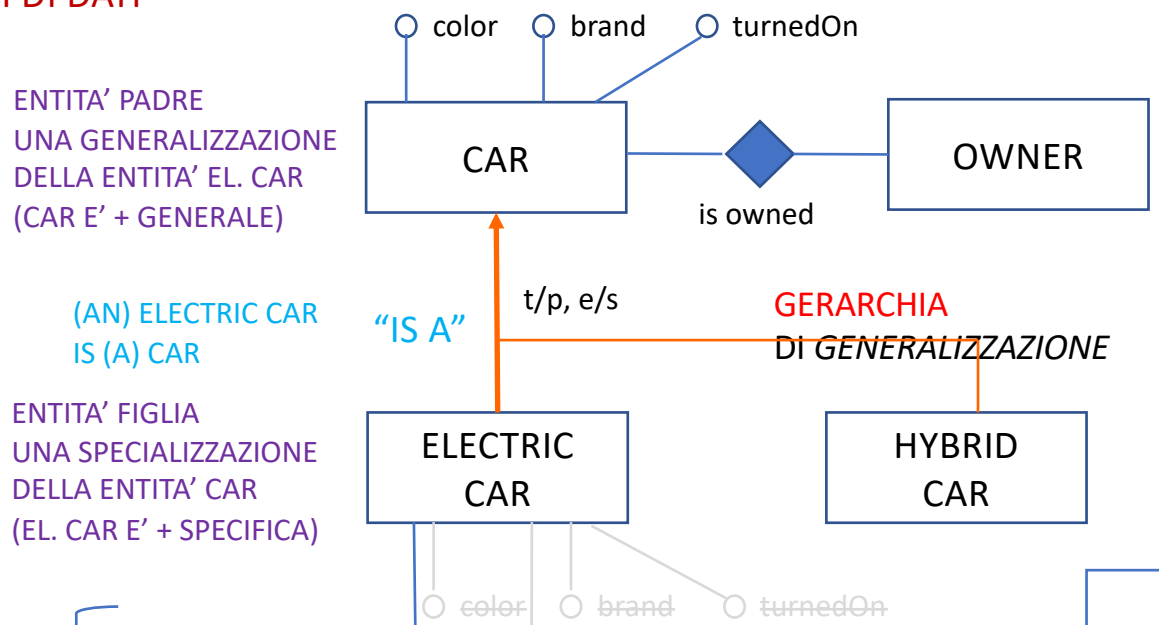
BASI DI DATI

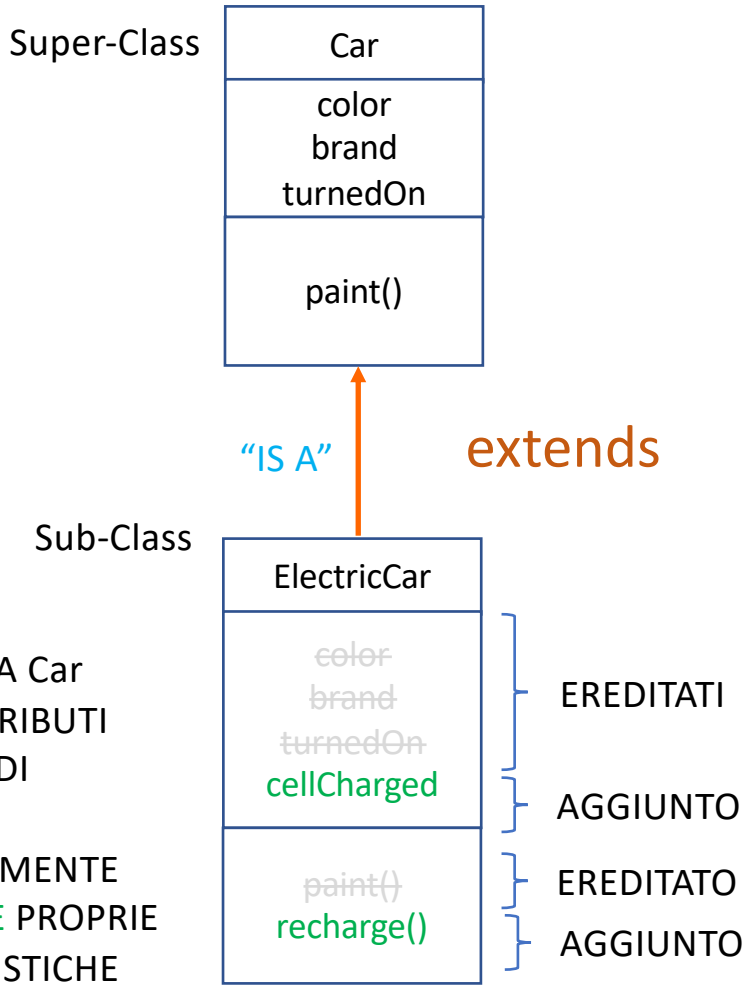


EREDITARIETA' SECONDO LE BASI DI DATI

ELECTRIC CAR
EREDITA DA CAR
- GLI ATTRIBUTI
- LE RELAZIONI

EVENTUALMENTE
AGGIUNGE PROPRIE
CARATTERISTICHE
(ATTR. O RELAZIONI)

