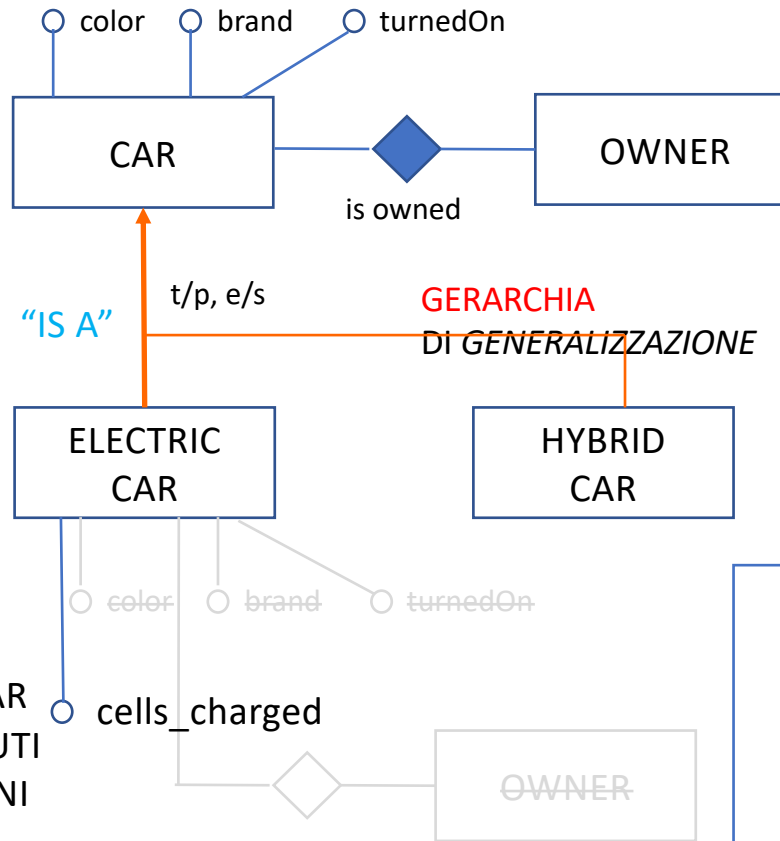


## BASI DI DATI

ENTITA' PADRE  
UNA GENERALIZZAZIONE  
DELLA ENTITA' EL. CAR  
(CAR E' + GENERALE)

(AN) ELECTRIC CAR  
IS (A) CAR

ENTITA' FIGLIA  
UNA SPECIALIZZAZIONE  
DELLA ENTITA' CAR  
(EL. CAR E' + SPECIFICA)



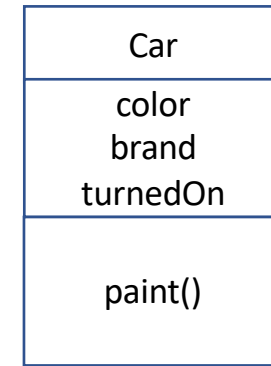
EREDITARIETA' SECONDO  
LE BASI DI DATI

ELECTRIC CAR  
EREDITA DA CAR  
- GLI ATTRIBUTI  
- LE RELAZIONI

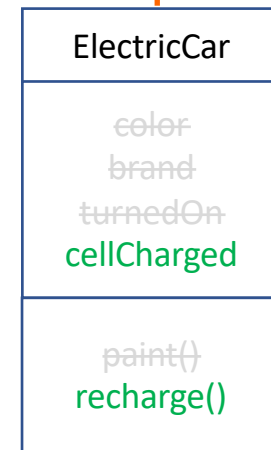
EVENTUALMENTE  
AGGIUNGE PROPRIE  
CARATTERISTICHE  
(ATTR. O RELAZIONI)

## OOP

Super-Class



Sub-Class



"IS A"

extends

ElectricCar  
EREDITA DA Car  
- GLI ATTRIBUTI  
- I METODI

EVENTUALMENTE  
AGGIUNGE PROPRIE  
CARATTERISTICHE  
(ATTR. O METODI)

