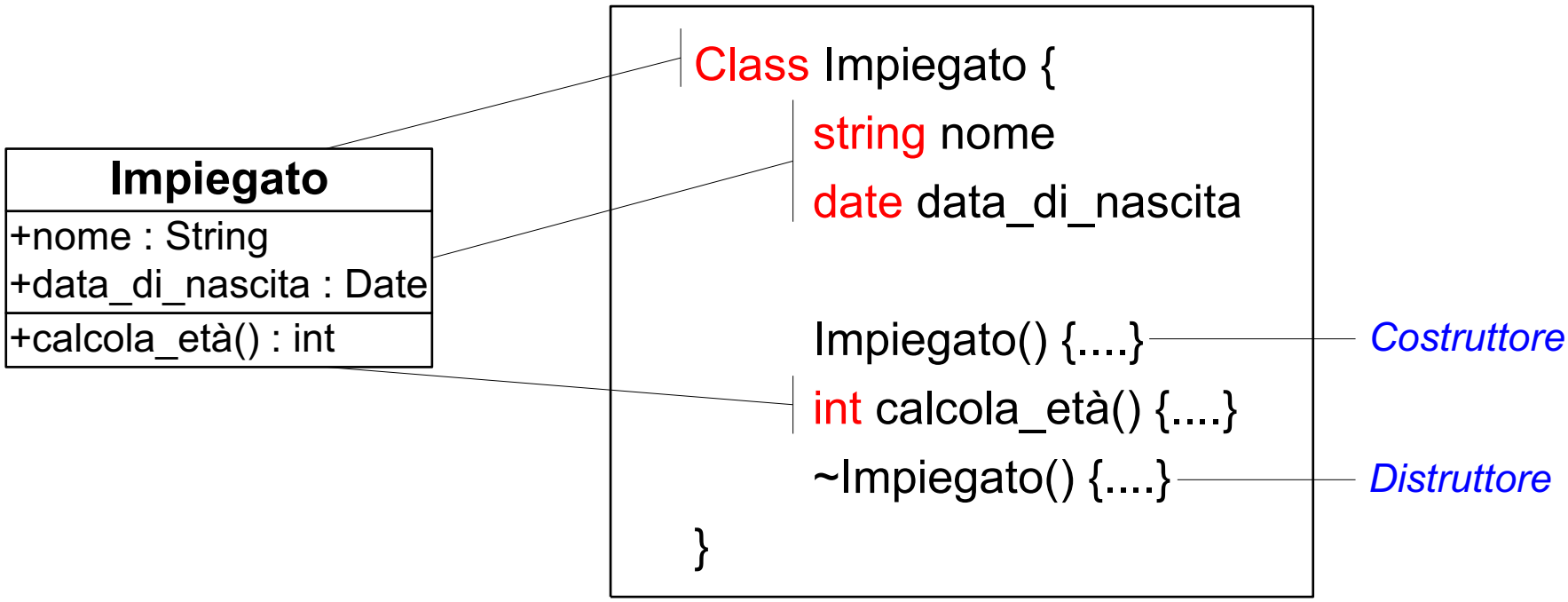
Vantaggi nell'utilizzo degli oggetti

Gli oggetti consentono di incrementare la qualità del software (in particolare *riusabilità* e *manutenibilità*), riducendo il *TTM* (*Time To Market*), grazie ai meccanismi di:

- Incapsulamento delle strutture dati (attributi) e delle operazioni (metodi) che le manipolano (encapsulation)
- Separazione tra la struttura dati propria di un oggetto e la struttura dati accessibile dall'esterno (information hiding)
- Astrazione del comportamento delle operazioni a prescindere dalla loro implementazione (abstraction)
- Condivisione di strutture dati ed operazioni, attraverso ereditarietà e polimorfismo (sharing)

Object Oriented Programming (OOP)

Dichiarazione di una Classe



OOP: istanziazione di un oggetto

- L'istanziazione di un oggetto è resa possibile dall'apposito operatore ***new***, attraverso il quale:
  - viene riservato *spazio in memoria* per l'oggetto
  - viene generato il *riferimento* per l'oggetto

ESEMPIO

```
Impiegato Nuovo_Impiegato;
Nuovo_Impiegato = new Impiegato("Manuel Fantoni","1/11/1947")
```

L'operatore new richiama il ***costruttore*** della classe attraverso il quale è possibile inizializzare i valori degli attributi