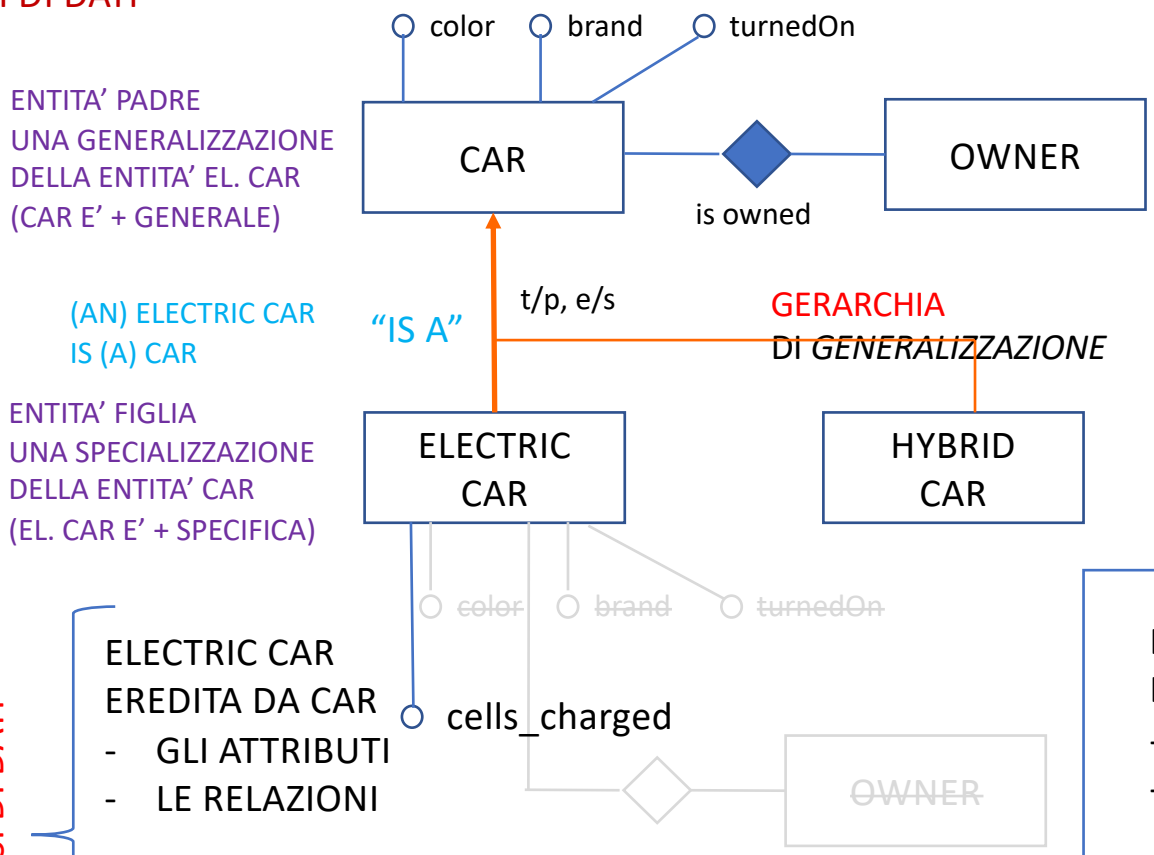
OOP



ElectricCar
EREDITA DA Car
- GLI ATTRIBUTI
- I METODI

EVENTUALMENTE
AGGIUNGE PROPRIE
CARATTERISTICHE
(ATTR. O METODI)

BASI DI DATI

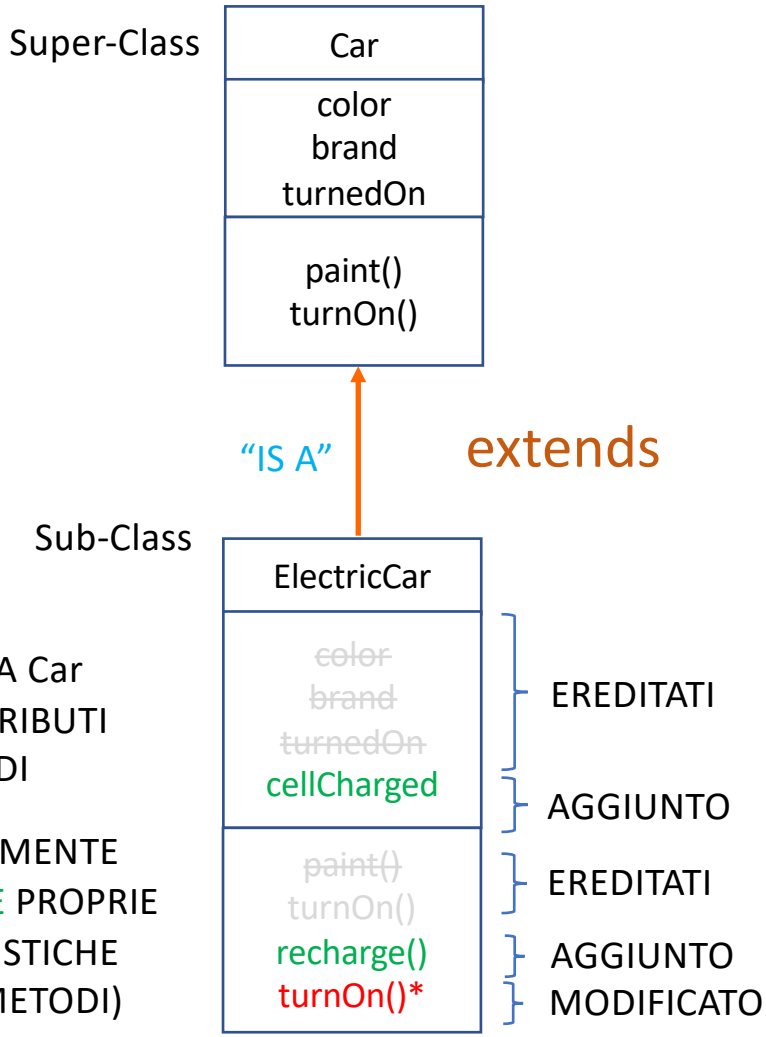


EREDITARIETA' SECONDO LE BASI DI DATI

ELECTRIC CAR
EREDITA DA CAR
- GLI ATTRIBUTI
- LE RELAZIONI

EVENTUALMENTE
AGGIUNGE PROPRIE
CARATTERISTICHE
(ATTR. O RELAZIONI)

OOP



ElectricCar
EREDITA DA Car
- GLI ATTRIBUTI
- I METODI

EVENTUALMENTE
AGGIUNGE PROPRIE
CARATTERISTICHE
(ATTR. O METODI)

CAMBIA CIO' CHE
"NON VA BENE"