EREDITATI

AGGIUNTO

EREDITATO

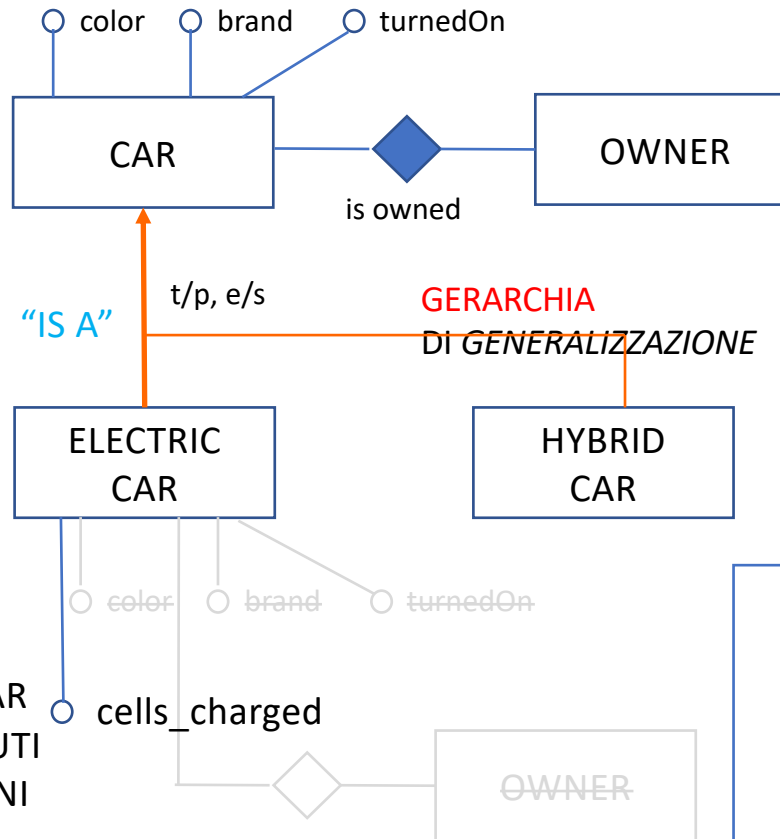
AGGIUNTO

## BASI DI DATI

ENTITA' PADRE  
UNA GENERALIZZAZIONE  
DELLA ENTITA' EL. CAR  
(CAR E' + GENERALE)

(AN) ELECTRIC CAR  
IS (A) CAR

ENTITA' FIGLIA  
UNA SPECIALIZZAZIONE  
DELLA ENTITA' CAR  
(EL. CAR E' + SPECIFICA)



"IS A"

t/p, e/s

GERARCHIA  
DI GENERALIZZAZIONE

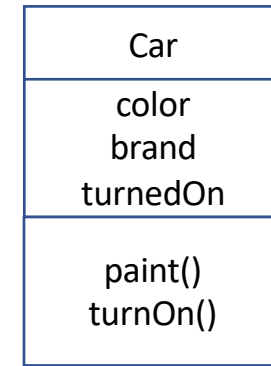
EREDITARIETA' SECONDO  
LE BASI DI DATI

ELECTRIC CAR  
EREDITA DA CAR  
- GLI ATTRIBUTI  
- LE RELAZIONI

EVENTUALMENTE  
AGGIUNGE PROPRIE  
CARATTERISTICHE  
(ATTR. O RELAZIONI)

## OOP

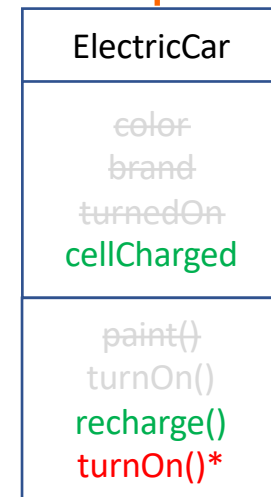
Super-Class



"IS A"

extends

Sub-Class



ElectricCar  
EREDITA DA Car  
- GLI ATTRIBUTI  
- I METODI

EVENTUALMENTE  
AGGIUNGE PROPRIE  
CARATTERISTICHE  
(ATTR. O METODI)

EREDITATI

AGGIUNTO

EREDITATI

AGGIUNTO  
MODIFICATO

CAMBIA CIO' CHE  
"NON VA BENE"

← OOP!

