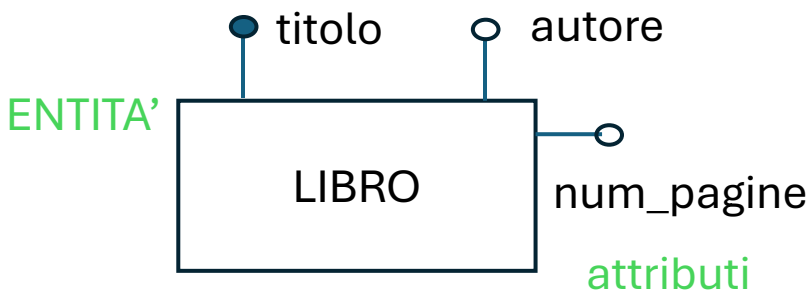
MONDO DELLE BASI DI DATI (E-R)

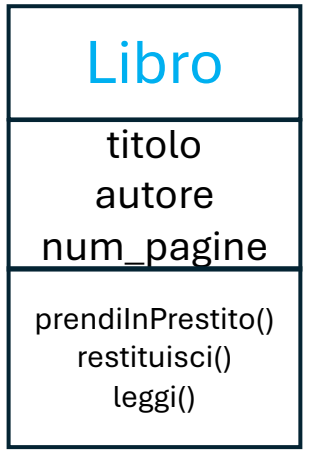


ASTRAZIONE

(STRUTTURA DEI DATI)

MONDO DELLA PROGRAMMAZIONE OOP (PYTHON)

CLASSE



attributi (dati)

```
class Libro:
    self.titolo =
    self.autore =
    self.num_pagine =
}
```

metodi (operazioni sui dati)

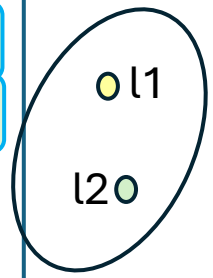
LA PROGRAMMAZIONE OR. AGLI OGGETTI MANTIENE INSIEME DATI E OPER. SUI DATI (ENCAPSULATION)

REIFICAZIONE

(DATI VERI E PROPRI, ISTANZE)

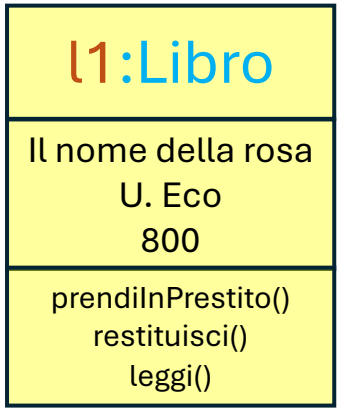
titolo	autore	num_pagine
Il nome della Rosa	U. Eco	800
Lo hobbit	K. Follet	3000