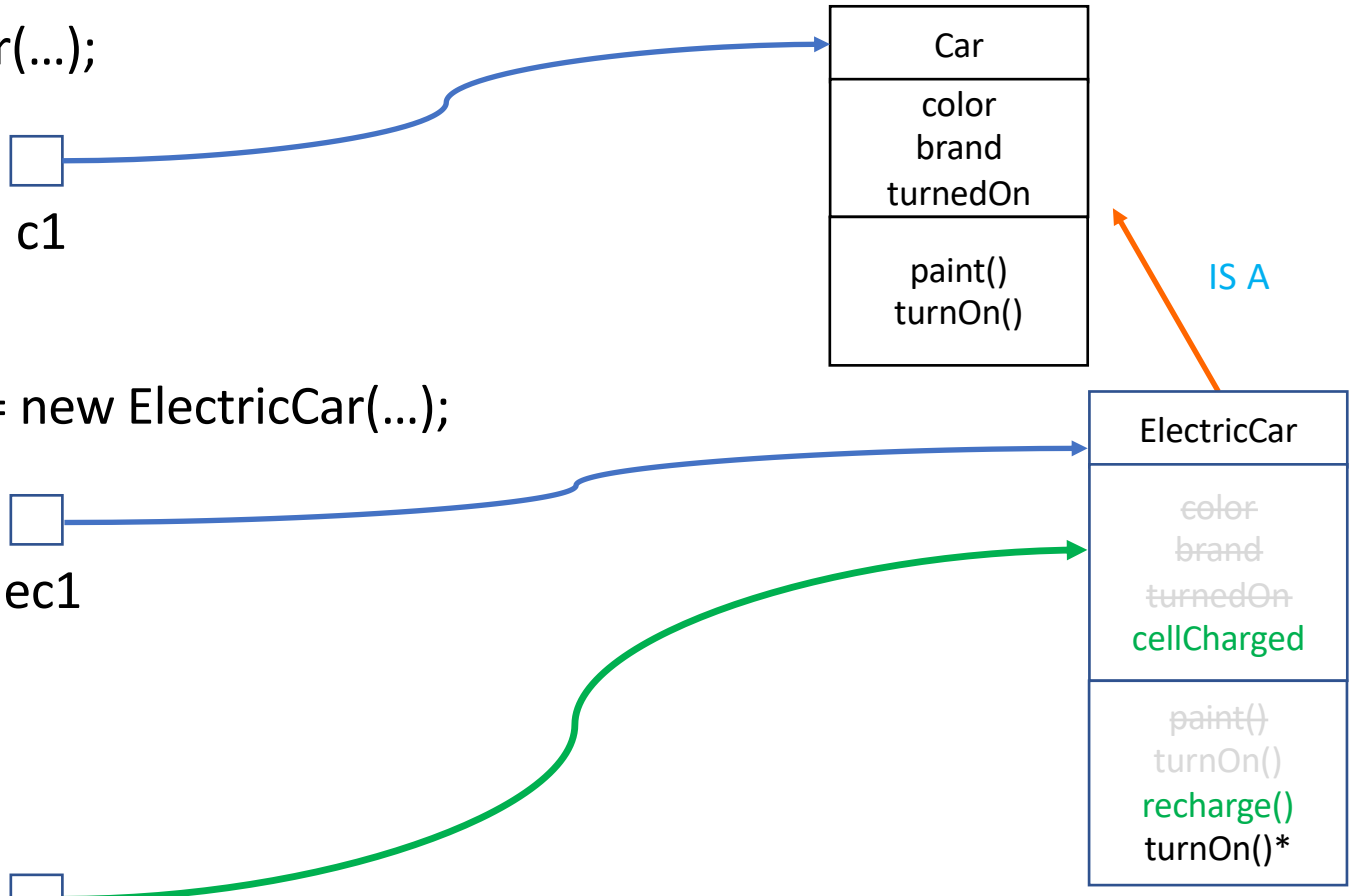
OOP!

ElectricCar ec = new ElectricCar(…);

JAVA UTILIZZA UN MECCANISMO DI COSTRUZIONE DELLE CLASSI FIGLIE CHE POTREBBE ESSERE DETTO "A CIPOLLA"→
PER COSTRUIRE UNA ElectricCar SI DEVE PRIMA COSTRUIRE LA Car (USANDO IL COSTRUTTORE DI Car)
POI …
SI AGGIUNGONO LE CARATTERISTICHE SPECIFICHE DI ElectricCar, es. cellsCharged, recharge(),…
O SI CAMBIA CIO' CHE NON VA BENE

UNA ElectricCar IS A Car

Car c1 = new Car(…);



ElectricCar ec1 = new ElectricCar(…);

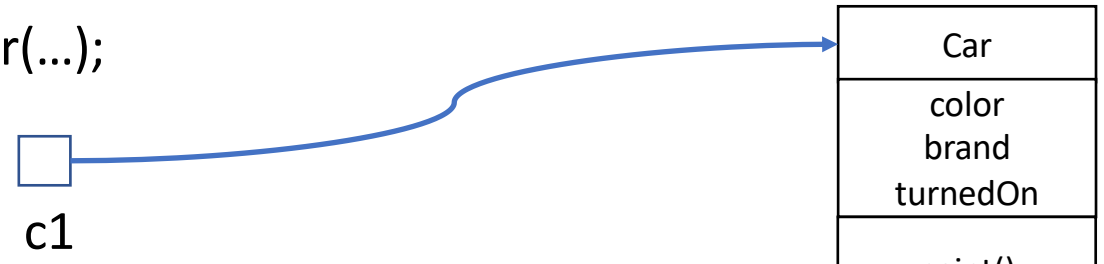
ec1

Car c = ec1;

c

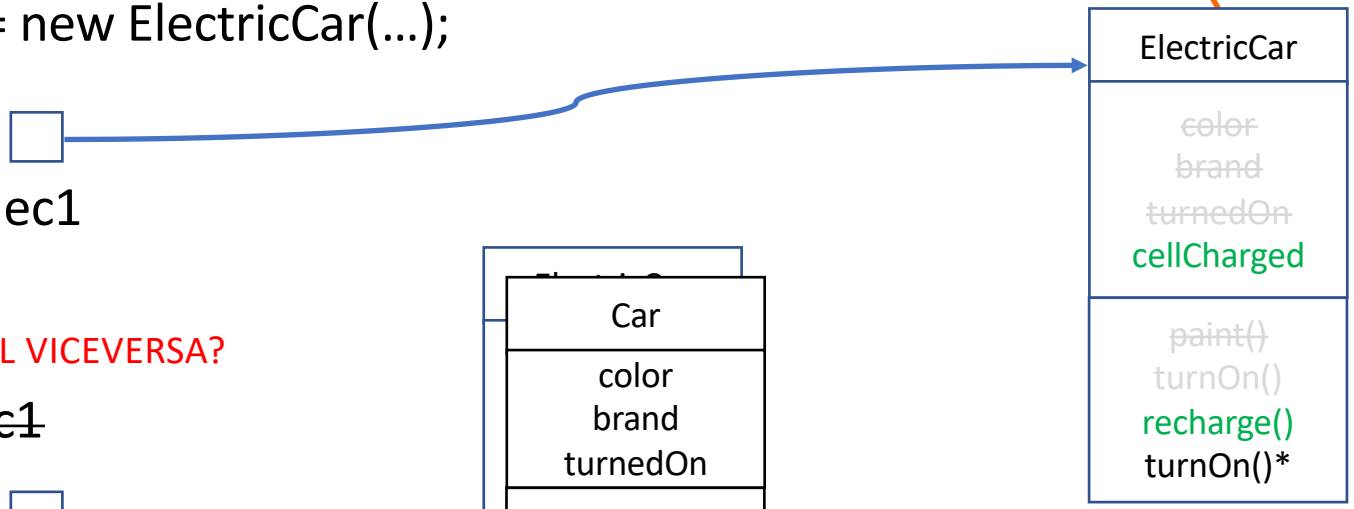
SI PUO' USARE UN RIFERIMENTO DI TIPO Car PER FARE RIFERIMENTO AD UNA ElectricCar