ElectricCar ec = new ElectricCar(...);

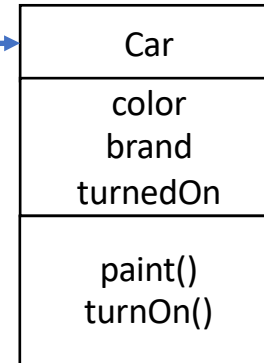


JAVA UTILIZZA UN MECCANISMO DI COSTRUZIONE DELLE CLASSI FIGLIE CHE POTREBBE ESSERE DETTO "A CIPOLLA" →  
PER COSTRUIRE UNA `ElectricCar` SI DEVE PRIMA COSTRUIRE LA `Car` (USANDO IL COSTRUTTORE DI `Car`)  
POI ...  
SI AGGIUNGONO LE CARATTERISTICHE SPECIFICHE DI `ElectricCar`, es. `cellsCharged`, `recharge()`,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

UNA `ElectricCar` IS A `Car`

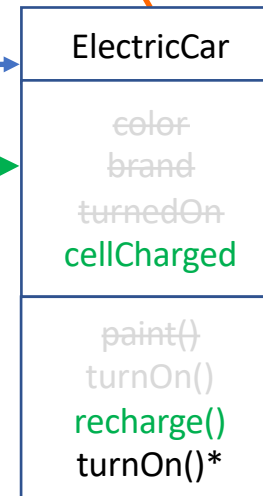
Car c1 = new Car(...);

  
c1



ElectricCar ec1 = new ElectricCar(...);

  
ec1



IS A

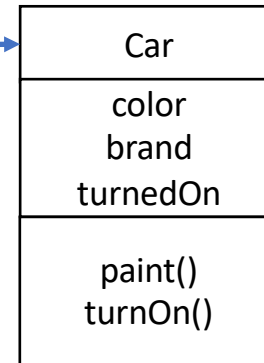
Car c = ec1;

  
c

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

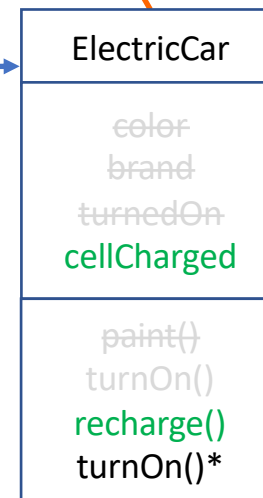
Car c1 = new Car(...);

  
c1



ElectricCar ec1 = new ElectricCar(...);

  
ec1



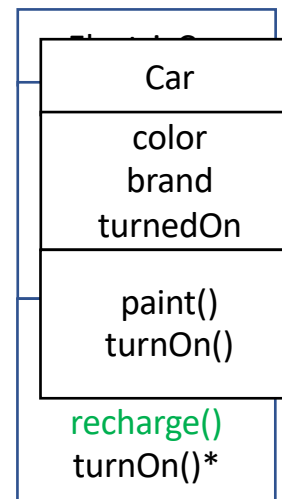
IS A

E' POSSIBILE ANCHE IL VICEVERSA?