TUPLA, ISTANZA




```
l1 = Libro()
```

CREA ON OGGETTO DI CLASSE Libro DI NOME l1

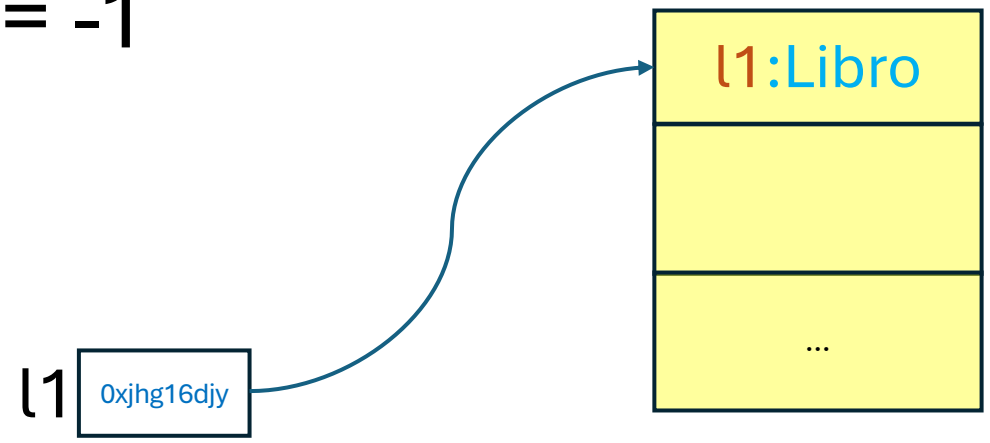


OGGETTO, ISTANZA

DATI E OPERAZIONI SUI DATI, INSIEME ENCAPSULATION

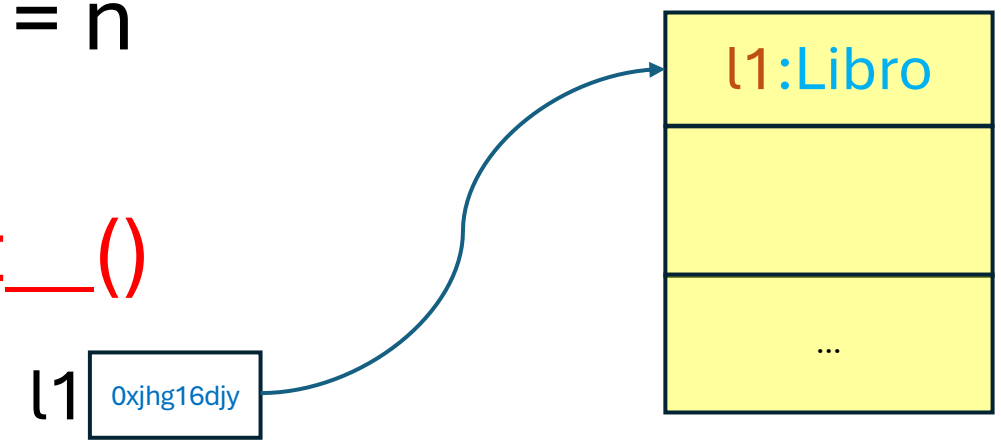
```
class Libro():
        # questa è la classe libro
      def __init__(self):
            # istruzioni per inizializz. gli attr. del libro
            self.titolo = “”
            self.autore = “”
            self.num_pagine = -1
```

```
l1 = Libro()
print(l1)
>>> 0xjhg16djy
```



VAR. RIFERIMENTO ALL’OGGETTO l1

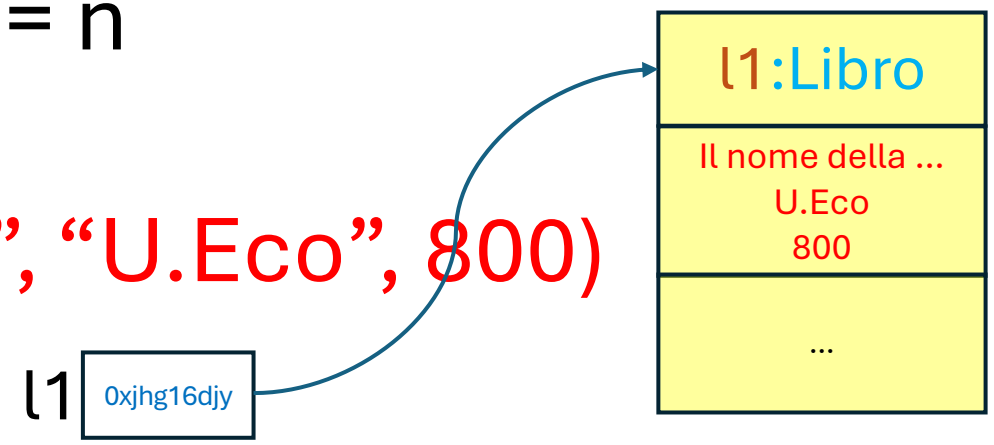
```
class Libro():
        # questa è la classe libro
      def __init__(self, t, a, n ):
            # istruzioni per inizializz. gli attr. del libro
            self.titolo = t
            self.autore = a
            self.num_pagine = n

l1 = Libro() // chiama __init__()
print(l1)
>>> 0xjhg16djy
```



VAR. RIFERIMENTO ALL'OGGETTO l1

```
class Libro():
        # questa è la classe libro
      def __init__(self, t, a, n ):
            # istruzioni per inizializz. gli attr. del libro
            self.titolo = t
            self.autore = a
            self.num_pagine = n
```

```
l1 = Libro("Il nome della ...", "U.Eco", 800)
print(l1)
>>> 0xjhg16djy
```