Car c1 = new Car(…);



IS A

ElectricCar ec1 = new ElectricCar(…);



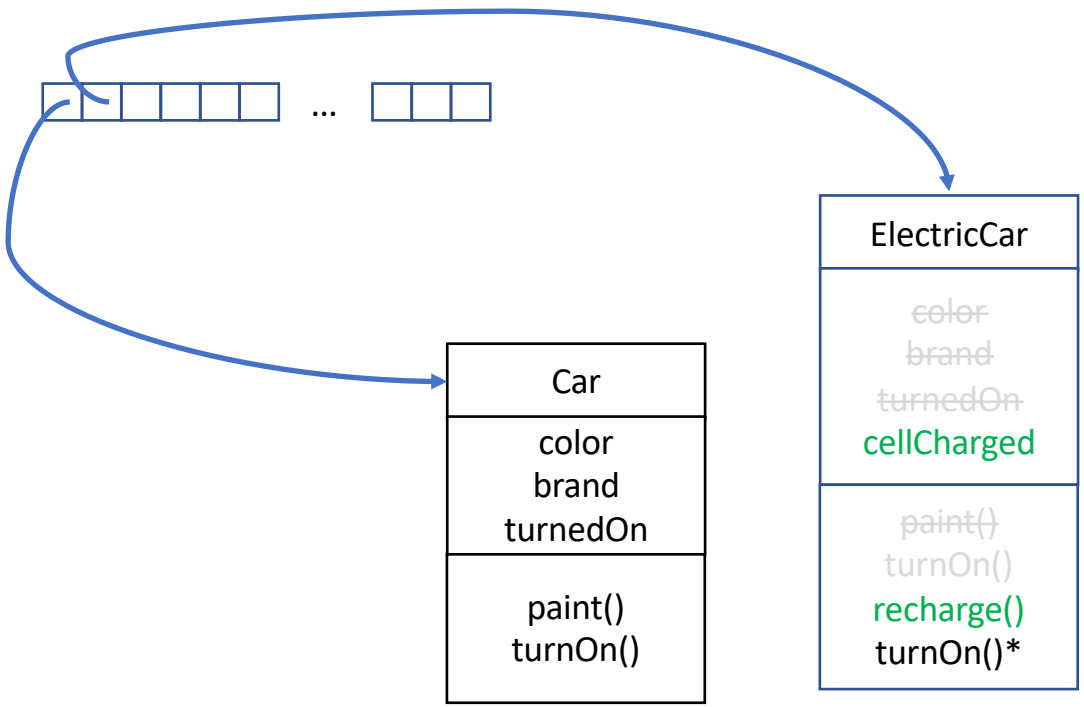
E' POSSIBILE ANCHE IL VICEVERSA?

~~ElectricCar ec = c1~~

NON E' AMMESSO

JAVA NON SAPREBBE COME
INIZIALIZZARE / GESTIRE
GLI ATTRIBUTO SPECIFICI DI CAR

Car cars[] = new Car[100];

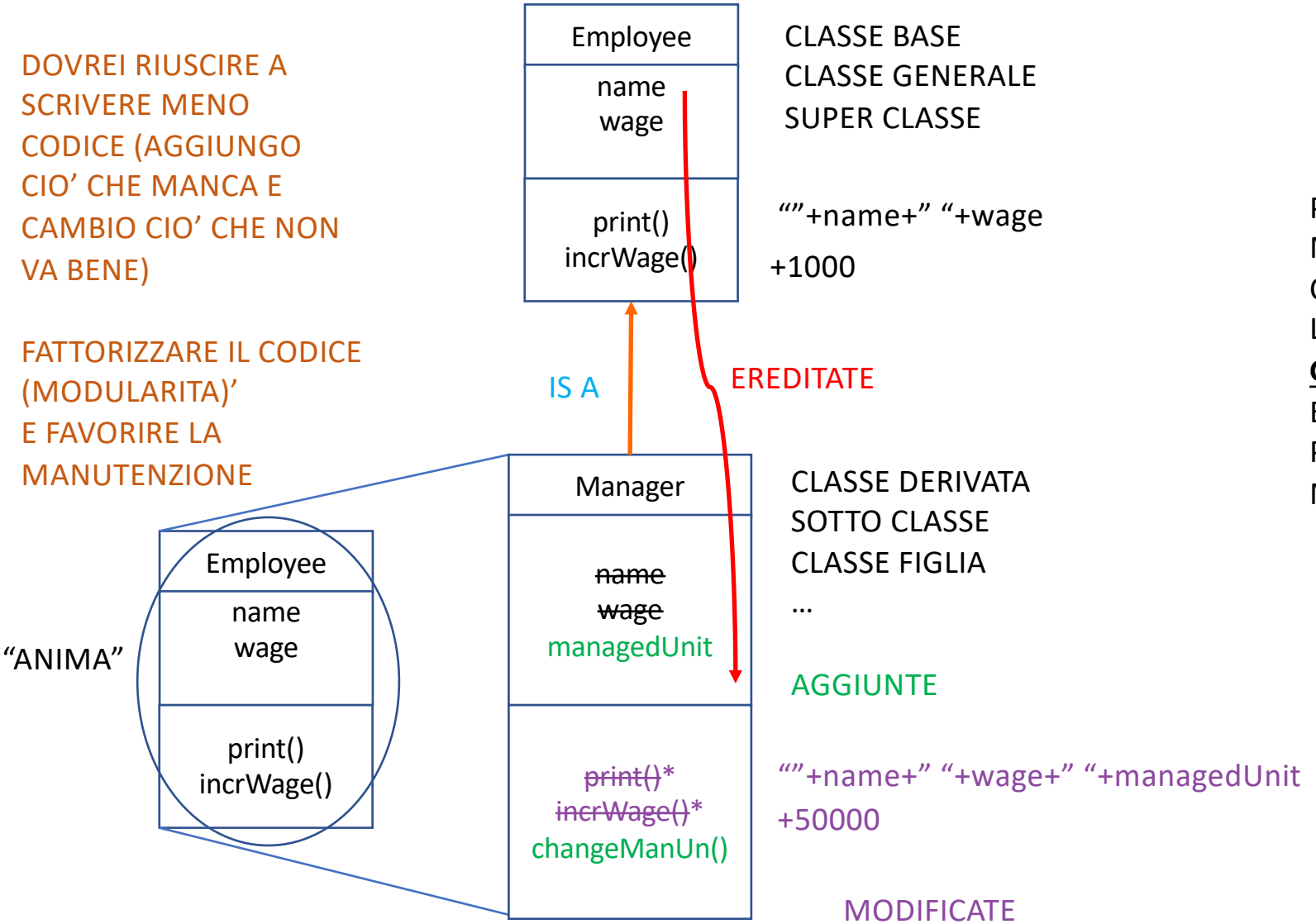


Array di 100 celle atte a contenere riferimenti ad oggetti di tipo Car …
… O SUE SPECIALIZZAZIONI (ElectricCar)