~~ElectricCar ec = c1~~

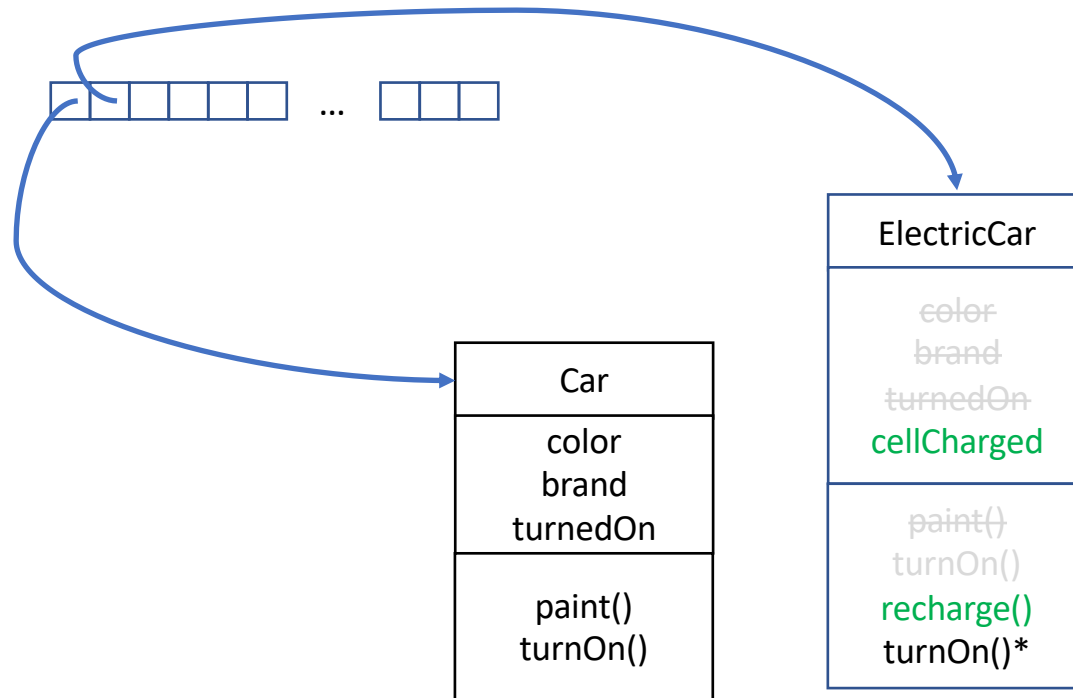
NON E' AMMESSO

  
ec



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

```
Car cars[] = new Car[100];
```



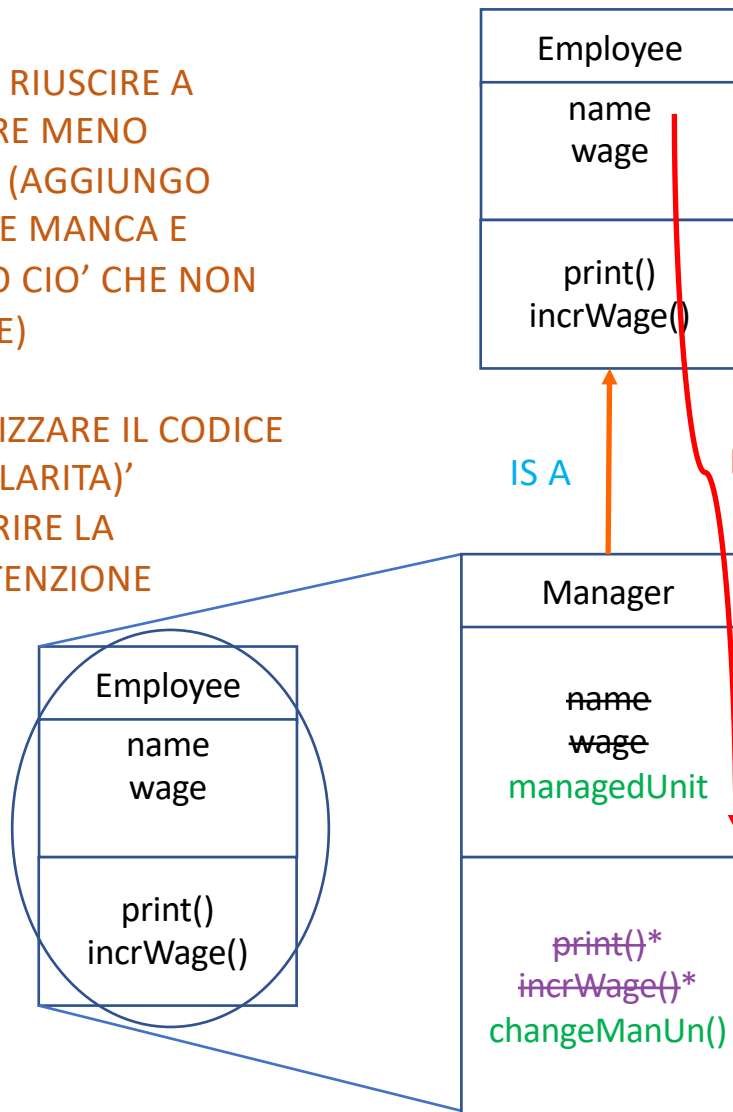
Array di 100 celle atte a  
contenere riferimenti ad  
oggetti di tipo Car ...  
... O SUE SPECIALIZZAZIONI  
(ElectricCar)

Quando java, a run time, “segue le frecce” (i riferimenti), potrà trovare (in memoria) sia oggetti Car che ElectricCar. Invocando un metodo sugli oggetti, Java sceglierà la versione più 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 definito 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

"ANIMA"



CLASSE BASE  
CLASSE GENERALE  
SUPER CLASSE

""+name+" "+wage  
+1000

IS A

EREDITATE

CLASSE DERIVATA  
SOTTO CLASSE  
CLASSE FIGLIA

...