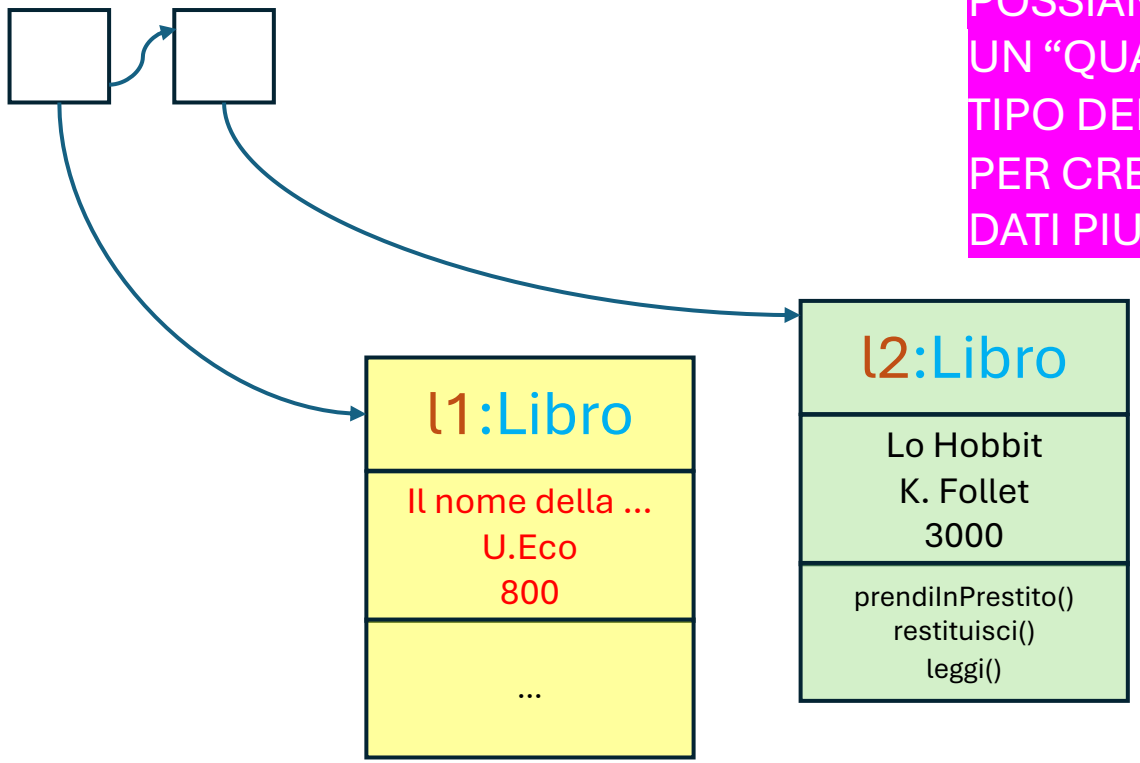


VAR. RIFERIMENTO ALL'OGGETTO l1

interi

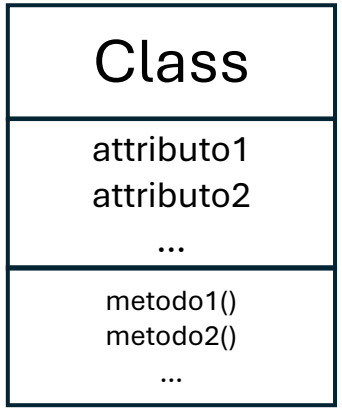
libri = [l1, l2]

DOPO CHE ABBIAMO DEFINITO UN TIPO DI DATO, ES. Libro POSSIAMO USARLO COME UN "QUALUNQUE" ALTRO TIPO DEL LINGUAGGO, ES. PER CREARE DELLE STRUTTURE DATI PIU' COMPLESSE (ES. LISTE)



libri[0]

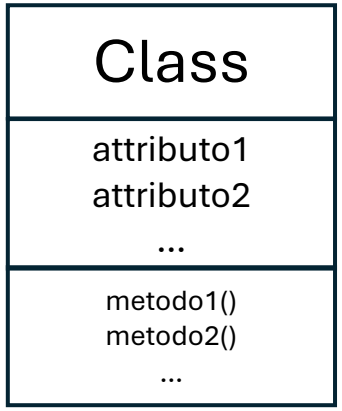
INFORMATION HIDING



SCEGLIERE
CIO' CHE
E' VISIBILE
DALL'ESTERNO
DELLA CLASSE
(CONTROLLARE
LA VISIBILITA')

MECCANISMI DELLA OOP PER RENDERE IL CODICE PIU' ROBUSTO

INFORMATION HIDING



SCEGLIERE CIO' CHE E' VISIBILE DALL'ESTERNO DELLA CLASSE (CONTROLLARE LA VISIBILITA', ES. PUBBLICA/PRIVATA)

MECCANISMI DELLA OOP PER RENDERE IL CODICE PIU' ROBUSTO ...

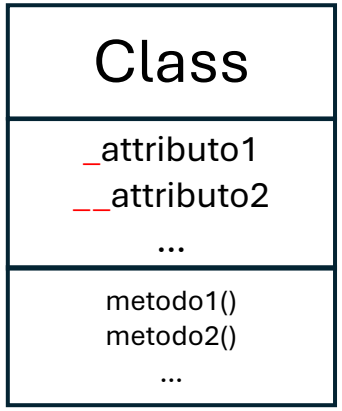
... MA PER PYTHON, DI DEFAULT ATTRIBUTI E METODI PUBBLICI

INFORMATION HIDING

SCEGLIERE CIO' CHE E' VISIBILE DALL'ESTERNO DELLA CLASSE (CONTROLLARE LA VISIBILITA', ES. PUBBLICA/PRIVATA)

TUTTAVIA IN PYTHON POSSO MODIFICARE LA VISIBILITA' DEGLI ATTRIBUTI CON _ E __

INFORMATION HIDING



SCEGLIERE CIO' CHE E' VISIBILE DALL'ESTERNO DELLA CLASSE (CONTROLLARE LA VISIBILITA', ES. PUBBLICA/PRIVATA)