Quando java, a run time, “segue le frecce” (i riferimenti), potra’ trovare (in memoria) sia oggetti Car che ElectricCar. Invocando un metodo sugli oggetti, Java scegliera’ la versione piu’ specifica (se disponibile) oppure, risalendo la gerarchia di classi, versioni meno specifiche. Ad esempio, la versione di turnOn() ridefinita, in override*, in ElectricCar, o quello definite originariamente in Car

DOVREI RIUSCIRE A SCRIVERE MENO CODICE (AGGIUNGO CIO' CHE MANCA E CAMBIO CIO' CHE NON VA BENE)

FATTORIZZARE IL CODICE (MODULARITA)' E FAVORIRE LA MANUTENZIONE



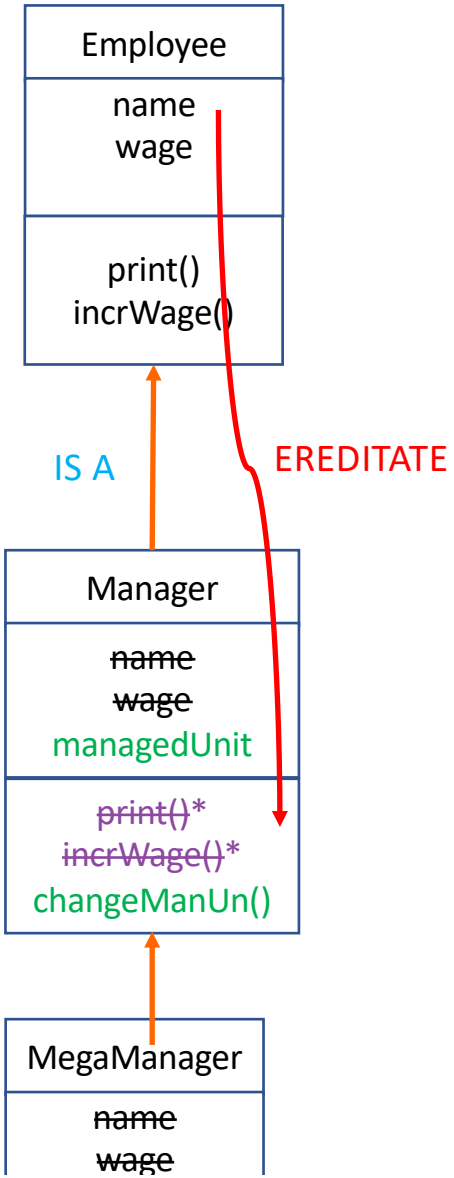
PER COSTRUIRE UN MANAGER, JAVA COSTRUISCE PRIMA LA SUA "ANIMA" **GENERICA** EMPLOYEE E POI LA SUA PARTE **SPECIFICA** MANAGER

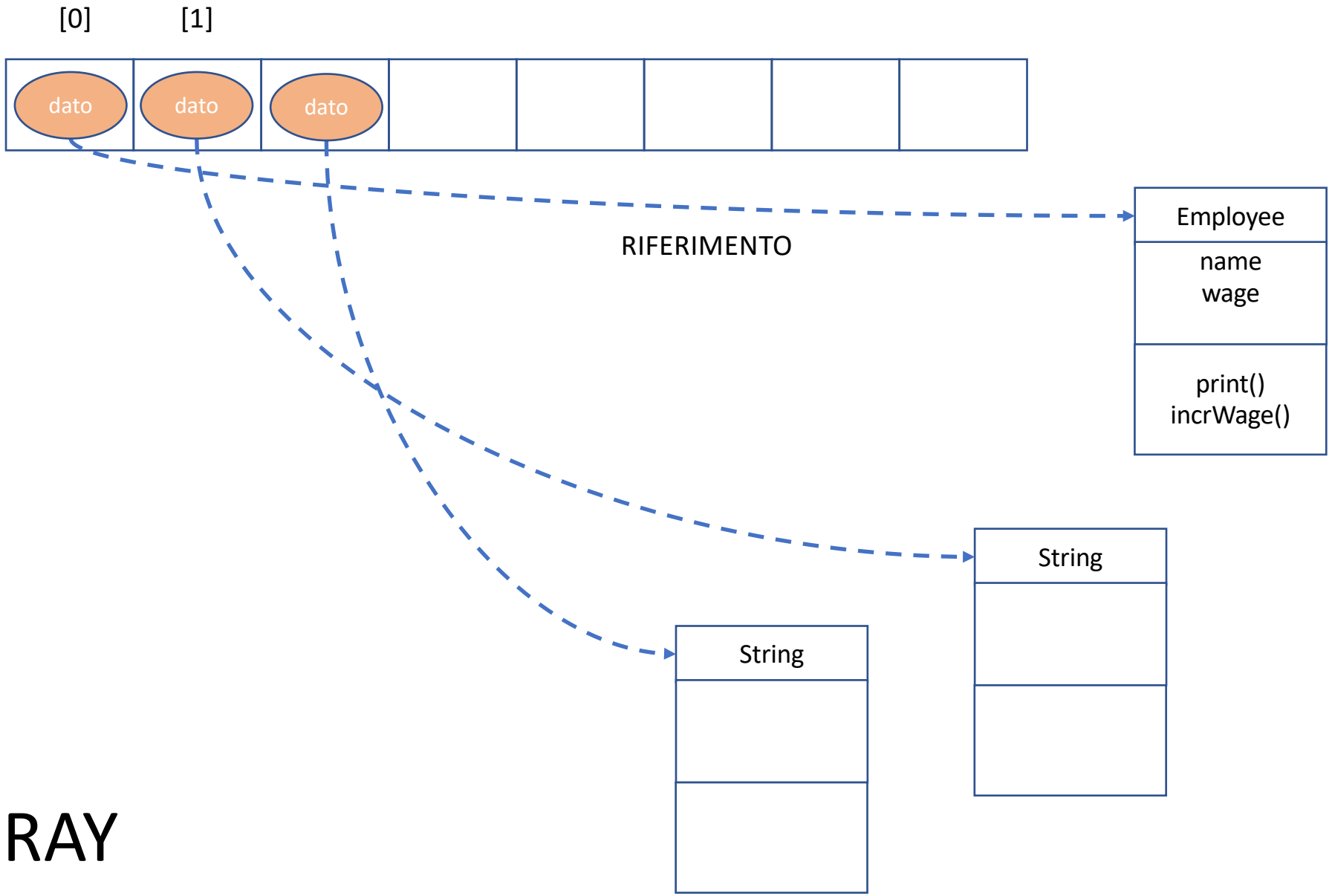
```
Employee e = new Employee();

Manager m = new Manager();

Employee e = m; // LECITA
```