AGGIUNTE

""+name+" "+wage+" "+managedUnit  
+50000

MODIFICATE

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

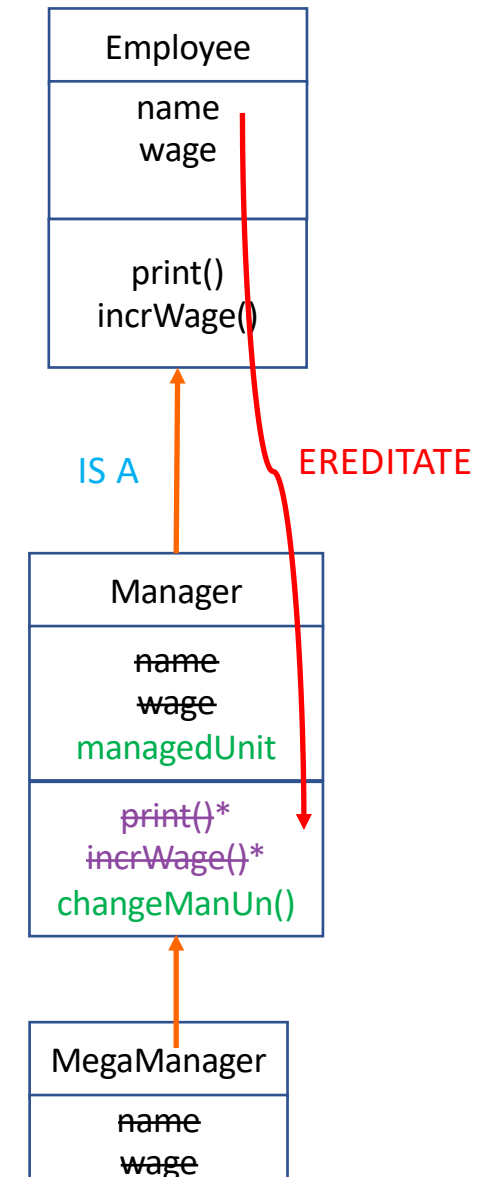
```
Employee emps[] = new Employee[100];
```

```
emps[0] = e;
```

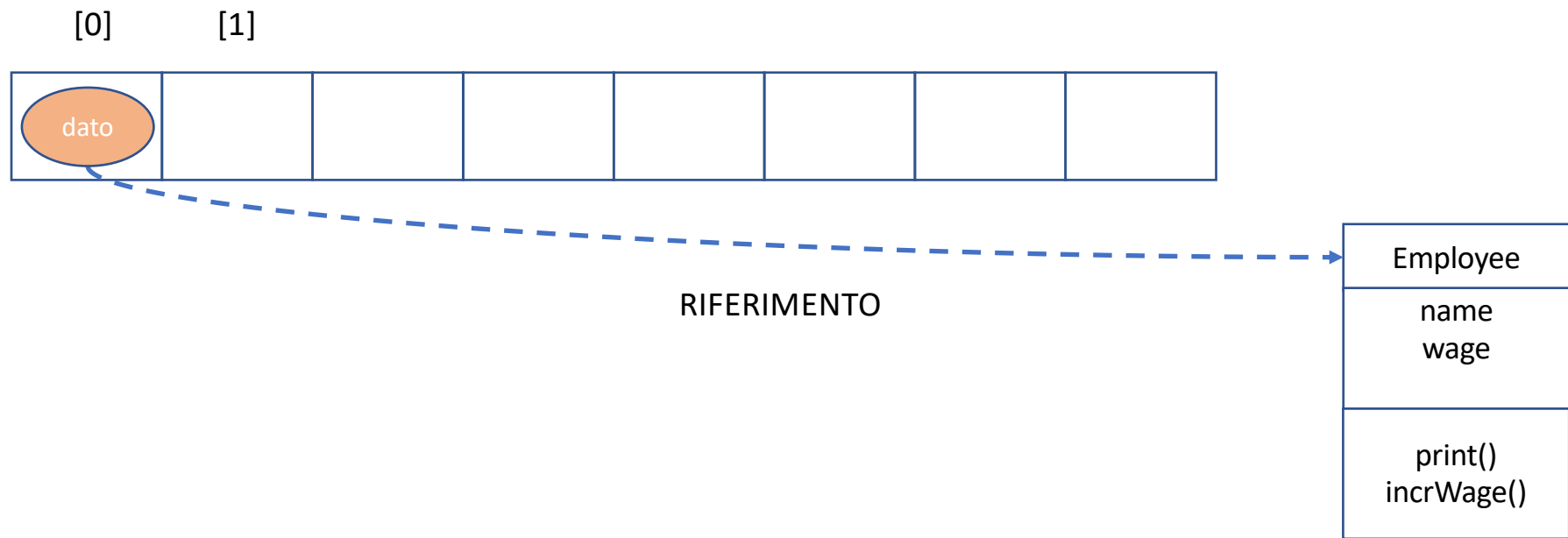
```
emps[1] = m;
```

```
emps[2] = new MegaManager();
```