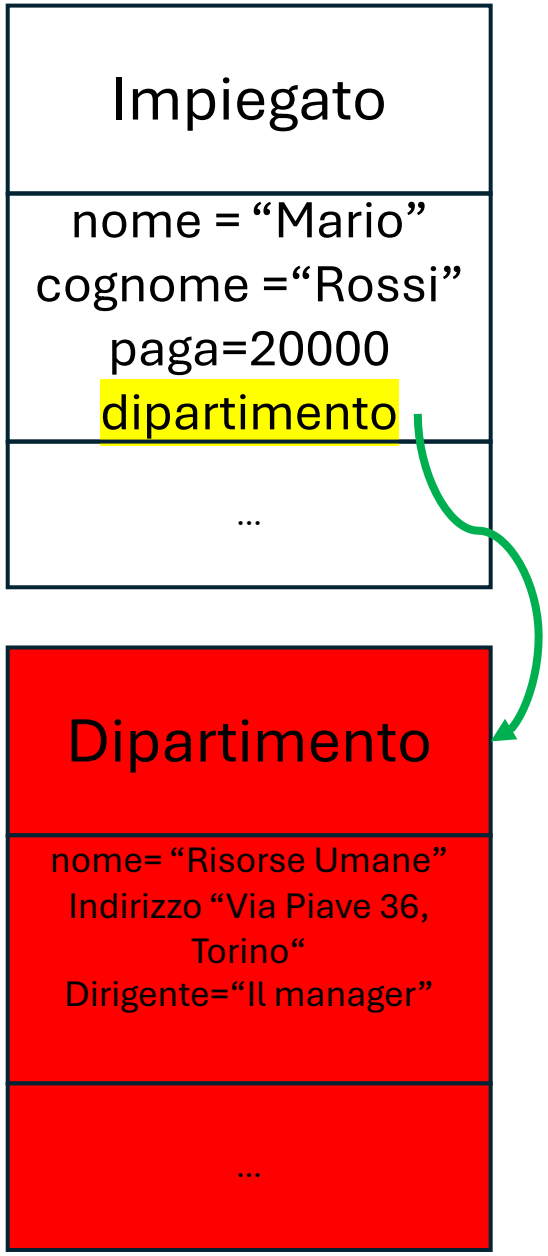
TUTTAVIA IN PYTHON POSSO MODIFICARE LA VISIBILITA' DEGLI ATTRIBUTI CON _ E __

... E RENDERLI ACCESSIBILI SOLO ATTRAVERSO METODI (NEI QUALI POSSO IMPLEMENTARE I CONTROLLI DEL CASO)

i1 = Impiegato (“Mario”, “Rossi”, 20000);

d1.dipartimento=dipartimentoA; # OGGETTO

dipartimentoA = Dipartimento (“Risorse umane”);

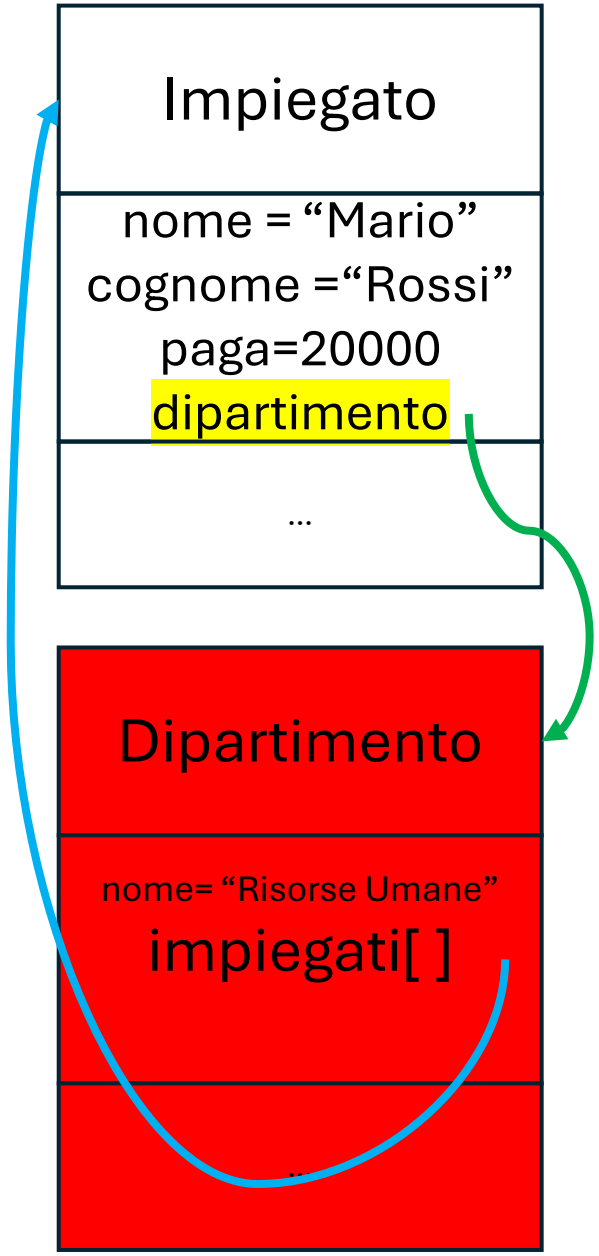


Come faccio a dire/a rappresentare il fatto che un impiegato lavori in un determinato dipartimento?