PERCHE' IN FONDO UN MANAGER E' (ANCHE) UN EMPLOYEE

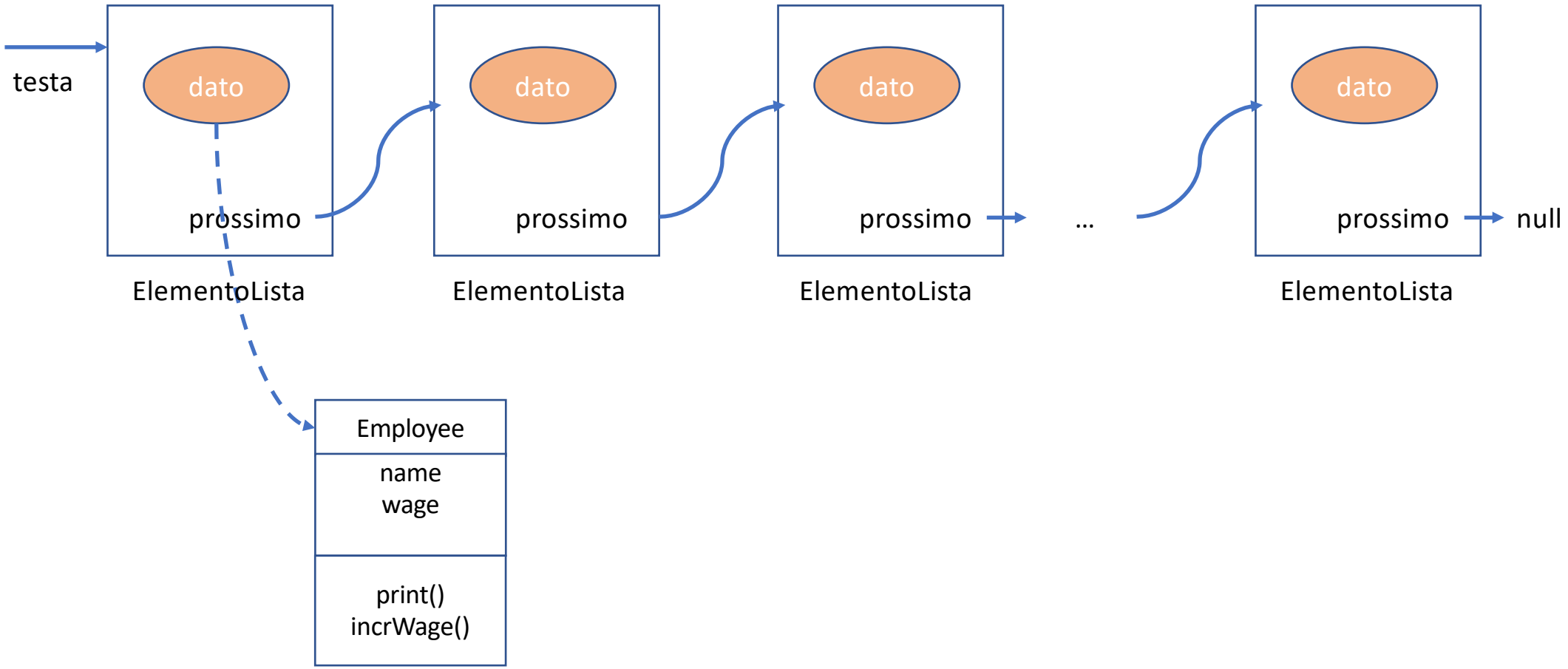
```
Employee emps[] = new Employee[100];

emps[0] = e;
emps[1] = m;
emps[2] = new MegaManager();
```

COLLEZIONE DI DATI "GENERICA" NEL SENSO CHE POSSO RIMANDARE LA SCELTA DEGLI OGGETTI DA MEMORIZZARE (A PATTO CHE DERIVINO DA EMPLOYEE)




[0]
[1]
dato
dato
dato
RIFERIMENTO
Employee
name
wage
print()
incrWage()
String
String

ARRAY

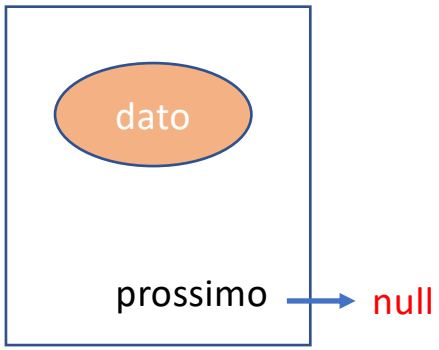


LISTA “LINKATA”

LISTA VUOTA

LISTA “LINKATA”