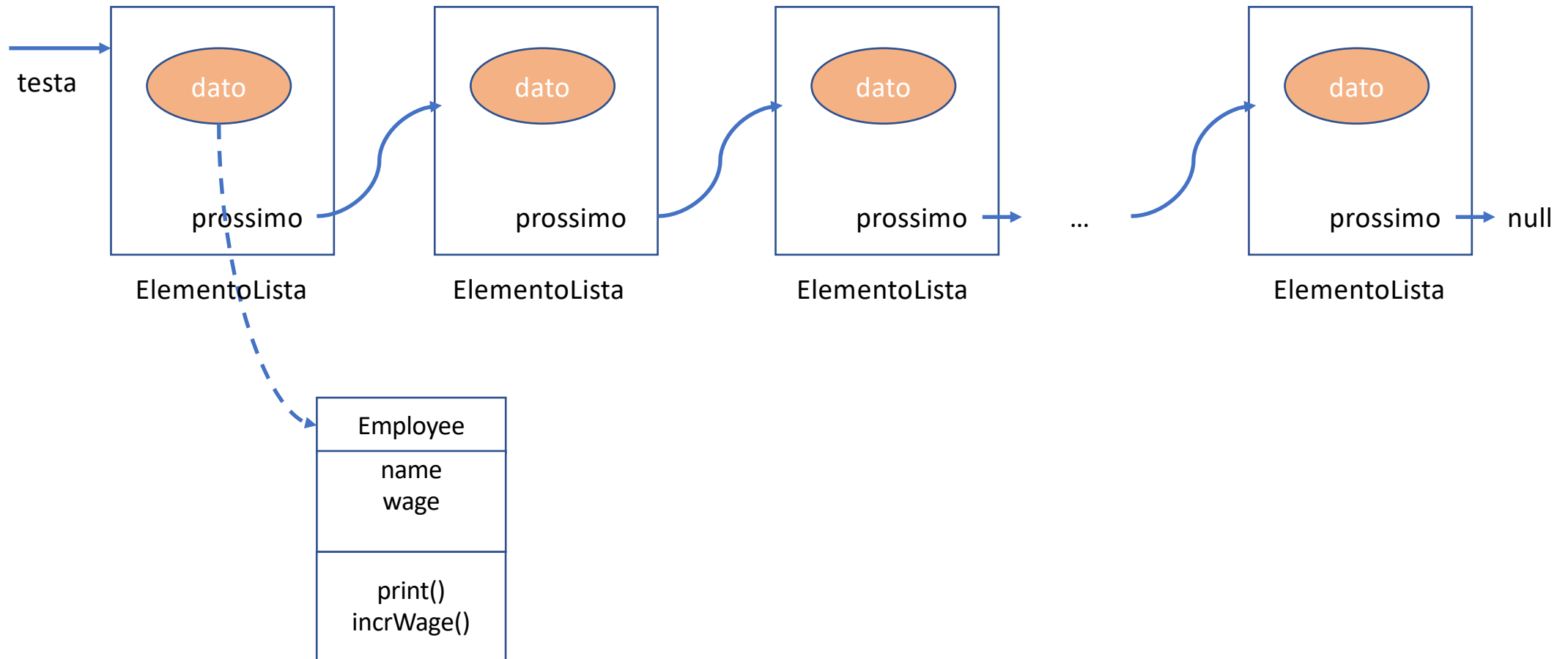
COLLEZIONE DI DATI "GENERICA" NEL SENSO CHE  
POSSO RIMANDARE LA SCELTA DEGLI OGGETTI DA  
MEMORIZZARE (A PATTO CHE DERIVINO DA EMPLOYEE)







ARRAY



# LISTA "LINKATA"

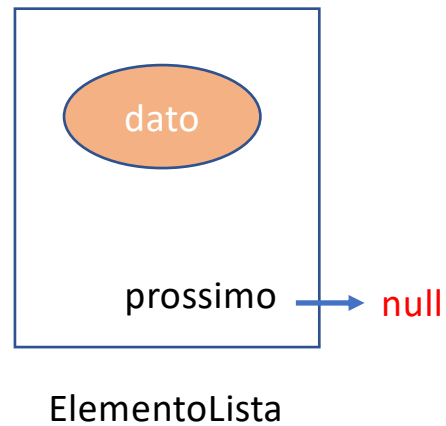
→ null  
testa

LISTA VUOTA

LISTA “LINKATA”