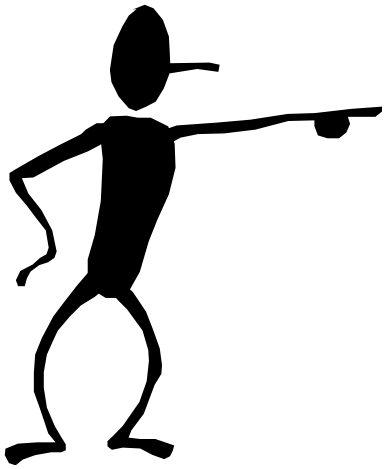
```
Impiegato(string Mio_Nome, date Mia_Data)
{
  this.nome = Mio_Nome;
  this.data_di_nascita = Mia_Data;
}
```

ESEMPIO

OOP: distruzione di un oggetto

- La distruzione di un oggetto consiste nel:
  - *liberare la memoria* allocata per l'oggetto
  - *eliminare il riferimento* dell'oggetto
- La distruzione di un oggetto è ottenibile attraverso:
  - operatore delete (es: C++)
  - garbage collector (es: Java)
- Prima che un oggetto venga realmente disallocato viene richiamato, qualora esista, il metodo distruttore, attraverso il quale è possibile specificare un insieme di operazioni da compiersi prima della distruzione

OOP: data encapsulation

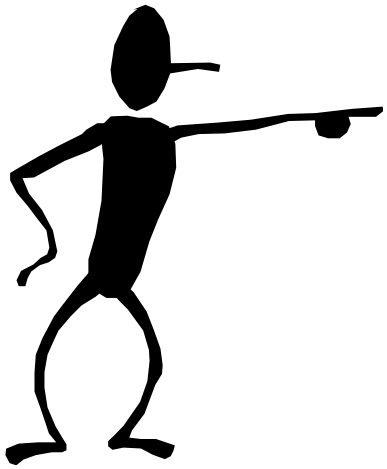


```
Class Posizione {
        Public Double Latitudine;
        Public Double Longitudine;

        ...

        Public Double calcola_distanza(Posizione p);
        Public Double calcola_direzione(Posizione p);
        ...
}
```

OOP: information hiding



```
Class Posizione {
        Private Double Latitudine;
        Private Double Longitudine;

        ...
        Public Void set_latitudine(Double lat);
        Public Void set_longitudine(Double long);
        Public Double get_latitudine();
        Public Double get_longitudine();
        Public Double calcola_distanza(Posizione p);
        Public Double calcola_direzione(Posizione p);
        ...
}
```

Associazioni tra classi

diagramma delle classi
(class diagram)

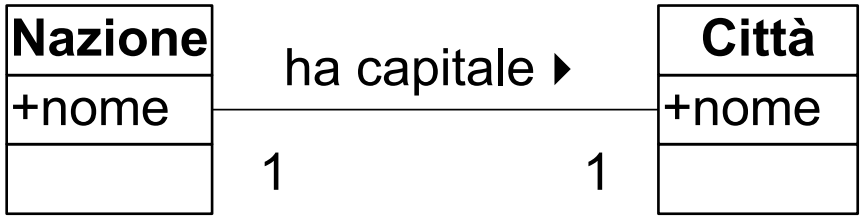


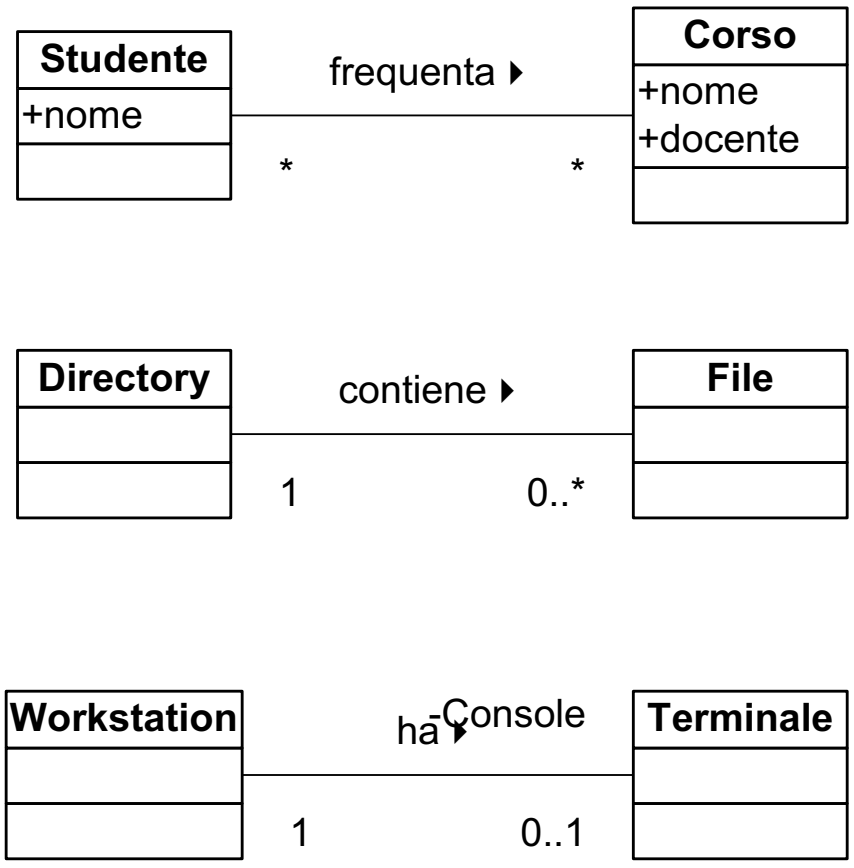
diagramma degli oggetti
(object diagram)