i1 = Impiegato (“Mario”, “Rossi”, 20000);

d1.dipartimento=dipartimentoA; # OGGETTO

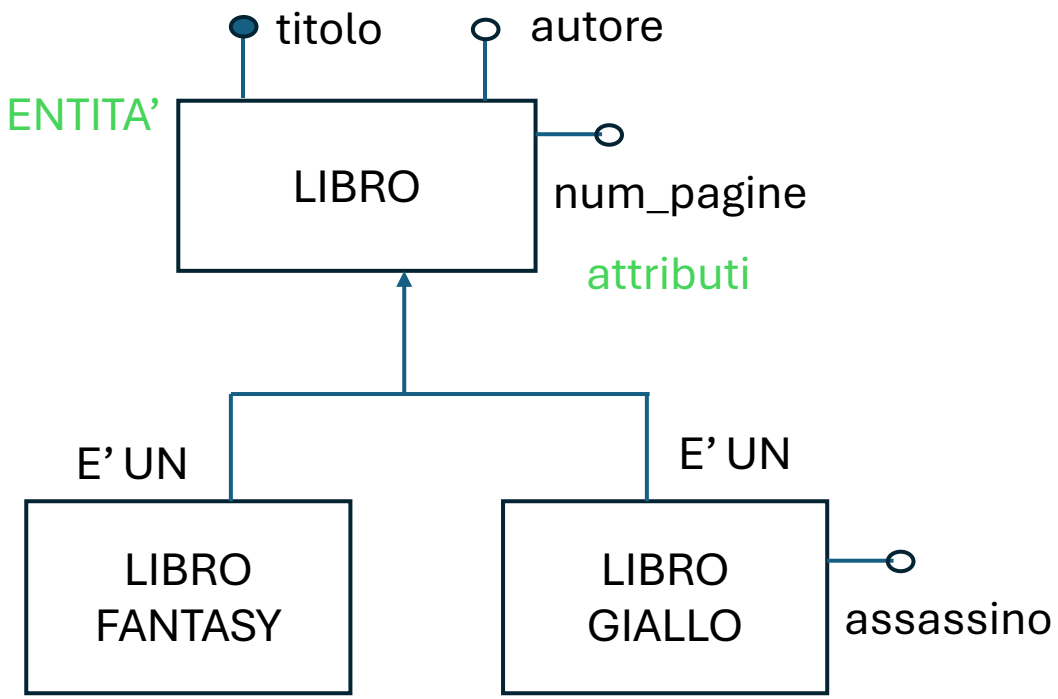
dipartimentoA = Dipartimento (“Risorse umane”);



Come faccio a dire/a anche che un dipartimento ospita dei dipendenti?

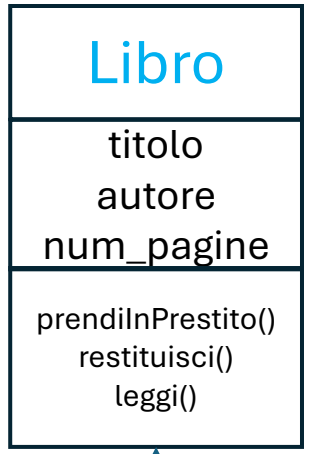
EREDITARIETA’

MONDO DELLE BASI DI DATI (E-R)



SBAGLIATO ANDARE A RIDISEGNARE, PER LE ENTITA’ FIGLIE, ATTRIBUTI E RELAZIONI DEL PADRE

MONDO DELLA PROGRAMMAZIONE OOP (PYTHON)