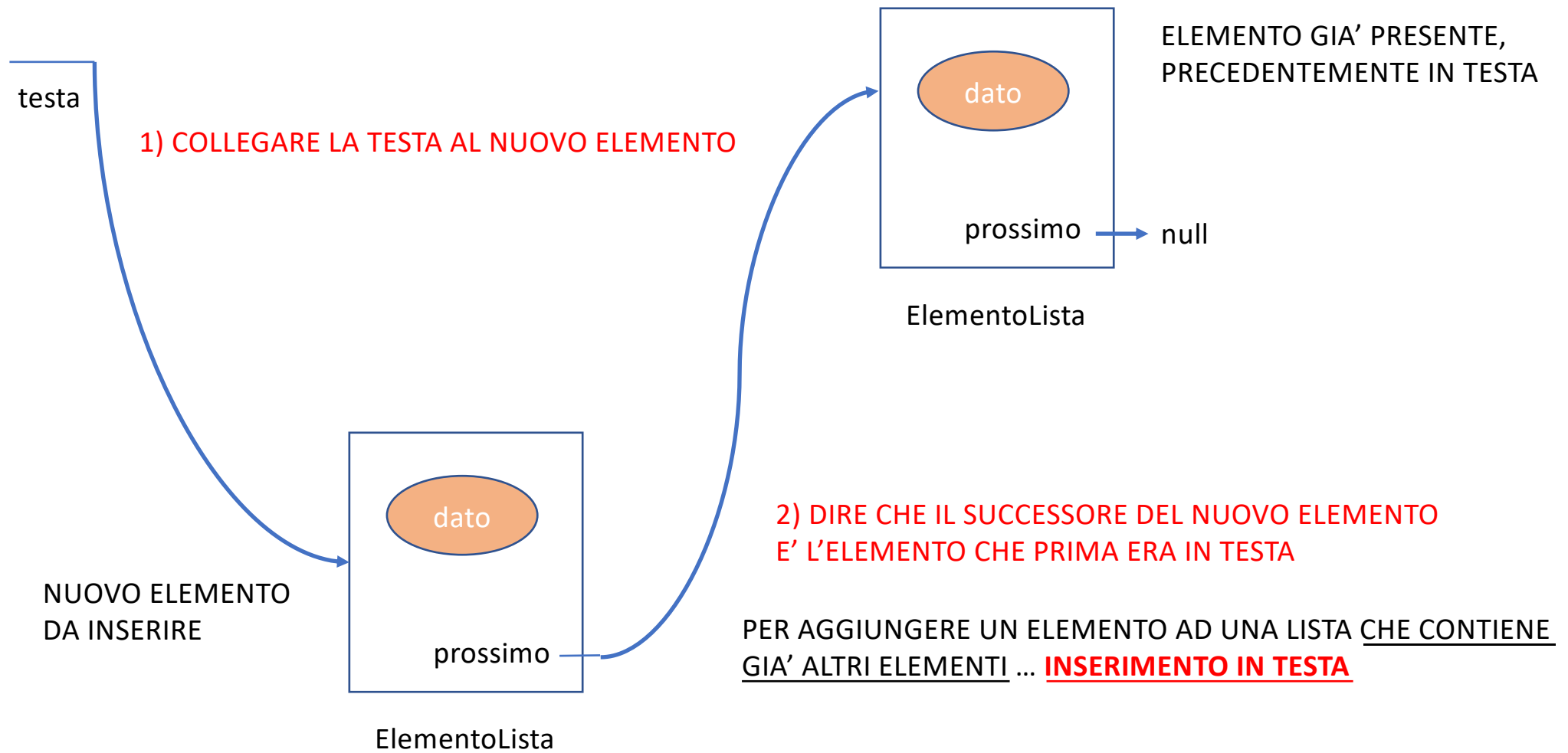


NUOVO ELEMENTO  
DA INSERIRE



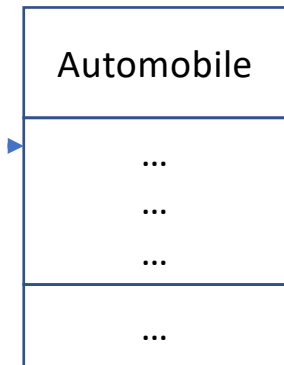
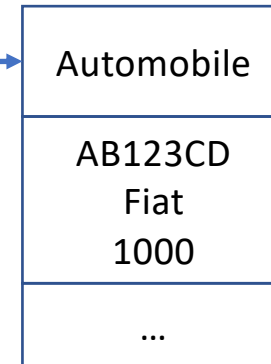
PER AGGIUNGERE UN ELEMENTO AD UNA LISTA VUOTA,  
AVENDO A DISPOSIZIONE L'ELEMENTO ...