Esempio OOP

```
Class Nazione {
    String      Nome;
    Città       *Ha_Capitale;
}

Class Città {
    String      Nome;
    Nazione     *E'_Capitale_di
}
```

Ulteriori tipi di associazione

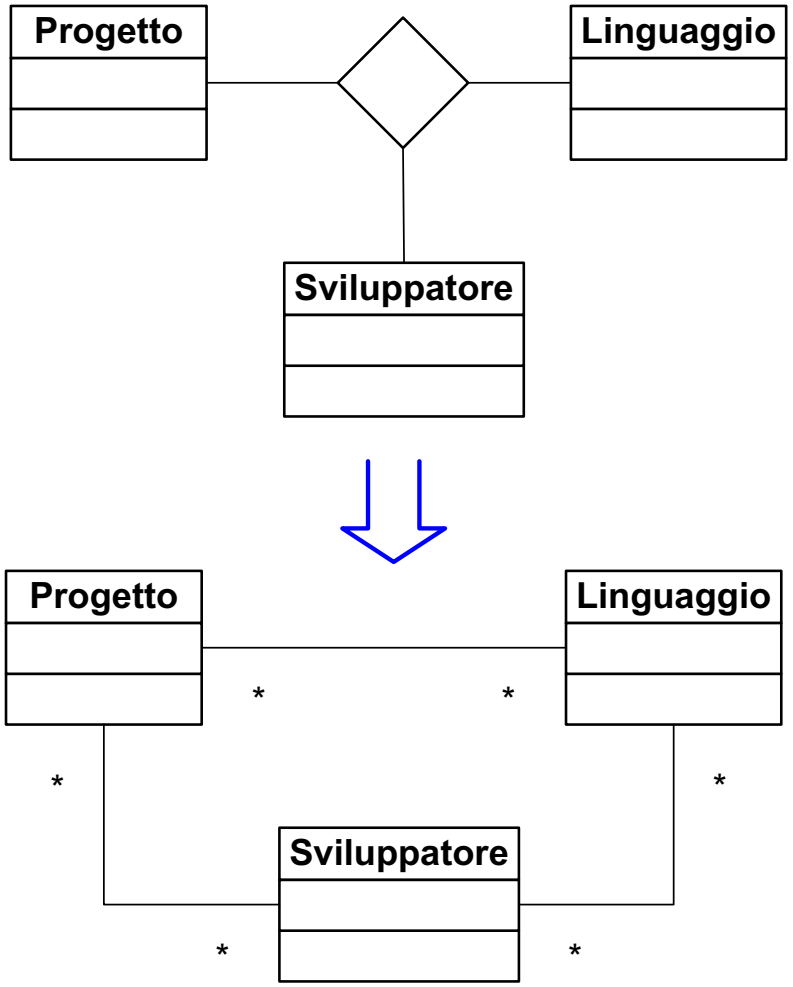


Esempio OOP

```
Class Studente {
    String      Nome;
    Corso       *Segue[n];
}

Class Corso {
    String      Nome;
    String      Docente;
    Studente    *Frequentano[m]
}
```