CLASSE PADRE, DI BASE, o SUPER-CLASSE

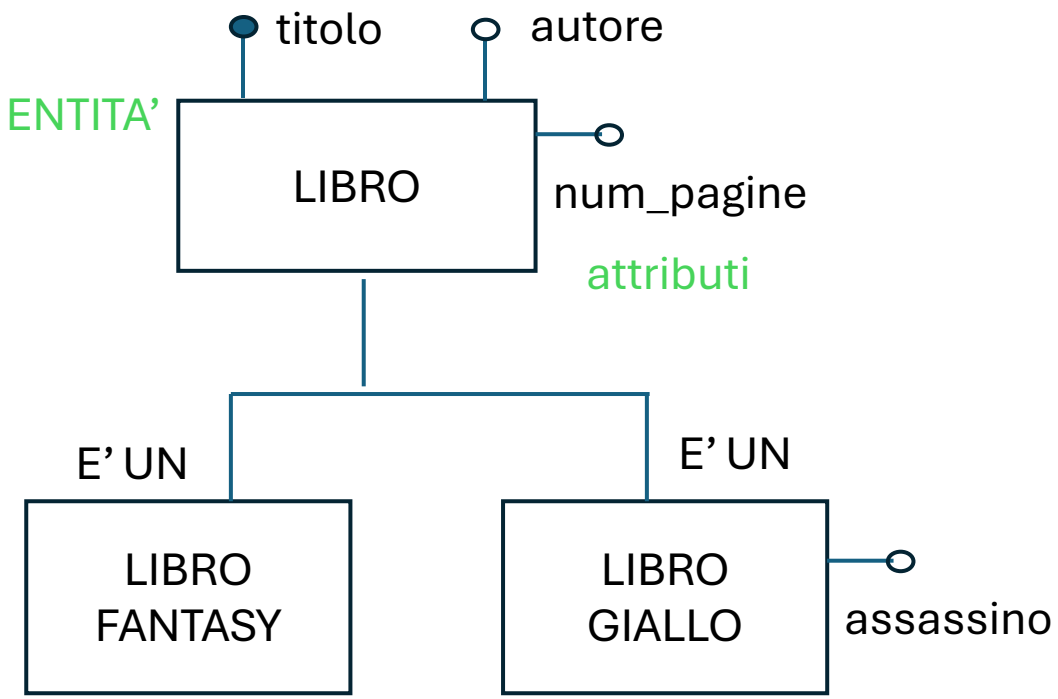
IS A

CLASSE FIGLIA, DERIVATA, o SOTTO-CLASSE

LA CLASSE FIGLIA (DERIVATA) EREDITA **TUTTI** GLI ATTRIBUTI ED I METODI DELLA CLASSE PADRE

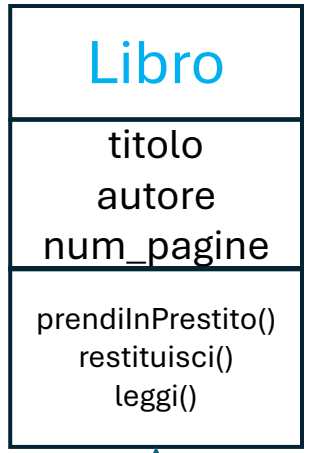
EREDITARIETA’

MONDO DELLE BASI DI DATI (E-R)



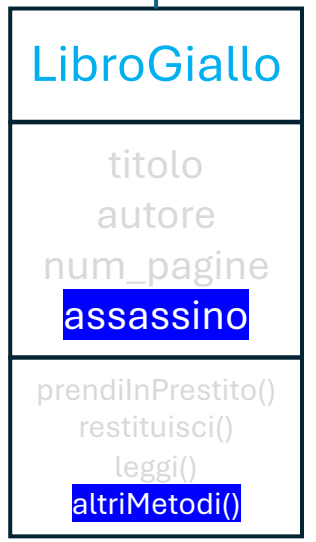
SBAGLIATO
ANDARE
A RIDISEGNARE
PER LE ENTITA’ FIGLIE
ATTRIBUTI E RELAZIONI
DEL PADRE

MONDO DELLA PROGRAMMAZIONE OOP (PYTHON)



CLASSE PADRE,
DI BASE, o
SUPER-CLASSE

IS A



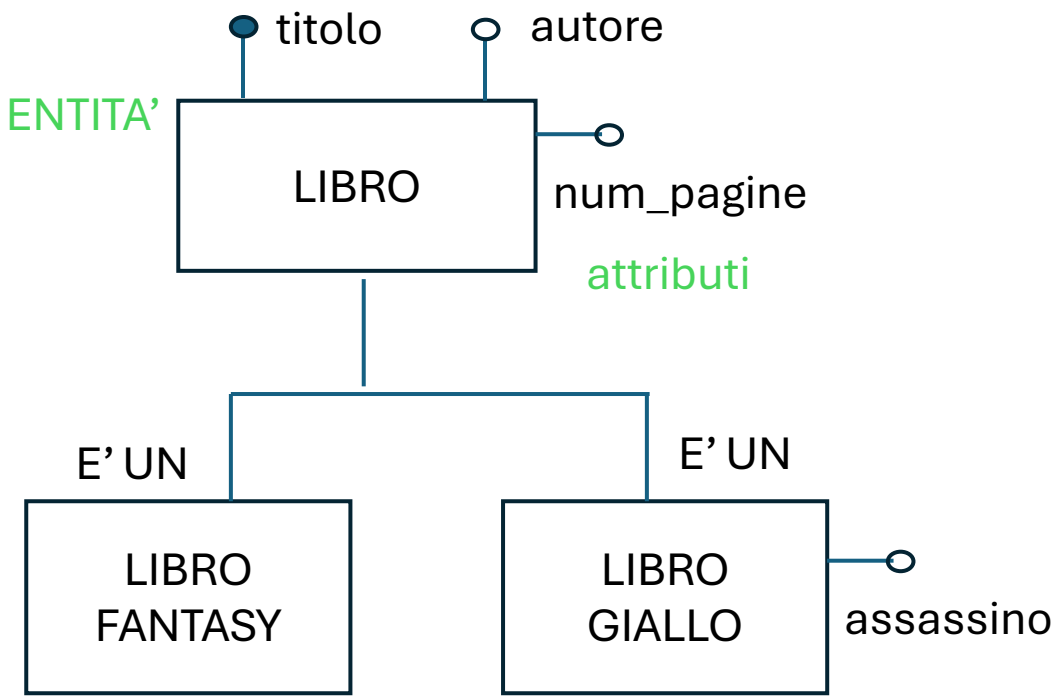
CLASSE
FIGLIA,
DERIVATA,
o SOTTO-
CLASSE