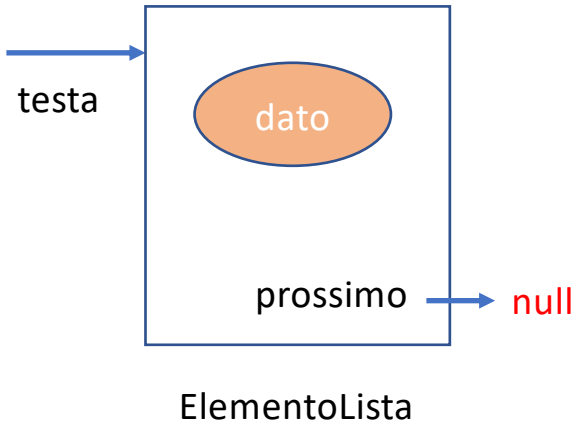
NUOVO ELEMENTO
DA INSERIRE



ElementoLista

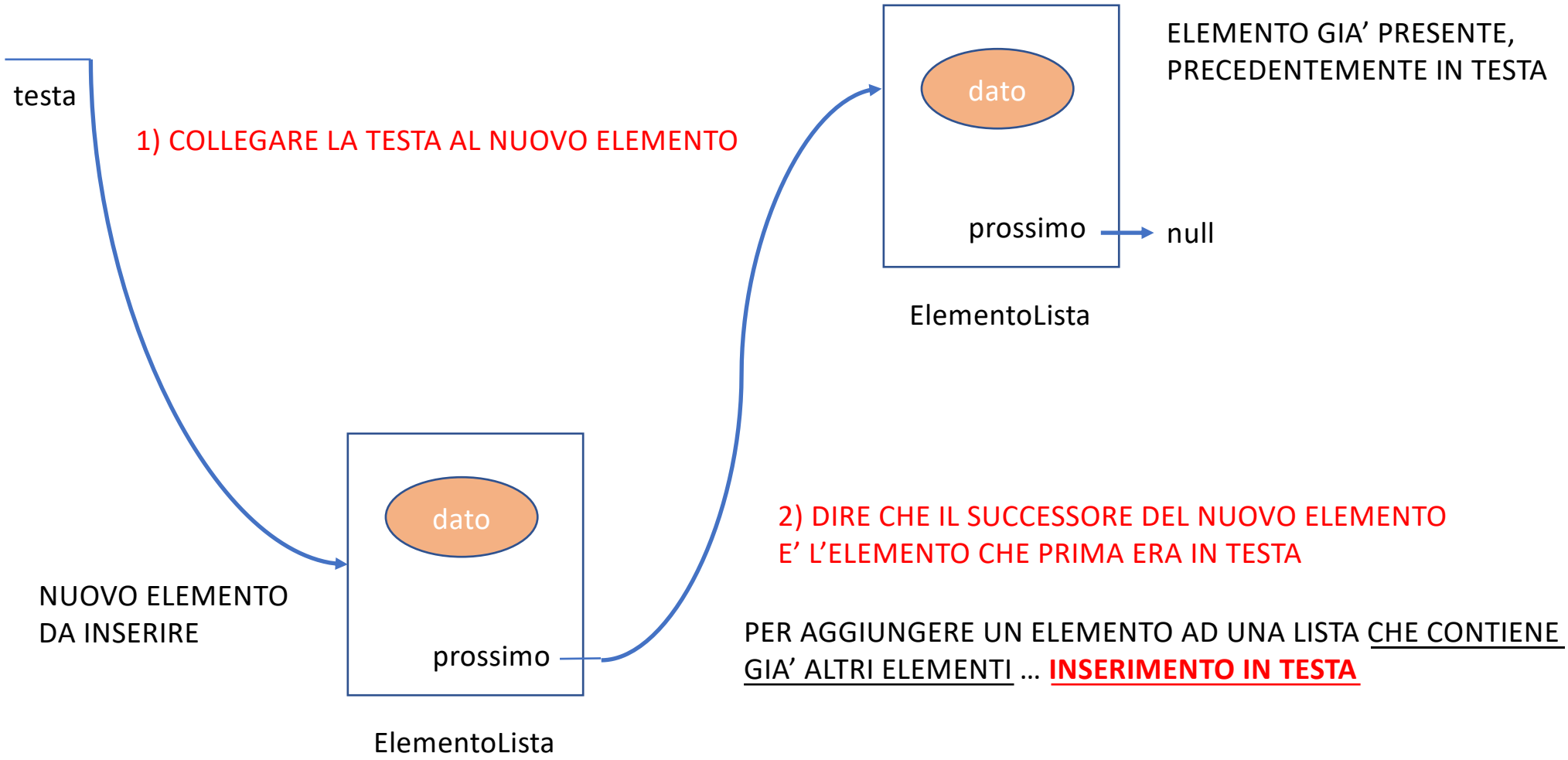
PER AGGIUNGERE UN ELEMENTO AD UNA **LISTA VUOTA**,
AVENDO A DISPOSIZIONE L'ELEMENTO …

LISTA "LINKATA"