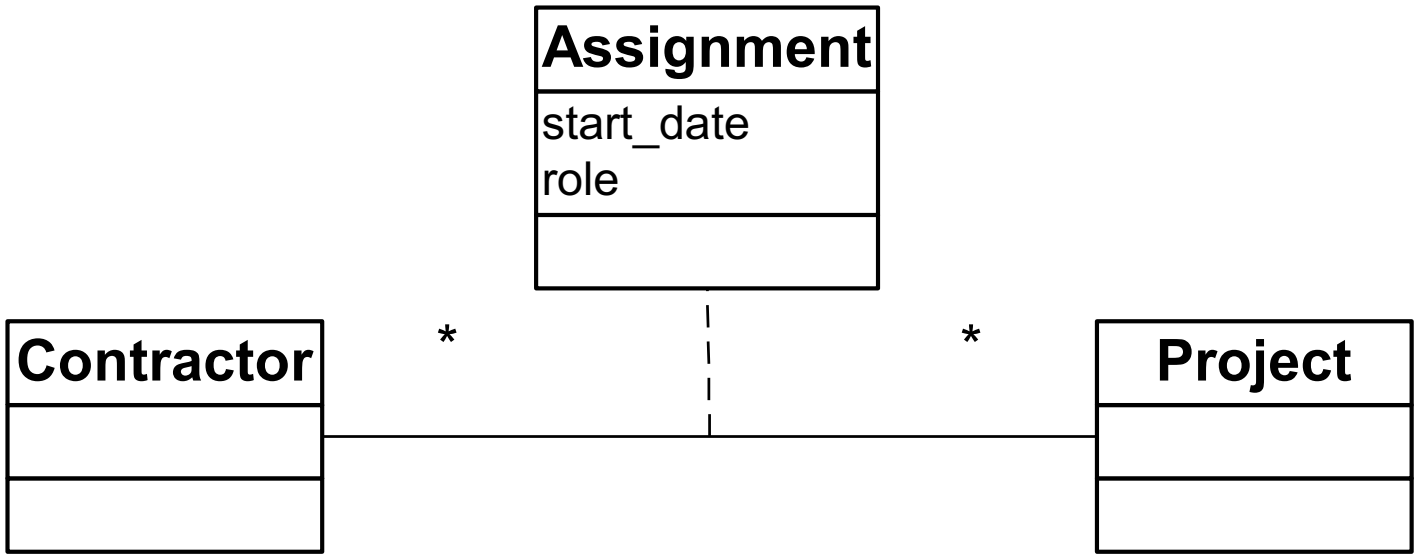
Associazione ternaria



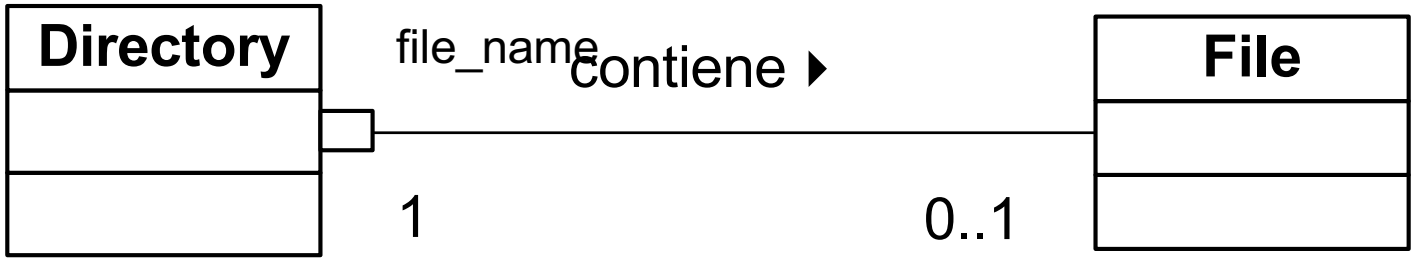
Esempio OOP

```
Class Progetto {
    Sviluppatore    **Assegnato;
    Linguaggio      **Sviluppato;
}
Class Sviluppatore {
    Linguaggio      **Sviluppa;
    Progetto        **Partecipa;
}
Class Linguaggio {
    Progetto        **Sviluppa;
    Sviluppatore    **Conosciuto;
}
```