LA CLASSE FIGLIA (DERIVATA)
EREDITA **TUTTI**
GLI ATTRIBUTI ED I METODI
DELLA CLASSE PADRE

E NE PUO’ AGGIUNGERE
DI SPECIFICI/SPECIALIZZATI

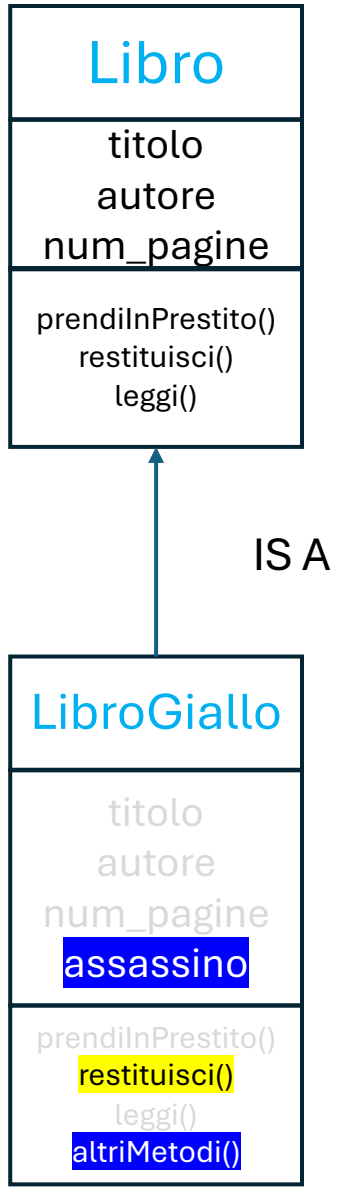
EREDITARIETA’

MONDO DELLE BASI DI DATI (E-R)



SBAGLIATO
ANDARE
A RIDISEGNARE
PER LE ENTITA’ FIGLIE
ATTRIBUTI E RELAZIONI
DEL PADRE

MONDO DELLA PROGRAMMAZIONE OOP (PYTHON)



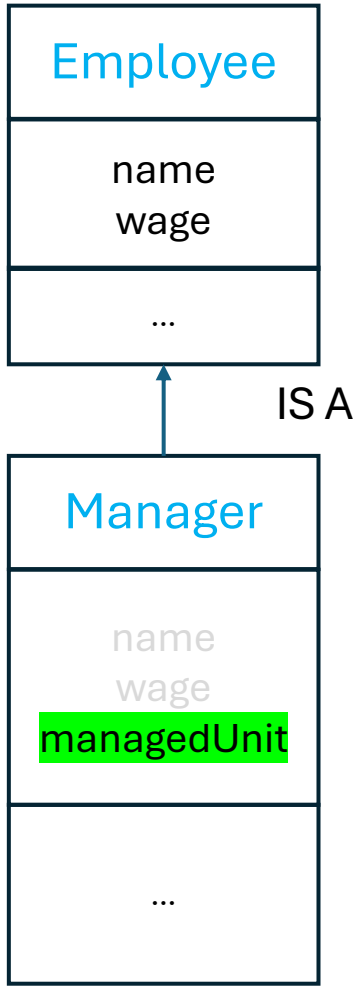
CLASSE PADRE,
DI BASE, o
SUPER-CLASSE

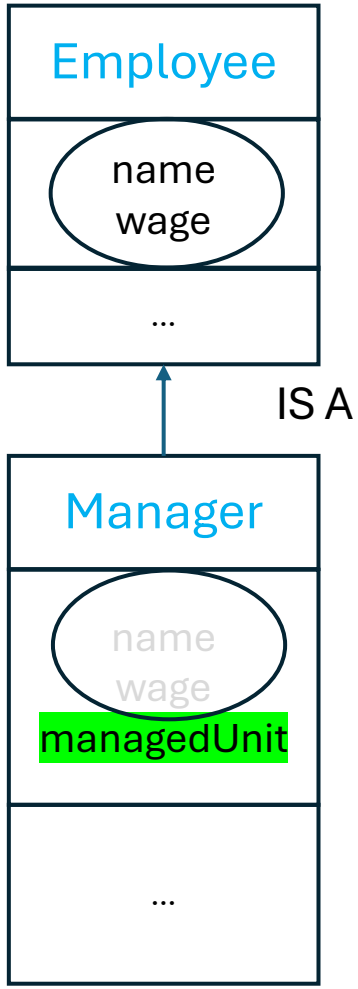
CLASSE
FIGLIA,
DERIVATA,
o SOTTO-
CLASSE