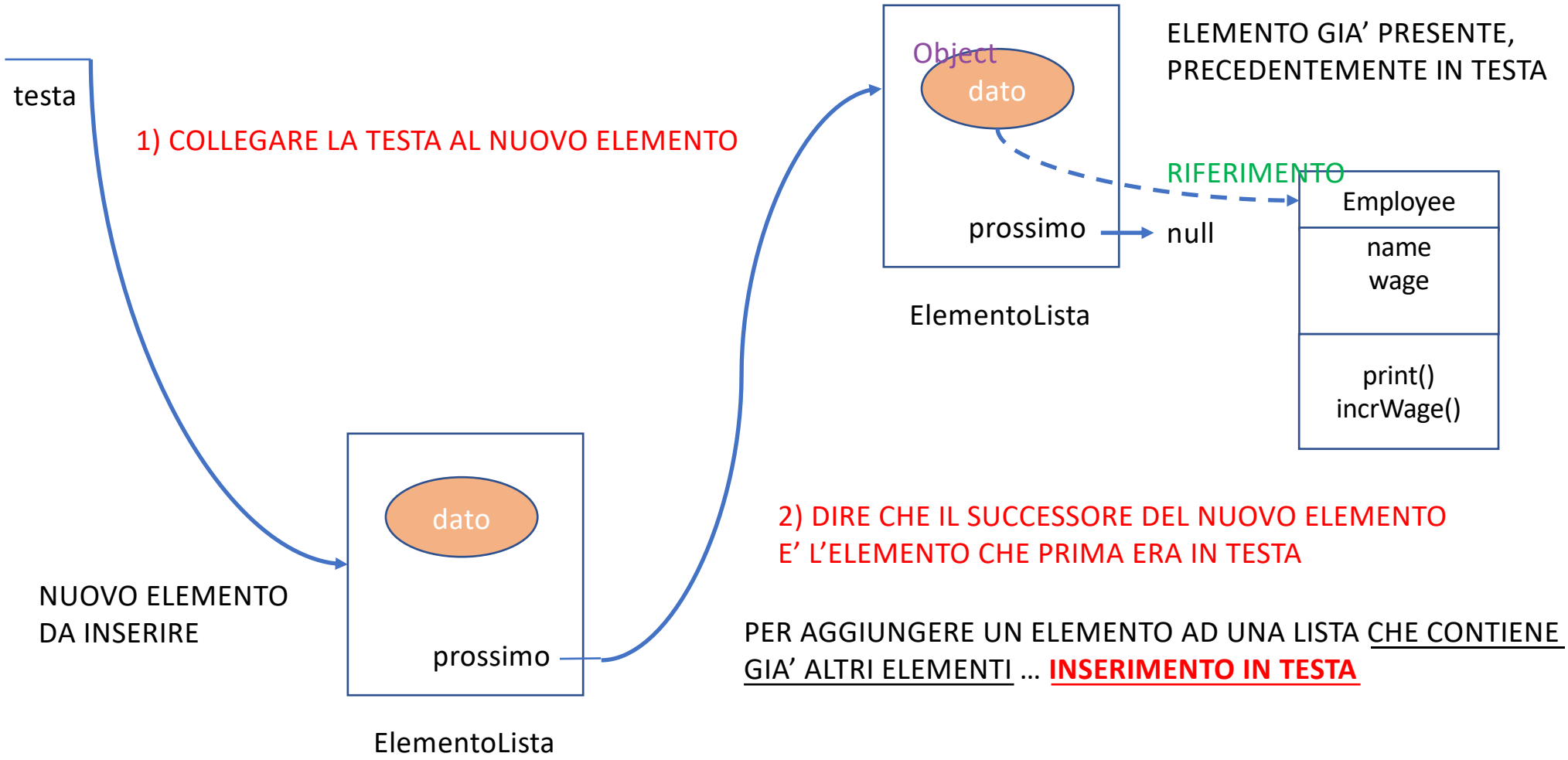


PER AGGIUNGERE UN ELEMENTO AD UNA LISTA VUOTA,
AVENDO A DISPOSIZIONE L'ELEMENTO … LO SI AGGIUNGE

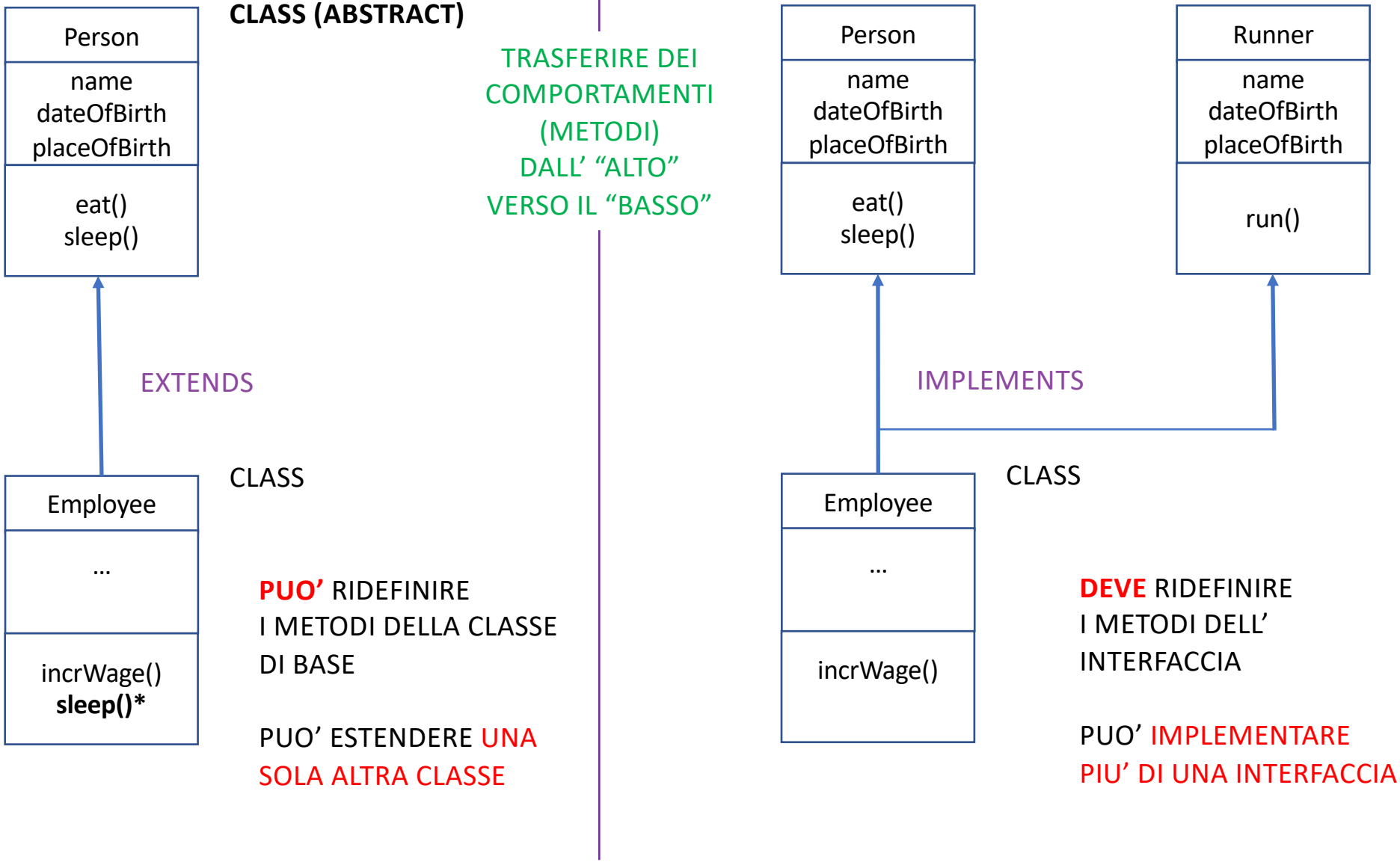
LISTA "LINKATA"



LISTA "LINKATA"



LISTA "LINKATA"


CLASS (ABSTRACT)
Person
name
dateOfBirth
placeOfBirth
eat()
sleep()
EXTENDS
CLASS
Employee
...
incrWage()
sleep()*
PUO' RIDEFINIRE
I METODI DELLA CLASSE
DI BASE
PUO' ESTENDERE UNA
SOLA ALTRA CLASSE
TRASFERIRE DEI
COMPORTAMENTI
(METODI)
DALL' "ALTO"
VERSO IL "BASSO"
INTERFACE
Person
name
dateOfBirth
placeOfBirth
eat()
sleep()
INTERFACE
Runner
name
dateOfBirth
placeOfBirth
run()
IMPLEMENTS
CLASS
Employee
...
incrWage()
DEVE RIDEFINIRE
I METODI DELL'
INTERFACCIA
PUO' IMPLEMENTARE
PIU' DI UNA INTERFACCIA