Association classes



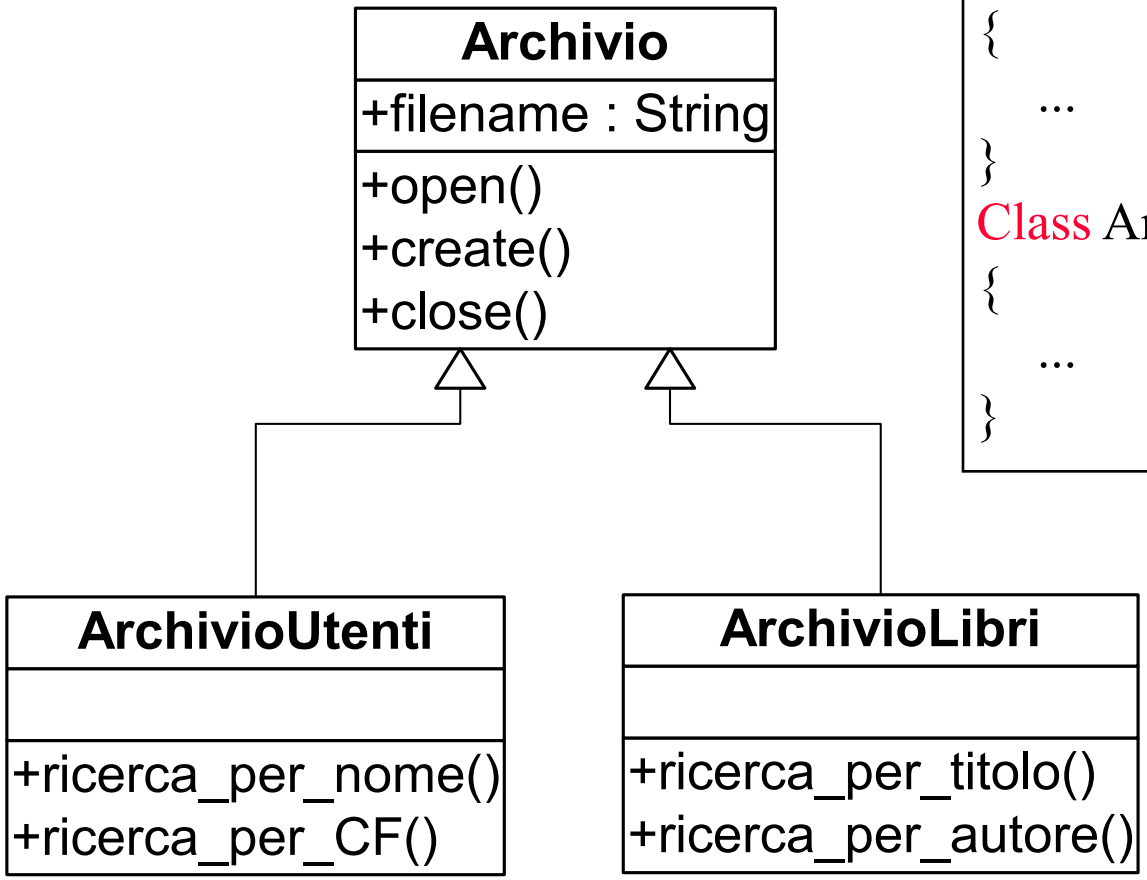
Associazioni qualificate



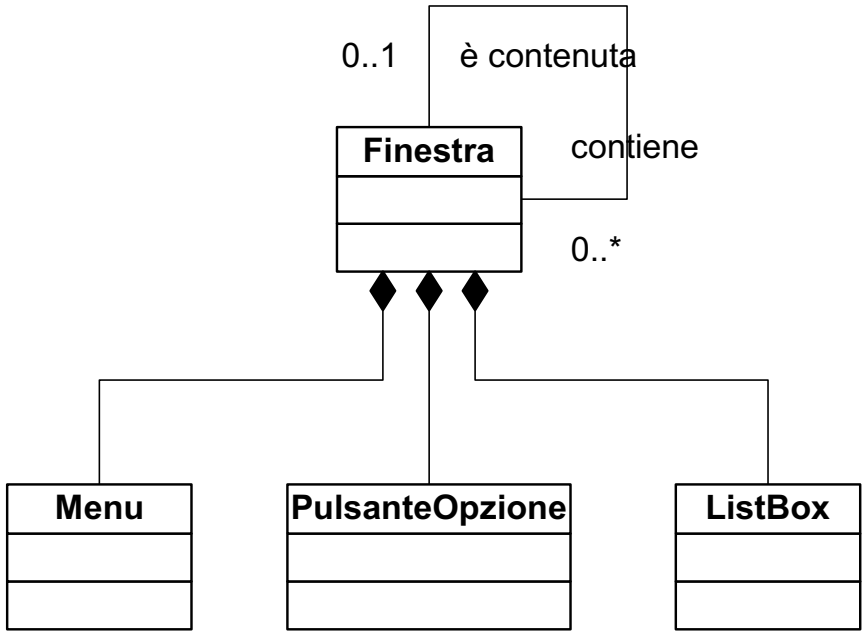
Ereditarietà

Esempio OOP

```
Class Archivio
{
    ...
}
Class ArchivioUtenti : public Archivio
{
    ...
}
```



Aggregazione



Esempio OOP

```
Class Menu {...}
Class PulsanteOpzione {...}
Class ListBox {...}

Class Finestra {
    Menu                Menu_in_Finestra[n];
    PulsanteOpzione     PO_in_Finestra[n];
    ListBox             LB_in_Finestra[n];
    Finestra            **Contiene;
    Finestra            *E'_Contenuta;
    ...
}
```