LA CLASSE FIGLIA (DERIVATA)
EREDITA **TUTTI**
GLI ATTRIBUTI ED I METODI
DELLA CLASSE PADRE

E NE PUO’ AGGIUNGERE
DI SPECIFICI/SPECIALIZZATI

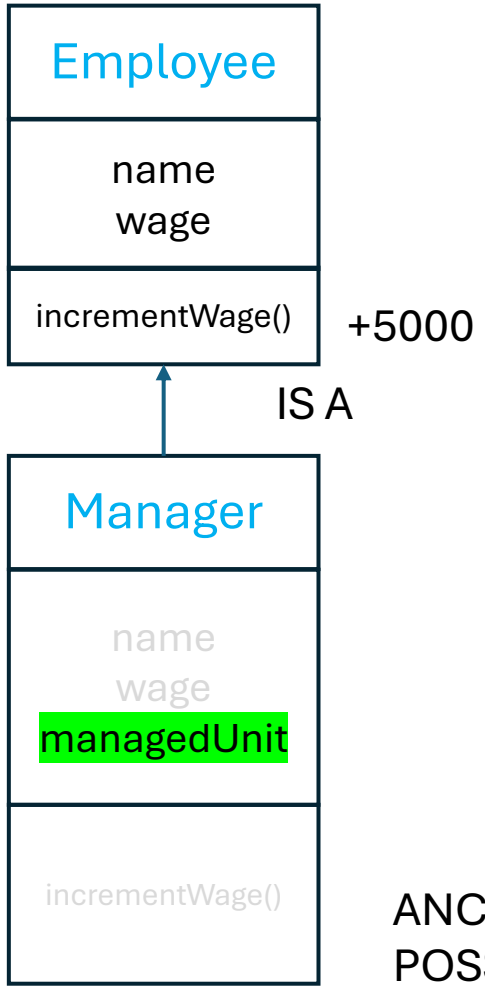
PUO’ ANCHE MODIFICARE CIO’
CHE DEL PADRE “NON VA BENE”


Employee
name
wage
...
IS A
Manager
name
wage
managedUnit
...

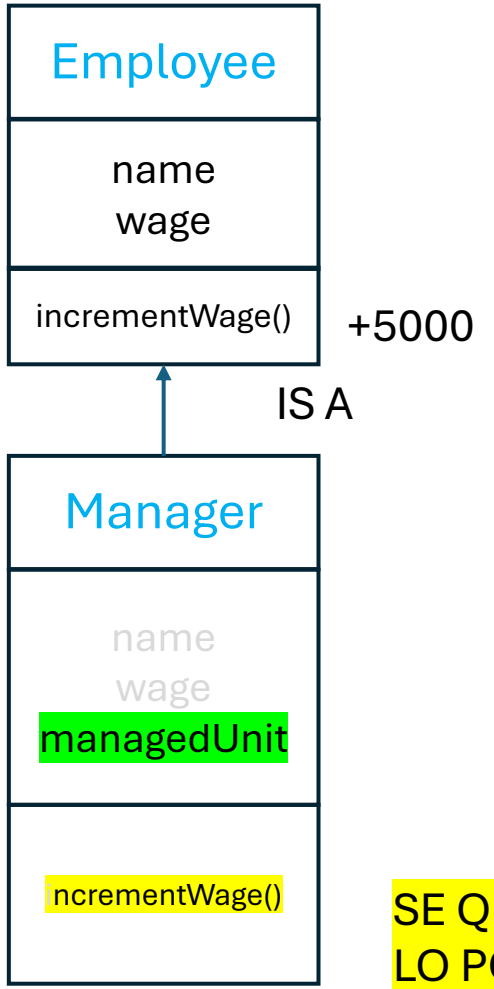


PER COSTRUIRE UN OGGETTO FIGLIO (DI TIPO Manager) E' NECESSARIO COSTRUIRE (PRIMA) LA SUA "PARTE" Employee

PER COSTRUIRE LA "PANCIA" DI TIPO Employee SI USA LA FUNZIONE super() ED IL COSTRUTTORE __init__() DELLA CLASSE PADRE Employee



ANCHE IL MANAGER
POSSIEDE incrementWage()
EREDITATO DA Employee



SE QUEL METODO NON CI VA BENE
LO POSSIAMO CAMBIARE (RIDEFINIRE)
NEL MANAGER (OVERRIDE), ES. PER
INCREMENTARE DI UN VALORE DIVERSO