# LISTA "LINKATA"



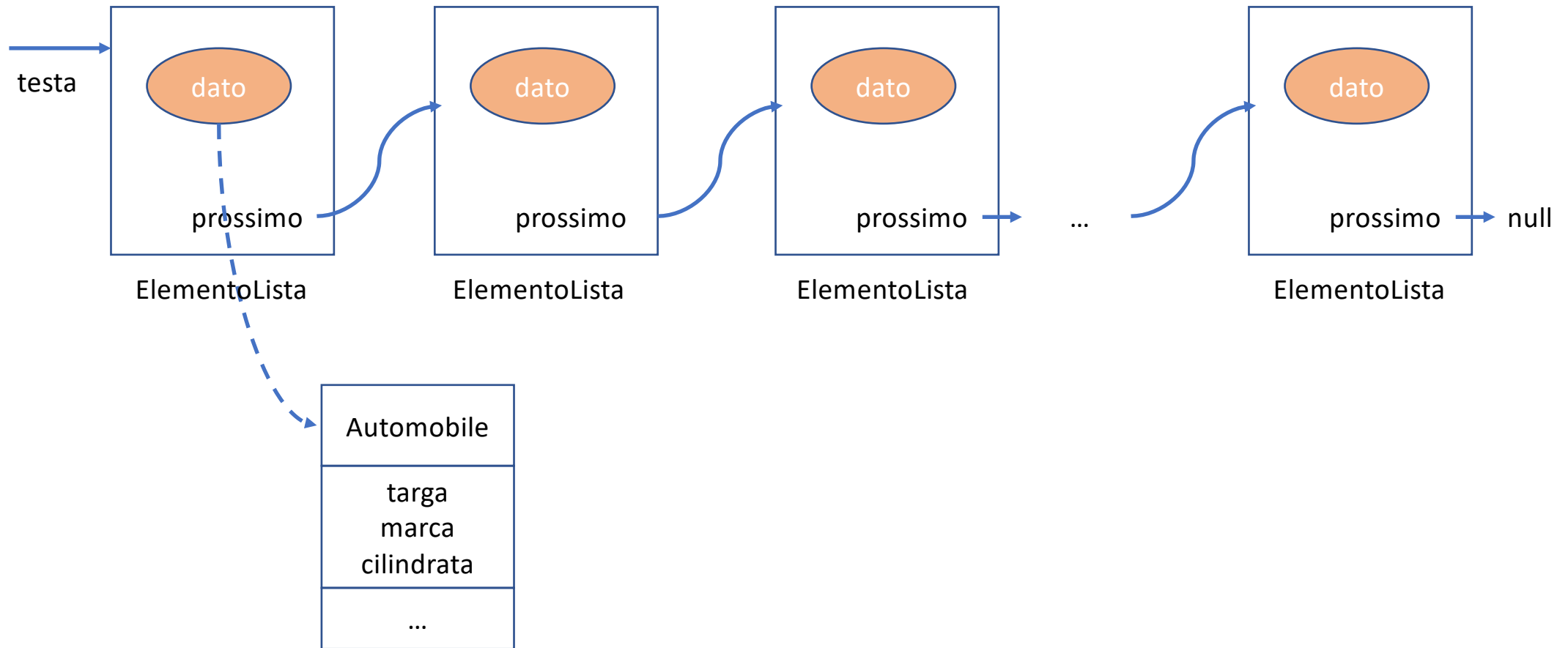
# LISTA "LINKATA"

K	V
<String>	<Automobile>
Chiave	Valori
AB123CD	Fiat, 1000      rif
AA222CC	VW, 1600      rif



MAPPA (JCF è Map)

LA STRUTTURA DATI  
DI TIPO MAPPA E' PARTICOLARMENTE  
VANTAGGIOSA PER EFFETTUARE  
ACCESSI AI DATI (VALUES) DATA  
LA CHIAVE (KEY)  
-> CONTENITORE ASSOCIATIVO



# LISTA LINKATA o LinkedList nel JCF